ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ

Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Санкт-Петербургский государственный университет телекоммуникаций
им. проф. М.А. Бонч-Бруевича»

На правах рукописи

Тихонов Сергей Владимирович

ИССЛЕДОВАНИЕ И РАЗРАБОТКА МОДИФИКАЦИЙ АППАРАТНО-РЕАЛИЗОВАННЫХ ЗАЩИТНЫХ БЛОКОВЫХ ПРЕОБРАЗОВАНИЙ, УСТОЙЧИВЫХ К ПОБОЧНЫМ АТАКАМ ПО ЦЕПЯМ ЭЛЕКТРОПИТАНИЯ

Специальность 05.13.19 – Методы и системы защиты информации,
информационная безопасность

Диссертация на соискание ученой степени
кандидата технических наук

Научный руководитель
доктор технических наук , профессор
Коржик Валерий Иванович

Санкт-Петербург – 2017

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ 6

ГЛАВА 1. ОБЗОР ФИЗИЧЕСКИХ И ТЕОРЕТИЧЕСКИХ ОСНОВ ПОБОЧНЫХ АТАК ПО ЦЕПЯМ ЭЛЕКТРОПИТАНИЯ 12

1.1 Краткая характеристика атак по побочным каналам 14

1.2 Физические основы побочных атак по цепям электропитания 16

1.3 Общее устройство анализируемых чипов 21

1.4 Математические модели «утечки» информации по цепи электропитания 25

1.5 Снятие данных об энергопотреблении чипа 28

1.5.1 Общее устройство измерительной установки 28

1.5.2 Требования к характеристикам измерительной установки 31

1.6 Теоретические основы построения побочных атак по цепям электропитания (простой анализ мощности) 33

1.7 Теоретические основы построения побочных атак по цепям электропитания (дифференциальный анализ мощности) 41

1.7.1 Обобщённый алгоритм атаки DPA 42

1.7.2 Аспекты реализации атаки DPA с расчётом корреляционных векторов 45

1.8 Краткая характеристика основных методов защиты от DPA 52

1.9 Выводы по главе 56

ГЛАВА 2. ЭКСПЕРИМЕНТАЛЬНОЕ ИССЛЕДОВАНИЕ ПОБОЧНЫХ АТАК ПО ЦЕПЯМ ЭЛЕКТРОПИТАНИЯ 58

2.1 Описание предлагаемой измерительной установки 58

2.1.1 Обоснование выбора и описание устройства сбора данных 58

2.1.2 Описание принципов монтажа анализируемого чипа 61

2.2 Оценка качества измерительной установки 63

2.2.1 Электронные шумы 64

2.2.2 Алгоритмический шум 65

2.3 Анализ характеристик снимаемых форм сигнала 66

2.4 Исследование зависимости энергопотребления микроконтроллера от выполняемых операций и обрабатываемых данных 71

2.4.1 Зависимость энергопотребления от обрабатываемых данных при выполнении операции записи в чистую ячейку ОЗУ 72

2.4.2 Зависимость энергопотребления от обрабатываемых данных при выполнении операции записи в предварительно заполненную ячейку ОЗУ 80

2.4.3 Зависимость энергопотребления от обрабатываемых данных при выполнении операции чтения из ячейки ОЗУ в чистую ячейку рабочего регистра 82

2.4.4 Зависимость энергопотребления от обрабатываемых данных при выполнении операции чтения из ячейки ОЗУ в предварительно заполненную ячейку рабочего регистра 83

2.4.5 Зависимость энергопотребления от обрабатываемых данных при выполнении операции чтения из ячейки ПЗУ в чистую ячейку рабочего регистра 85

2.4.6 Зависимость энергопотребления от обрабатываемых данных при выполнении операции чтения из ячейки ПЗУ в предварительно заполненную ячейку рабочего регистра 86

2.4.7 Зависимость энергопотребления от обрабатываемых данных при выполнении операции копирования данных между рабочими регистрами 88

2.4.8 Зависимость энергопотребления от обрабатываемых данных при выполнении операции копирования данных из одного рабочего регистра в другой, предварительно заполненный регистр 88

2.4.9 Зависимость энергопотребления от обрабатываемых данных при выполнении операции XOR между комбинациями из двух рабочих регистров 90

2.5 Оценка эффективности реализации DPA в отношении различных операций, с использованием экспериментальных данных 92

2.5.1 Описание методологии измерений 92

2.5.2 Реализация DPA для операции записи данных в ячейку ОЗУ 95

2.5.3 Реализация DPA для иных операций 103

2.6 Выводы по главе 111

ГЛАВА 3. ИССЛЕДОВАНИЕ ТЕОРЕТИЧЕСКИХ И ПРАКТИЧЕСКИХ АСПЕКТОВ РЕАЛИЗАЦИИ DPA 113

3.1 Описание используемой модели энергопотребления чипа 113

3.2 Особенности реализации DPA с расчётом ковариационных векторов 115

3.3 Особенности реализации DPA с расчётом корреляционных векторов 128

3.4 Сравнительная оценка эффективности реализации DPA с использованием ковариационных и корреляционных векторов 139

3.5 Анализ отличий в эффективности применения DPA к шифрам «Магма» и «Кузнечик» 148

3.6 Экспериментальное исследование реализации DPA применительно к шифру «Магма» 161

3.6.1 Особенности реализации шифра «Магма» на анализируемом микроконтроллере 161

3.6.2 Оценка эффективности реализации DPA в отношении шифра «Магма», с использованием экспериментальных данных 164

3.7 Выводы по главе 168

ГЛАВА 4. РАЗРАБОТКА УНИВЕРСАЛЬНОГО МЕТОДА ЗАЩИТЫ ОТ DPA 170

4.1 Обоснование необходимости разработки нового метода защиты от DPA 170

4.2 Краткая характеристика программных методов защиты от DPA 171

4.2.1 Описание метода маскировки 171

4.2.2 Описание метода программной балансировки 174

4.3 Оценка возможности адаптации метода программной балансировкой для защиты шифра «Кузнечик» 181

4.4 Обобщённая структура универсального метода защиты от DPA 184

4.5 Описание деталей предлагаемой структуры УМЗ 187

4.6 Характеристики предлагаемой структуры УМЗ 195

4.7 Выводы по главе 201

Заключение 202

Список сокращений и условных обозначений 205

Словарь терминов 206

СПИСОК ЛИТЕРАТУРЫ 208

Приложение А. Краткое описание принципа работы параллельных АЦП..... 214

Приложение Б. Краткое описание принципа работы АЦП конвейерного типа..... 216

Приложение В. Краткое описание принципа работы микроконтроллера Atmel ATmega16 217

Приложение Г. Характеристики отсчётов форм сигнала при выполнении операции записи в чистую ячейку ОЗУ..... 220

Приложение Д. Графики, характеризующие зависимости энергопотребления анализируемого микроконтроллера от выполняемых операций и обрабатываемых данных 221

Приложение Е. Таблицы, характеризующие зависимости энергопотребления анализируемого микроконтроллера от выполняемых операций и обрабатываемых данных 233

Приложение Ж. Реализация табличного преобразования на микроконтроллерах Atmel с использованием подхода «чтение со смещением» 244

Приложение И. Графики, корреляционных векторов для рассматриваемых операций.... 248

Приложение К. Реализация УМЗ на основе балансировки на микроконтроллерах фирм Atmel и Microchip 255

Приложение Л. Раунд шифра «Кузнечик» 264

Приложение М. Предлагаемая структура УМЗ 265

Приложение Н. Акт о внедрении научных результатов 266

ВВЕДЕНИЕ

Актуальность темы исследования. Безопасность современных информационно-коммуникационных систем неразрывно связана с алгоритмами, обеспечивающими конфиденциальность и целостность хранящейся и передаваемой информации, а также функции идентификации и аутентификации. Стойкость этих алгоритмов, базируется на вычислительной сложности решения некоторых задач. Причём до недавнего времени, разработчики оценивали их защищённость в отрыве от практической реализации на конкретном устройстве – интегральной схеме. Однако, в 1999 году впервые появилась публикация П.Кёхера [48] на эту тему, в которой было показано, что, к примеру, такой параметр как энергопотребление интегральной схемы, имеет явную зависимость от обрабатываемых данных. Следовательно, сняв формы сигнала, характеризующие энергопотребление чипа в процессе выполнения алгоритма некоторого защитного преобразования (к примеру шифра, DES), и задействовав относительно несложный математический аппарат простого или дифференциального анализа энергопотребления (соответственно атаки SPA или DPA [48]), оказывается возможным восстановить секретную информацию, обрабатываемую чипом (в частности секретный ключ шифра). Данный класс атак называют *атаками по побочным каналам* (или *побочными атаками*), в отличие от криптографических атак, которые используют уязвимости самих алгоритмов защитных преобразований. Побочные атаки основываются на идее использования «утечки» информации о секретных данных, содержащихся в чипе, от его физических параметров. (В данной работе не рассматриваются криптографические методы взлома алгоритмов шифрования, а также методы разработки шифров для защиты от подобного рода криптоанализа. Поэтому можно обоснованно предполагать, что в настоящей работе не исследуются методы криптографии.)

Интегральные чипы, реализующие защитные преобразования, могут применяться в конечных устройствах, имеющих разное назначение, наиболее распространёнными и чувствительными к побочным атакам по цепям электропитания, можно считать оригинальные расходные материалы и запасные части к широкому кругу электронных устройств, смарт-карты и биометрические документы. Последние в настоящее время приобретают всё большее распространение, в том числе на территории России. Оборудование, способное обеспечить высококачественную полиграфию доступно не только государственным структурам и крупным частным компаниям, но и обычным людям. Чип, вмонтированный в одну из страниц документа (к примеру водительских прав, карточки медицинского страхования или паспорта), выполняет роль дополнительной защиты от подделки, в биометрических документах, помимо подделки полиграфии, требуется подделка данных, хранящихся в чипе (которые как минимум дублируют, нанесённую полиграфическим способом информацию). Эти данные защищаются при помощи

комплекса алгоритмических средств (защитных преобразований), среди которых могут использоваться алгоритмы хеширования, ассиметричного, симметричного шифрования и другие подобные методы. (Причём алгоритмы симметричного шифрования используются значительно интенсивнее остальных, поскольку они предъявляют меньшие требования к, обычно, весьма сильно ограниченным аппаратным ресурсам чипа). Отсюда, успешный взлом выполняющегося на чипе защитного преобразования, позволяет сделать копию чипа (или даже поместить в него другие данные). Это даёт возможность сделать копию биометрического документа или смарт-карты. Для биометрических документов это нивелирует дополнительный уровень защищённости, обеспечиваемый использованием подобного чипа. Для смарт-карты, применяемой, к примеру, в качестве ключа доступа это также нивелирует основное её преимущество в плане безопасности по сравнению с классическим (металлическим) ключом, то есть невозможность сделать её полную копию. Аналогичная ситуация наблюдается с оригинальными запасными частями и расходными материалами, в качестве наиболее простого примера можно привести картридж для принтера. Возможность изготовления копии чипа, вмонтированного в оригинальный картридж, позволит изготовить корректно работающий совместимый расходный материал, что в итоге приведёт к убыткам фирмы-изготовителя устройства.

Первые публикации, описывающие возможность взлома защитных преобразований (в частности аппаратных реализаций шифров), выполняющихся на интегральных чипах, сразу же поставили под угрозу все сферы их применения. Однако, необходимо отметить, что, подобные чипы в то время не были столь широко распространены, как сейчас. С тех пор эта тема активно разрабатывалась, в том числе предлагались разные по эффективности и простоте реализации методы защиты от побочных атак по цепям электропитания. Однако результаты теоретических исследований оказывались недостаточно апробированы на практике, и в основном основывались на некоторых идеализированных моделях энергопотребления чипов. Поэтому оставался весьма актуальным вопрос проработки ряда практических аспектов реализации атаки, для того, чтобы учитывать их в используемых теоретических моделях. Кроме того, несмотря на чрезвычайную актуальность темы, по ней существовало, до настоящего времени, крайне небольшое количество российских публикаций, что говорит о недостаточной разработанности темы в отечественной научной среде (по крайней мере открытой её части). Это, в свою очередь, влечёт за собой и практические недоработки заключающиеся, например, в том, что зарубежные адаптации известных программных методов защиты от побочных атак по цепям электропитания, для аппаратных реализаций российских шифров (в частности «Магма» и «Кузнечик») оказываются мало эффективными, поскольку, изначально они разрабатывались преимущественно для американских шифров, например, AES.

Заключая вышесказанное, можно сделать общий вывод, что рассматриваемая тема имеет значительную актуальность не только в плане повышения общей разработанности темы, но и в плане увеличения надёжности российских идентификационных документов.

Степень разработанности темы. Практические и теоретические аспекты побочных атак по цепям электропитания разрабатывались с 1999-го года множеством зарубежных исследователей. Наиболее значимые работы, в плане проработки теоретического аппарата этих атак, принадлежат следующим авторам: S.Mangard, E.Peeters, M.Akkar, E.Brier, P.Kocher, T.Messerges [10; 11; 37; 42; 48; 51; 55]. Значительный вклад в разработанность вопросов практической реализации атак внесли: J.Balasch, T.Eisenbarth, A.Moradi, D.Oswald [41; 44; 57; 59]. Наибольший вклад в разработку методов защиты от этих атак внесли: M.Akkar, J.Ambrose, A.Arora, J.Lv, F.Mace, M.Masoumi, D.May, A.Neve, G.Ratanpal, K.Tiri: [35; 36; 38; 39; 40; 49; 50; 52; 53; 58; 60; 62].

Несмотря на весьма достаточную разработанность данной темы, даже в зарубежной литературе остался целый ряд нераскрытых вопросов, как в части деталей практической реализации атаки, так и теоретических аспектов. Кроме того, существующие методы защиты от побочных атак по цепям электропитания были или слишком дороги в реализации, или обеспечивали защищённость аппаратных реализаций лишь небольшого числа зарубежных шифров. В частности, оставался открытым вопрос разработки дешёвого и надёжного метода защиты аппаратных реализаций российских шифров ГОСТ.

Цель и задачи исследования. Цель исследования состоит в повышении безопасности информационно-коммуникационных технологий в части реализации защитных преобразований, выполняемых интегральными чипами. Для достижения цели исследования в работе были поставлены и решены следующие частные задачи:

1. Разработка рациональной конструкции аппаратно-программного комплекса, предназначенного для анализа «утечек» информации по цепям электропитания (при ограничении стоимости используемого оборудования).
2. Экспериментальное исследование «утечек» информации по цепям электропитания интегрального чипа в зависимости от выполняемых им операций.
3. Оценка эффективности реализации побочных атак по цепям электропитания с проведением детального комплекса измерений.
4. Доказательство возможности применения побочной атаки по цепям электропитания на аппаратные реализации российских шифров.
5. Проведение сравнительного анализа сложности применения побочных атак по цепям электропитания к аппаратным реализациям шифров «Магма» и «Кузнечик».

6. Разработка универсального метода защиты от побочных атак по цепям электропитания (не зависящего от операций, составляющих защищаемый алгоритм и, в частности, применимого к аппаратным реализациям российских блочных шифров).

Объект исследования – аппаратно-реализованные защитные преобразования.

Предмет исследования – Исследование теоретических и практических аспектов реализации атак по побочным каналам, использующих информацию, утекающую по цепи электропитания, а также разработка методов защиты от атак этого типа.

Научная новизна.

1. В отличие от известных работ, для снятия данных об энергопотреблении чипа, вместо осциллографа, предложено использовать значительно более дешёвое и наилучшим образом подходящее оборудование – аналого-цифровой преобразователь.
2. В модели «утечки» информации учтены новые особенности энергопотребления интегральных чипов, выявленные в результате проведённого комплекса измерений.
3. Впервые представлена модель побочной атаки по цепям электропитания на аппаратные реализации обоих российских блочных шифров ГОСТ и произведён сравнительный анализ сложности её выполнения.
4. В отличие от других программных методов защиты, предлагаемый метод является универсальным (применим к защите любых аппаратно-реализованных защитных блоковых преобразований, в том числе российских шифров ГОСТ).

Теоретическая и практическая значимость работы.

Теоретическая значимость научных положений, изложенных в работе, состоит в следующем:

1. Расширен ряд аппаратных средств, позволяющих реализовать побочную атаку по цепям электропитания.
2. Дополнена модель, связывающая энергопотребление интегральных чипов с обрабатываемыми данными.
3. Расширена область применения побочных атак по цепям электропитания на аппаратные реализации шифров ГОСТ.
4. Расширен класс методов, обеспечивающих защиту преобразований на интегральных чипах от атак по побочным каналам.

Практическая значимость результатов проведенных исследований состоит в следующем:

1. Предлагаемая архитектура позволяет разработать измерительные системы на базе АЦП для решения задач анализа энергопотребления чипов при значительно меньшей стоимости чем существующие решения (базирующиеся на использовании осциллографа).

2. Модель «утечки» информации может быть использована в дальнейшем для оценки эффективности реализации побочных атак по цепям электропитания и методов защиты от них.
3. Представленная модель побочной атаки позволяет осуществить взлом аппаратных реализаций шифров ГОСТ за обозримое время. Модель может быть использована в дальнейшем для разработки методов защиты от подобной атаки.
4. Предложенный метод защиты обладает низкими требованиями к дополнительным аппаратным ресурсам чипа, простотой реализации и невысокой стоимостью.

Часть основных научных результатов реализована в АО «Научные приборы», по итогам чего получен акт о внедрении научных результатов. Внедрение результатов позволило:

- уточнить требования к разрабатываемым на предприятии электронным идентификационным документам;
- повысить надёжность функционирования производимого оборудования со встроенными блоками электронной идентификации.

Методология и методы исследования. Для решения поставленных задач использовались как общенаучные, так и специальные методы исследования, а именно аппарат и методы алгебры и теории вероятности (в том числе корреляционный анализ), методология программирования (на языках низкого и высокого уровней), компьютерное моделирование, эксперимент.

Положения, выносимые на защиту.

1. Архитектура средства моделирования побочных атак по цепям электропитания. Модель «утечки» информации, обрабатываемой интегральным чипом, по цепи электропитания, основанная на результатах проведённого комплекса измерений.
2. Доказательство возможности успешного применения побочных атак по цепям электропитания к аппаратным реализациям шифров ГОСТ.
3. Метод защиты преобразований на интегральных чипах от побочных атак по цепям электропитания.

Достоверность и апробация результатов.

Достоверность результатов подтверждается строгими теоретическими доказательствами, компьютерным моделированием и результатами проведённых экспериментов.

Основные положения работы докладывались на Международной научно-технической и научно-методической конференции «Актуальные проблемы инфотелекоммуникаций в науке и образовании» в СПБГУТ (Санкт-Петербург, 2013-2017), 24-й научно-технической конференции «Методы и технические средства обеспечения безопасности информации» (Санкт-Петербург, 2015).

Публикации. Основные результаты диссертационного исследования опубликованы в 13-ти научных трудах [15, 23-34], из них: 6 – в рецензируемых научных изданиях из Перечня ВАК; 6 – в сборниках научных статей, трудов, тезисов докладов и материалах конференций; 1 – отчет по НИР.

Структура и объем работы. Диссертационная работа состоит из введения, основной части (содержащей 4 раздела), заключения, списка литературы и приложений.

Общий объем работы – **266** страниц, из них основного текста – 213 страниц. Работа содержит 118 рисунков и 31 таблицу. Список литературы включает 64 библиографических источника.

ГЛАВА 1. ОБЗОР ФИЗИЧЕСКИХ И ТЕОРЕТИЧЕСКИХ ОСНОВ ПОБОЧНЫХ АТАК ПО ЦЕПЯМ ЭЛЕКТРОПИТАНИЯ

Вопрос безопасности современных информационных систем неразрывно связан с надёжностью применяющихся в них алгоритмов и протоколов, позволяющих обеспечить конфиденциальность информации, а также функции идентификации и аутентификации. Математическим базисом в данном случае выступают методы защитного преобразования информации, такие как алгоритмы шифрования (симметричные и несимметричные) и хэш-функции (ключевые или бесключевые).

Реализация таких, защитных алгоритмов в программном виде с использованием средств операционной системы (ОС) персонального компьютера (ПК), сервера или мобильного телефона, можно считать весьма рискованной. Современные ОС являются чрезвычайно сложными и не безопасными средами для обработки чувствительной информации. Поэтому в настоящее время выполнение этих операций, как-правило выносят в отдельные интегральные схемы – *невскрываемые чипы*. Такие чипы используются в электронных устройствах самого широкого профиля, например [15]:

- ПК и серверное оборудование (в виде чипа TPM, смарт-карты или USB-ключа);
- системы доступа в помещения (в виде смарт-карты);
- электронные идентификационные документы государственного образца, например: паспорта, водительские удостоверения, медицинские карточки (в виде отдельной смарт-карты или NFC чипа, встраиваемого в одну из страниц соответствующего документа);
- устройства для совершения электронных платежей (в виде смарт-карты или телефона, использующего технологию NFC);
- устройства для доступа к системам платного телевидения [57];
- принтеры (для идентификации оригинальных расходных материалов);
- широкий спектр гражданского и военного оборудования.

В данной работе будет сделан акцент на чипах, в основе работы которых лежат такие защитные преобразования как алгоритмы симметричного шифрования, в частности блочные шифры. Функцией последних является преобразование информации (некоторых входных данных, предварительно разбитых на блоки фиксированной длины в выходные) с использованием некоторого секрета, называемого секретным ключом [4, с. 12]:

$$E = f\left(M,\, k\right), \tag{1.1}$$

где M и E соответственно блоки входных и выходных данных, обычно имеющие двоичный вид; k – секретный ключ; $f(\cdot)$ – функция шифрования.

Блочные симметричные шифры нашли широкое применение в невскрываемых чипах за счёт значительно меньших требований, предъявляемых к вычислительным ресурсам, в отличие от систем с открытым ключом. В частности, такие шифры как 3DES и AES активно используются во всех перечисленных выше видах электронных устройств [15].

Согласно принципу, сформированному в 19-м веке Огюстом Керхгоффом [4, с. 12], такие системы строятся из расчёта, что злоумышленнику окажутся известны все детали их реализации, кроме используемого ими секретного ключа. При этом без знания ключа, расчёт неизвестного M по известному E, или наоборот, оказывается вычислительно сложной задачей (то есть невозможной с использованием любых существующих вычислительных ресурсов за разумное время). На столько же сложной задачей оказывается и расчёт новой пары (E', M'), где $M' \neq M$.

Под невскрываемыми чипами имеются ввиду интегральные схемы, изготовленные как-правило на одном кристалле (то есть интегральная схема в едином корпусе для поверхностного монтажа, выполняющая функции целого устройства – как правило имеющая в своём составе процессор, память, интерфейсы ввода-вывода, счётчики, таймеры и тому подобное). Функцией таких устройств является как защищённое выполнение процедур идентификации и аутентификации, так и надёжное хранение секретного ключа. Они принимают команды и данные от внешнего устройства, выполняют заложенные алгоритмы (в том числе с применением секретного ключа), результат выполнения отсылается назад. Такие чипы являются закрытыми платформами, со своей, весьма простой ОС, при этом, в отличие от ПК, у них отсутствует интерфейс подключения к интернету и на них обычно невозможно установить стороннее программное обеспечение. Секретный ключ хранится в специальной, защищённой области памяти, к которой могут обращаться лишь определённые программы, выполняемые на чипе, при этом ни прямыми, ни косвенными способами чтение ключа извне оказывается невозможно [15].

При анализе защищённости преобразований, производимых невскрываемым чипом очевидно следует рассматривать стойкость всего аппаратно-программного комплекса в целом, а не только заложенных в нём алгоритмов в плане их устойчивости к криптографическим методам взлома.

Далее, для упрощения, невскрываемые чипы будут называться просто *чипами*, под атаками на чипы, будет подразумеваться несанкционированная попытка извлечения хранящегося в нём секретного ключа. А при оценке безопасности чипов, согласно принципу Керхгоффа, будет анализироваться наихудший вариант – когда злоумышленнику известна детальная информация

как о технических характеристиках чипа, так и о реализованных на нём программных алгоритмах, неизвестным для него является лишь секретный ключ.

1.1 Краткая характеристика атак по побочным каналам

Алгоритмы идентификации и аутентификации, использующиеся в современных чипах, считаются защищёнными от известных криптографических атак, поэтому с математической стороны большинство современных чипов являются абсолютно защищёнными. Однако существует класс атак, называемых атаками по *побочным* (или *сторонним*) *каналам* [23; 24; 61], использующие дополнительную информацию (так называемую утечку) при физической реализации алгоритма, такие атаки условно можно разделить на два класса [11, с. 12-13]:

- Пассивные – атаки, не предполагающие вмешательство злоумышленника в выполняемые чипом операции. Утечка информации получается от анализа физических параметров чипа, которые, как-правило зависят от обрабатываемых данных (энергопотребление, электромагнитное излучение, время выполнения операций и тому подобное).
- Активные – атаки, предполагающие воздействие на вычислительный процесс в чипе, с целью перевести устройство в уязвимое состояние. В этом случае к утечке информации приводят появляющиеся ошибки при выполнении операций, составляющих алгоритм.

Каждый из этих двух классов атак, в свою очередь может быть разделён на три подкласса [11, с. 11-17; 15; 61]:

- Агрессивные атаки – подкласс весьма эффективных атак, предполагающих получение прямого доступа к внутренним компонентам чипа, с полным или частичным разрушением его корпуса. Наиболее тривиальный пример агрессивной атаки – создание отверстия в корпусе чипа (механическим или химическим способом) и прямое электрическое подключение к шине данных. В дальнейшем, в зависимости от того применяется пассивная атака или активная, данные могут просто считываться, или каким-либо образом модифицироваться. Этот тип побочных атак весьма сложен, требуются высококвалифицированные специалисты, дорогое лабораторное оборудование и длительное время на реализацию в отношении каждого взламываемого устройства.
- Полуагрессивные атаки – обычно, требуется разрушение корпуса чипа, однако, данный класс атак не предполагает установления электрических соединений с внутренними узлами чипа – оказывается воздействие на различные области памяти при помощи лазера, УФ-излучения или рентгеновского излучения (в последнем случае физическое разрушение

корпуса может не потребоваться), с целью изменения программного кода чипа или провоцирования ошибок в вычислениях. Этот тип побочных атак требует значительно более дешёвого оборудования, чем агрессивные атаки, также их реализация занимает меньшее время.

- Неразрушающие атаки – не требуют повреждения корпуса чипа или оказания каких-либо иных воздействий на чип, которые могут быть обнаружены после реализации атаки. Они могут быть активными: к примеру изменение тактовой частоты задающего генератора, напряжения питания или температуры чипа, от стандартных значений, с целью провоцирования ошибок в выполняемых операциях. В качестве пассивных неразрушающих атак можно выделить анализ звуковых колебаний чипа, времени выполнения операций, электромагнитного излучения, энергопотребления – эти параметры работы чипа, зависят от выполняемых им операций и обрабатываемых данных. Применяя специализированные алгоритмы анализа указанных параметров, в предельном случае становится возможным определение секретного ключа чипа. Этот тип побочных атак является наиболее опасным по двум причинам: владелец скомпрометированного чипа не сможет определить факт реализации атаки; эти атаки хоть и требуют длительного и детального изучения анализируемого чипа, но после этого они весьма просто и дёшево масштабируются в отношении аналогичных чипов, в итоге стоимость атаки в пересчёте на каждое взломанное устройство может оказаться ничтожной.

Наиболее известной, универсальной и опасной побочной атакой можно считать пассивную неразрушающую атаку, использующую *анализ энергопотребления*, который был впервые предложен Полом Кёхером в 1999 году [48]. Идея этого подхода основана на физических особенностях работы современных чипов, приводящих к тому, что их энергопотребление зависит от выполняемой операции и обрабатываемых на ней данных. Данная работа посвящена именно этому типу побочных атак. В следующих разделах, на основе проведённого анализа литературных источников, будут даны общие сведения об этой атаке, включая описание физических особенностей современных интегральных чипов, которые делают этот тип атак возможным, описание оборудования необходимого для их реализации. А также теоретические основы побочных атак по цепям электропитания, включая специализированные алгоритмы, позволяющие восстановить секретный ключ практически любого защитного алгоритма, реализованного на чипе, и предлагаемые методы защиты от этого вида атак.

1.2 Физические основы побочных атак по цепям электропитания

Современные интегральные схемы (ИС) можно разделить на три большие группы:

- ASIC [19] – интегральные схемы специального назначения, они представляют собой чипы с заранее определённой логикой и являются узко специализированными. Используются для решения конкретной задачи в конкретном устройстве массового производства (мобильные телефоны, компьютеры, телевизоры и тому подобное). Вследствие того, что такие чипы изготавливаются для выполнения строго ограниченных функций, характерных только для данного устройства, то ASIC выполняют их очень быстро, а сами схемы (в силу отсутствия ненужных компонентов) имеют минимальный размер. ASIC чаще всего, изготавливаются под заказ и обладают низкой себестоимостью – от нескольких центов, до нескольких долларов, только при массовом производстве (сотнями тысяч и даже миллионными тиражами). При мелкосерийном и единичном производстве ASIC микросхемы обходятся очень дорого (в этом случае их производство оказывается не рентабельным).
- Микроконтроллеры [1; 3; 7] – устройства, сочетающие на одном кристалле функции процессора и периферийных устройств, в том числе ОЗУ, ПЗУ (являются миниатюрными, однокристальными ЭВМ). Микроконтроллеры более универсальны чем ASIC, так как логика их работы не определяется на заводе-изготовителе – они могут быть сконфигурированы программистом для выполнения широкого спектра задач. В этом случае используется массовое производство (миллионными тиражами) и как следствие обеспечивается низкая стоимость, а широкий спектр характеристик выпускаемых микроконтроллеров, позволяет подобрать оптимальный для решения конкретной задачи. В отличие от ASIC, вследствие своей универсальности, имеют больший размер, по причине программного способа реализации алгоритмов они значительно уступают в производительности, а при массовом производстве обходятся дороже. Сфера применения очень широка, охватывает практически всю потребительскую электронику и промышленность.
- ПЛИС [57] – устройства, позволяющие создать на одном кристалле любую цифровую интегральную схему. В отличие от ASIC, логика работы ПЛИС не определяется при изготовлении, а задаётся посредством программирования. В отличие от микроконтроллера, программирование которого предусматривает ввод в его ПЗУ набора инструкций, согласно которым встроенный процессор выполняет требуемые операции (функции реализуются программным способом с использованием уже имеющейся внутренней электрической схемы), при программировании ПЛИС на аппаратном уровне создаётся схема соединения между его внутренними элементами. На одном кристалле ПЛИС может быть

запрограммирована любая интегральная схема вплоть до процессора. Наиболее мощные ПЛИС эквивалентны ASIC схемам, а их тактовая частота может достигать гигагерца – производительность ПЛИС лежит где-то посередине между микроконтроллерами и ASIC, а их стоимость при массовом производстве значительно превосходит стоимость микроконтроллеров, а тем более ASIC.

Примитивным элементом ИС любого типа являются электронные ключи, которые, в подавляющем большинстве случаев, физически выполняются в виде полевых МОП (металл-оксид-полупроводник) транзисторов p-типа и n-типа (рисунок 1.1), технология производства ИС с их использованием называется КМОП (комплементарная структура металл-оксид-полупроводник). Транзисторы n-типа проводят ток от истока к стоку в том случае если к управляющему электроду – затвору приложено положительное (относительно истока) напряжение, p-типа – если отрицательное [11, с. 25; 45].

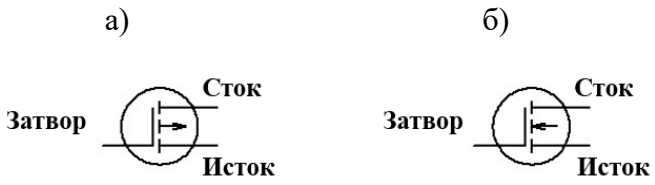


Рисунок 1.1 – Полевые транзисторы: а) p-типа; б) n-типа

Транзисторы, соединённые в определённой последовательности, образуют так называемые логические вентили (ЛВ) – базовые логические элементы ИС, выполняющие элементарную логическую операцию, преобразуя один или несколько входных логических значений, в выходные. В свою очередь, ЛВ образуют узлы большей агрегации, такие как сумматоры, счётчики, триггеры, дешифраторы и тому подобное.

Основным достоинством КМОП технологии является низкое энергопотребление. Причиной этого является использование полевых транзисторов, которые в отличие от биполярных управляются напряжением, а не током, при этом сопротивление управляющего электрода (между затвором и истоком) имеет порядок МОМ или ГОМ. Ниже будет приведён пример работы простейшего ЛВ типа «НЕ» (инвертора), состоящего всего из двух транзисторов. Данный пример является весьма показательным (он часто используется зарубежными авторами для объяснения причин утечки информации по цепи электропитания [10, с. 28-29; 11, с. 24-25; 45; 55]) и может быть весьма просто экстраполирован на другие типы ЛВ, которые отличаются друг от друга количеством составляющих их транзисторов и схемой их взаимного соединения.

Принципиальная схема инвертора приведена на рисунке 1.2, он состоит из двух последовательно соединённых транзисторов p-типа (сверху) и n-типа (снизу). Вход инвертора соединён с затворами обоих транзисторов, а в качестве выхода выбирается точка соединения стока нижнего транзистора с истоком верхнего. Если на вход инвертора подать «0» (рисунок 1.2), то транзистор p-типа (верхний) откроется, а n-типа (нижний) закроется – выход окажется отсоединённым от земли и подсоединённым к питанию, на нём будет логическая «1». При подаче на вход инвертора логической «1» – верхний транзистор закроется, а нижний откроется выход окажется соединённым с землёй – на нём будет логический «0».

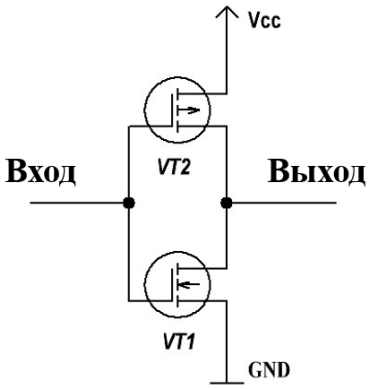


Рисунок 1.2 – Принципиальная схема инвертора

Из описания понятно, что верхний транзистор (p-типа, расположенный между контактом питания и логическим выходом), отвечает за формирование логической «1». А нижний транзистор (n-типа, расположенный между землёй и логическим выходом), отвечает за формирование логического «0». Подобный подход к проектированию ЛВ позволяет добиться того, что при любых логических уровнях на входе, оба транзистора не окажутся одновременно открытыми. Данный пример характеризует одно из главных достоинств КМОП технологии – в *статическом режиме* (интервал времени в течении, которого логические уровни, подаваемые на вход логического вентиля, не изменяются) ток $I_{\text{СТАТ}}$ между цепью питания и земли, образуемый исключительно токами утечки, оказывается ничтожно мал – порядка 1 пА на один вентиль [10, с. 29]. Следовательно, статическая потребляемая мощность может быть рассчитана по следующей формуле [10, с. 29]:

$$P_{\text{СТАТ}} = I_{\text{СТАТ}} \times V_{\text{DD}}, \qquad (1.2)$$

что для напряжения питания в 5В составляет пренебрежимо малые $P_{\text{СТАТ}} = 10^{-12} \cdot 5 = 5$ пВт.

Однако входной сигнал может совершить четыре типа переходов, и если при (условных) переходах 0→0 или 1→1, потребляемая из сети питания мощность $P_{0\to0}$ и $P_{1\to1}$ будет практически отсутствовать, то при переходах 0→1, 1→0 – она будет значительной (работа логического

вентиля, в процессе которой, логический уровень на входе ЛВ меняется – называют [10, с. 29-31; 11, с. 25] *динамическим режимом*). Мощность, потребляемая логическим вентилем в динамическом режиме, помимо статической составляющей, определяется ещё и динамической (таблица 1.1), которая, в свою очередь, обусловлена двумя причинами: током короткого замыкания, и током заряда паразитных ёмкостей.

Таблица 1.1 – Переходы логических уровней инвертора (по материалам [10, с. 30])

Переход	Потребляемая мощность	Тип потребляемой мощности
0→0	$P_{0\to0}$	Статическая
0→1	$P_{0\to1}$	Статическая + Динамическая
1→0	$P_{1\to0}$	Статическая + Динамическая
1→1	$P_{1\to1}$	Статическая

Ток короткого замыкания возникает по той причине, что в моменты изменения входного логического уровня 0→1, 1→0, одни транзисторы вентиля открываются, другие закрываются, при этом переключение происходит не одновременно и моментально – существует очень короткий промежуток времени, когда оба транзистора оказываются открытыми, что обуславливает кратковременный всплеск потребляемой мощности, за счёт протекания тока короткого замыкания, через открытые транзисторы (от VDD к GND). Потребляемую в данном случае мощность можно оценить по следующей формуле [10, с. 31]:

$$P_{\text{КЗ}} = V_{\text{DD}} \cdot I_{\text{СРЕД}} \cdot p_{0\to1} \cdot f \cdot t_{\text{КЗ}}, \qquad (1.3)$$

где $t_{\text{КЗ}}$ – интервал времени, в течение которого оба транзистора оказываются открытыми; f – тактовая частота микросхемы; $I_{\text{СРЕД}}$ – средний ток, протекающий через транзисторы; $p_{0\to1}$ – вероятность возникновения переходов 0→1, 1→0.

По оценке авторов [11, с. 24] током короткого замыкания обусловлено не более 20% от всей потребляемой мощности вентиля. Наибольшую мощность вентиль потребляет во время заряда своей ёмкостной составляющей (представляющей собой совокупность собственных паразитных ёмкостей транзисторов и линий питания, соединяющих их, они условно обозначены на рисунках 1.3 в виде одного конденсатора $C_{\text{ОБЩ}}$). Когда после логической «1», на вход инвертора подаётся логический «0» (рисунок 1.3а), транзистор с n-каналом (VT1) закрывается, с p-каналом (VT2) открывается. В этот момент ёмкостная составляющая ЛВ заряжается, что обуславливает значительный всплеск энергопотребления, который может быть обнаружен путём измерения падения напряжения на резисторе $R_{\text{ИЗМ}}$, вставленном в разрыв линии питания чипа. При подаче

на вход инвертора логической «1», происходит разряд $C_{\text{ОБЩ}}$ через открытый транзистор n-типа в землю (рисунок 1.3б), что также можно обнаружить измерением падения напряжения на резисторе, вставленном уже в разрыв линии земли чипа.

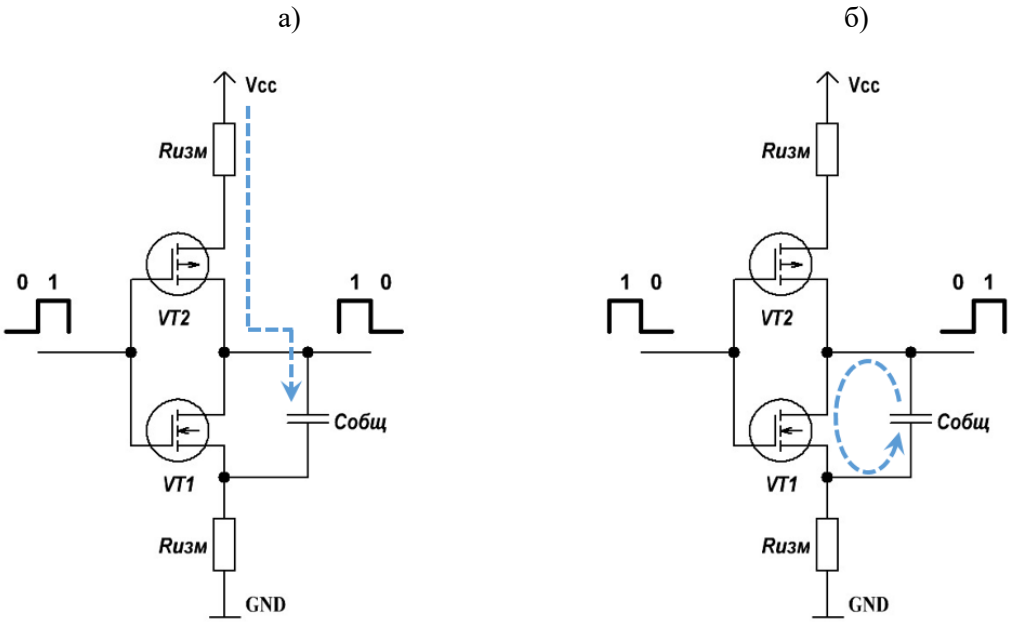


Рисунок 1.3 – Схема протекания тока в инверторе при: а) подаче логического нуля; б) подаче логической единицы (по материалам [11, с. 25])

Потребляемую мощность, обусловленную данным эффектом, можно оценить по следующей формуле [10, с. 31; 11, с. 25]:

$$P = C_{\text{L}} \cdot V_{\text{DD}}^2 \cdot p_{0\to1} \cdot f\,, \qquad (1.4)$$

где C_{L} – величина паразитной ёмкости (ёмкость «конденсатора» $C_{\text{ОБЩ}}$); V_{DD} – напряжение питания; $p_{0\to1}$ – вероятность того, того что логический сигнал на выходе инвертора совершит переход 0→1; f – тактовая частота работы устройства.

Экспериментально было доказано [10, с. 24-25; 11, с. 29; 58], что в КМОП микросхемах наибольшая доля энергопотребления приходятся на описанный выше, заряд паразитной ёмкости (когда на вход инвертора на смену логической единице приходит логический ноль). Когда же на смену логическому нулю, приходит логическая единица (паразитная ёмкость перестаёт заряжаться и начинает разряжаться), также наблюдается всплеск энергопотребления, но значительно меньший, он обусловлен, лишь описанным выше, кратковременным открытием сразу двух транзисторов.

Величина паразитной ёмкости определяется прежде всего типом вентиля и технологическим процессом производства транзисторов, в том числе длиной линий питания,

связывающих ЛВ, а также числом связанных вентилей. (Современный технологический процесс производства интегральных схем не позволяет полностью устранить паразитные ёмкости ЛВ). Типичные значения ёмкости C_L составляют порядка $10^{-15} \div 10^{-12}$Ф=1фФ÷1пФ, таким образом для напряжения питания 5 В, частоты задающего генератора чипа в 4 МГц, вероятности перехода 0→1 равной 0.25, потребляемая логическим вентилем мощность, обусловленная зарядом паразитной ёмкости, составит до $P = 10^{-12} Ф \cdot 25 В \cdot 0.25 \cdot 4 \cdot 10^6 Гц = 25 \cdot 10^{-6} = 10$ мкВт на один вентиль [10, с. 30].

Из групп логических вентилей формируют тот или иной функциональный блок чипа с большей функциональностью и сложностью (сумматоры, счетчики и другие), а те, в свою очередь, соединяясь шинами (увеличивая степень агрегации), образуют интегральные схемы. Следовательно, энергопотребление КМОП микросхемы в целом, определяется совокупным энергопотреблением всех ЛВ чипа, поэтому оно зависит от данных обрабатываемых в любой момент времени, отдельными сборками ЛВ. В следующем разделе будет дана краткая характеристика строения интегральных чипов на примере микроконтроллеров, что необходимо для более детального понимания того, как измеряя энергопотребление всего чипа в целом, можно определить данные, обрабатываемые на том или ином его узле.

1.3 Общее устройство анализируемых чипов

В данной работе атака по побочным каналам будет рассматриваться применительно к микроконтроллерам. Микроконтроллеры очень похожи по своему строению на не вскрываемые чипы, более того, при мелкосерийном производстве, в качестве чипов, осуществляющих процедуры идентификации и аутентификации, весьма часто используются именно микроконтроллеры, по той причине, что заказывать производство ASIC чипов оказывается абсолютно невыгодно. С другой стороны, микроконтроллеры значительно более универсальны, а особенности их строения не представляют секрета – их с лёгкостью можно запрограммировать на выполнение любых доступных операций, и исследовать утечки информации по цепи питания при выполнении этих операций. Также микроконтроллеры являются более доступными для приобретения, поэтому полученные результаты могут быть с лёгкостью воспроизведены, а при необходимости уточнены и доработаны другими исследователями.

В известных публикациях [10; 11; 41] при экспериментальной демонстрации реализаций атак по цепям электропитания, а также тестировании предлагаемых методов защиты, наиболее

часто использовались микроконтроллеры фирм Atmel (семейство ATmega) и Microchip (семейство PIC). Ниже будут кратко рассмотрены основные особенности строения микроконтроллеров этих двух семейств, в частности двух восьмиразрядных моделей: Atmel ATmega16-8PU и Microchip PIC16F877A.

Центральным узлом любого микроконтроллера является *арифметико-логическое устройство* (*АЛУ*), служащее для выполнения всех логических и арифметических операций (обработка данных) [1, с. 24; 3, с. 39; 7, с. 28]. Обобщённая блок-схема АЛУ показана на рисунке 1.4. На его вход поступают:

- два операнда (данные);
- код операции – двоичная последовательность, переключающая АЛУ в режим, соответствующий операции, которую необходимо совершить над операндами (к примеру: сложение по mod2, арифметическое сложение, и тому подобные);
- флаги – информация о результате выполнения предыдущей команды (например, о переносе бита в старшем разряде, при выполнении предыдущей операции арифметического сложения, который необходимо учесть при выполнении текущей операции).

На выходе АЛУ появляется результат выполнения операции, а также, соответствующие ему флаги.

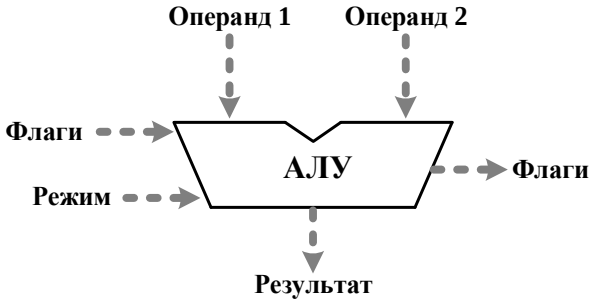


Рисунок 1.4 – Обобщённая блок-схема АЛУ

Упрощённо структуру АЛУ можно представить в виде нескольких сборок ЛВ, каждая из которых реализует ту или иную арифметическую, логическую функцию или функцию передачи данных (например, перемещение данных между регистрами, ОЗУ, ПЗУ). При этом АЛУ современных чипов является многоразрядным, то есть оно способно за один машинный цикл совершать операции с операндами, каждый из которых состоит из нескольких бит. Сейчас широкое распространение имеет восьми разрядная архитектура – такие чипы выпускаются уже несколько десятилетий, они являются дешёвыми и чрезвычайно надёжными, а именно эти факторы и являются наиважнейшими при изготовлении невскрываемых чипов.

Помимо АЛУ в микроконтроллерах присутствует два типа памяти: энергозависимая и энергонезависимая [1, с. 29-32; 3, с. 60; 7, с. 27-32]. В первом случае для хранения данных

требуется постоянное наличие электрического питания, а во втором случае данные сохраняются даже после его отключения. Также, как и АЛУ, каждая ячейка памяти представляет собой транзисторную сборку, к примеру, один из способов построения ячеек энергозависимой памяти заключается в параллельном соединении двух D-триггеров, построенных на базе рассмотренных выше ЛВ типа инвертор [1, с. 44-46].

Энергозависимая память может быть поделена на две области:

- Рабочие регистры
- ОЗУ

В микроконтроллере ATmega16 имеется 32 восьми разрядных рабочих регистра, б**о**льшая часть операций (логические, арифметические) могут быть выполнены лишь с данными загруженными в рабочие регистры, причём эти операции выполняются наиболее быстро (большинство за один машинный цикл). ОЗУ как-правило используется для хранения промежуточных данных при выполнении чипом программы так как в отношении ОЗУ доступны лишь команды чтения/записи, а его объём значительно превышает объём рабочих регистров.

Строение восьми разрядных микроконтроллеров PIC имеет ряд принципиальных отличий от микроконтроллеров ATmega. В частности, PIC имеют всего один рабочий регистр и четыре так называемых банка (*bank*) оперативной памяти, часть ячеек этой памяти используется для управления микроконтроллером, но б**о**льшая часть – для хранения данных. Для выполнения подавляющего большинства операций с одной переменной – она должна быть предварительно загружена в рабочий регистр (например, логические операции переменной с константой, пересылка данных между двумя ячейками ОЗУ и тому подобные). При выполнении операций с двумя переменными (например, логические или арифметические операции между двумя переменными) – одна из них также непременно должна содержаться в рабочем регистре. Учитывая тот факт, что рабочий регистр в микроконтроллерах PIC один – программирование на его ассемблере является весьма нетривиальным процессом, а скорость выполнения программы оказывается в 8-10 раз медленнее чем на микроконтроллерах ATmega.

Отдельно стоит выделить регистр статуса, реализованный в обоих микроконтроллерах [1, с. 24; 3, с. 66]. Он выполнен в виде восьми разрядного регистра, напрямую соединённого с АЛУ. Функциональность этого регистра в микроконтроллерах PIC и ATmega несколько отличается, однако они имеют как-минимум два бита, одинакового функционального назначения:

- бит, называемый флагом переноса, он устанавливается в единицу в том случае, если при выполнении предыдущей операции в старшем разряде произошёл перенос/заём бита;
- бит, называемый флагом нуля, он устанавливается в единицу если результат выполнения предыдущей операции оказался нулевым (байт, состоящий из восьми нулей).

Энергонезависимую память также можно разделить на две области:

- Flash память (память программ)
- EEPROM

Во *Flash памяти* хранится выполняемая чипом программа, также туда могут быть записаны дополнительные «пользовательские» данные (например, таблицы преобразования) и эти данные могут быть использованы при выполнении программы (хотя процедура чтения из Flash памяти занимает гораздо большее время чем из ОЗУ). Однако модифицировать данные, хранящиеся во Flash памяти, возможно лишь при программировании чипа, в отличие от *EEPROM памяти*, основное функциональное отличие которой заключается в том, что хранящиеся в ней данные могут быть изменены в процессе выполнения программы (в ассемблере чипа, кроме команд чтения данных, доступны также команды записи), причём скорость чтения/записи для EEPROM памяти также крайне низка.

Все области памяти связаны с АЛУ *шиной данных* – линией связи, физически выполненной в виде токопроводящих дорожек. При выполнении кода программы из соответствующей области памяти, данные поступают по шине в АЛУ, результат также по шине пересылается в выбранную область памяти. Очевидно, что разрядность шины данных (количество параллельно идущих дорожек) должна быть не меньше разрядности АЛУ – для рассматриваемых микроконтроллеров это восемь бит. Шина данных из-за своей относительно большой протяжённости обладает существенной ёмкостной составляющей. В ряде работ (например, в [10]) утверждается, что энергопотребление, затрачиваемое на передачу данных по шине, занимает весьма большую долю от общего энергопотребления чипа.

Выполняемая чипом программа как-правило создаётся на языке высокого уровня (например, на языке C), а в том случае, когда необходимо обеспечить максимальную скорость работы программы, при минимальных требованиях к аппаратным ресурсам, используется язык низкого уровня – ассемблер (создание сложных программ на котором является весьма трудоёмкой задачей). Затем, она компилируется в результате чего преобразуется в так называемый машинный код, где каждая команда программы представлена в виде двоичного числа – слова. Например, длина слова в рассматриваемых микроконтроллерах PIC равна 14-ти битам. Слова, в свою очередь, записываются во Flash память чипа, а в процессе выполнения программы они последовательно считываются и исполняются в АЛУ. Каждое слово содержит всю необходимую информацию для выполнения конкретной операции, в частности код операции (двоичная последовательность, определяющая выполняемую чипом команду), а также, в зависимости от команды, информацию об операндах, и месте сохранения результата выполнения операции. Таким образом любой алгоритм, выполняемый чипом, представляется в виде набора простых операций, которые им последовательно выполняются. В процессе выполнения этих

операций, задействуются описанные выше узлы чипа, при этом логично предположить, что энергопотребление чипа при выполнении различных операций может значительно отличаться, так как различные операции выполняются в АЛУ отличающимися конструктивно сборками ЛВ, а также при этом задействуются разные области памяти. В конце этой главы будет показано что данная особенность оказывается весьма полезной в таких видах побочных атак как простой и дифференциальный анализ мощности.

В следующем разделе будут приведены основные математические модели, позволяющие связать энергопотребление различных узлов чипа, с данными, которые им на этих узлах обрабатываются.

1.4 Математические модели «утечки» информации по цепи электропитания

Для того, чтобы связать энергопотребление чипа с обрабатываемыми данными, необходимо представить его математическую модель. Эта модель может обладать разными уровнями точности. Чем выше уровень точности, тем больше требуется вычислительных ресурсов и информации о чипе. Следовательно, применение той или иной модели определяется, имеющимися у злоумышленника вычислительными ресурсами и ресурсами, которые он готов затратить на получение информации о технических особенностях анализируемого чипа. Исходя из работы [10, с. 34-43] модели энергопотребления чипа можно разделить как минимум на две группы в зависимости от уровня абстракции: модели, рассматривающие работу чипа на аналоговом уровне и на логическом уровне.

Моделирование работы чипа на *аналоговом уровне* является наиболее подробным, полным и точным. В этом случае предполагается наличие наиболее полной информации о строении чипа вплоть до перечисления составляющих его транзисторов, дорожек, соединяющих их, а также размеры всех паразитных ёмкостей. На основе этих данных строится математическая модель, которая путём решения дифференциальных уравнений (например, с использованием популярного программного пакета SPICE или его аналогов) позволяет рассчитать напряжения и токи в любой части чипа. Для атак на чипы эта модель практически неприменима в основном по причине отсутствия у злоумышленника столь подробной информации об анализируемом чипе.

Моделирование на *логическом уровне* – предполагает наличие информации о логических вентилях чипа, порядке их соединения, часто к ней добавляется информация о задержках сигнала, проходящего от вентиля к вентилю. Модели, описывающие энергопотребление чипа на логическом уровне, требуют значительно меньших вычислительных ресурсов чем аналоговое

моделирование, а также меньших знаний о чипе. Недостатком таких моделей очевидно является их меньшая точность. На первом этапе для каждого ЛВ определяется – какие переходы и с какой задержкой, осуществят связанные с ним вентили, при всех возможных вариантах входных значений (0→0, 0→1, 1→0, 1→1). На втором этапе, для совокупности всех логических вентилей чипа, рассчитываются предполагаемые уровни энергопотребления чипа, при совершении тех или иных логических переходов на отдельных вентилях (другими словами при появлении определённых данных в определённом узле чипа). Данный шаг предполагает наличие моделей, связывающих логические переходы на отдельных вентилях с затрачиваемой ими (на эти переходы) мощностью. Данные модели, в основном основываются на информации о мощности, потребляемой разными типами ЛВ в статическом и динамическом режимах.

Модель, описывающая энергопотребление чипа на логическом уровне, на практике также требует таких знаний о строении анализируемого чипа, которые, как правило, отсутствуют у злоумышленника. Поэтому практическую применимость имеют более универсальные модели [10, с. 38-42; 11, с. 31-33; 21; 42; 63]: так называемая модель *веса Хэмминга* и модель *расстояния Хэмминга*. Как было показано выше, ЛВ, на основе которых изготавливаются чипы, потребляют мощность в основном в моменты переключения. Идея модели расстояния Хемминга состоит в том, чтобы вычислять количество переходов 0→1, 1→0, происходящих в чипе в определённый интервал времени, что в последствии даст возможность построить подобие формы сигнала, с тем лишь отличием, что вместо значений потребляемой мощности она будет содержать количество переключившихся ЛВ.

Положим, что в момент времени i (к примеру в рабочем регистре) присутствует комбинация M_i, а в следующий момент времени $i+1$, в данный регистр из ОЗУ была загружена комбинация M_{i+1}. Тогда, затрачиваемая на данное изменение мощность, будет пропорциональна $HW\left(M_i \oplus M_{i+1}\right)$, где $HW(\cdot)$ – это вес Хэмминга. Например, если восьми битовая двоичная комбинация «1111 1111» изменилась на комбинацию «1111 0000», то расстояние Хэмминга между этими двумя комбинациями будет равно четырём – проще говоря в четырёх из восьми разрядов значения бит изменятся (рисунок 1.5), следовательно, из сети питания, помимо статической мощности (потребляемой всеми ЛВ чипа), будет потребляться динамическая мощность, обусловленная переключением четырёх ЛВ, составляющих четыре ячейки рассматриваемого регистра. Если рассматривать чипы с восьми разрядной архитектурой, то все возможные, комбинации, появляющиеся в регистрах, ОЗУ, EEPROM или на шине данных, могут давать всего девять уровней энергопотребления: например, минимальный нулевой уровень, будет наблюдаться в том случае, если комбинация не изменится ни в одном разряде, первый уровень – при изменении значения одного бита, и так далее вплоть до максимального восьмого

уровня – при изменении значений всех восьми бит. Следовательно, используя эту модель, можно оценить энергопотребление чипа, которое будет пропорционально количеству переходов (0→1 или 1→0).

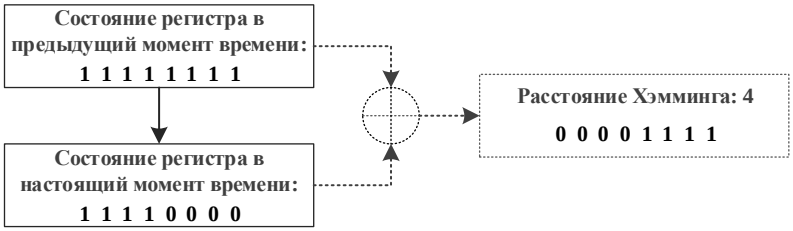


Рисунок 1.5 – Модель энергопотребления по расстоянию Хэмминга (по материалам [21, с. 16])

При использовании модели расстояния Хэмминга обычно, делается ряд допущений:

- в статическом режиме энергопотребление вентилей отсутствует (модель её не учитывает);
- на переход 0 → 1 и на переход 1 → 0, затрачивается одинаковая мощность (в принципе этот недостаток можно нивелировать, раздельно подсчитывая переходы 0 → 1 и 1 → 0);
- отсутствует разница в энергопотреблении между различными типами логических вентилей.

Описанные допущения, сильно упрощают процесс составления модели энергопотребления, требования к вычислительным ресурсам и объём необходимой информации об особенностях строения чипа. Очевидная сложность этой модели заключается в том, что необходимо знать предыдущие значения бит рассматриваемых разрядов, для определения количества разрядов, изменивших своё значение.

В ряде случаев предыдущая модель может быть упрощена тем допущением что в некоторых частях схемы, перед выработкой новой комбинации, биты рассматриваемых разрядов могут обнуляться. В этом случае, очевидно, отпадёт необходимость расчёта значений рассматриваемых бит в предшествующий момент времени – так как принимается, что предыдущие значения бит были равны нулям. Поэтому потребляемая мощность будет пропорциональна количеству единиц (весу Хэмминга) в появляющейся n-битовой комбинации. К примеру (рисунок 1.6) если на восьми разрядной шине данных появляется комбинация «00000010» – это означает, что комбинация, в предшествующий момент времени равная восьми нулям изменилась в одном разряде, таким образом будет потребляться мощность, требующаяся для изменения состояния одного разряда. Если на шине данных появляется комбинация из восьми единиц – это означает, что изменятся все восемь разрядов, следовательно, будет потребляться мощность, требующаяся для изменения состояния восьми разрядов.

Рисунок 1.6 – Модель энергопотребления по весу Хэмминга (по материалам [21, с. 15])

Данная модель крайне проста, практически не требует знания конструктивных особенностей анализируемого чипа и во многих случаях может оказаться весьма эффективной, в частности она использовалась в оригинальной работе Пола Кёхера [48], в работах французских [35; 36; 37] и китайских [49] авторов по защите шифра DES от DPA, а также в работах автора по взлому и защите шифра ГОСТ от DPA [23; 24; 25; 28; 31; 32]. В тех случаях, когда данная модель оказывается неэффективной рекомендуется применять модель расстояния Хэмминга.

В настоящем разделе были описаны основные модели, позволяющие сопоставить энергопотребление чипа с обрабатываемыми им данными. Далее будет описана типичная измерительная установка, позволяющая снять данные об энергопотреблении чипа, причём с такими свойствами, которые позволяют использовать их в дальнейшем при реализации побочных атак по цепям электропитания.

1.5 Снятие данных об энергопотреблении чипа

В разделе 1.4 были описаны две главные, на сегодняшний день, модели утечки, применяемые для взлома чипов. Для проведения практической атаки по цепям электропитания, необходимо произвести измерения энергопотребления чипа во время выполнения защитного преобразования. В данном разделе будет описан каждый компонент требуемой измерительной установки.

1.5.1 Общее устройство измерительной установки

Классическая блок-схема измерительной установки представлена на рисунке 1.7, она состоит из четырёх основных компонентов [10, с. 45]:

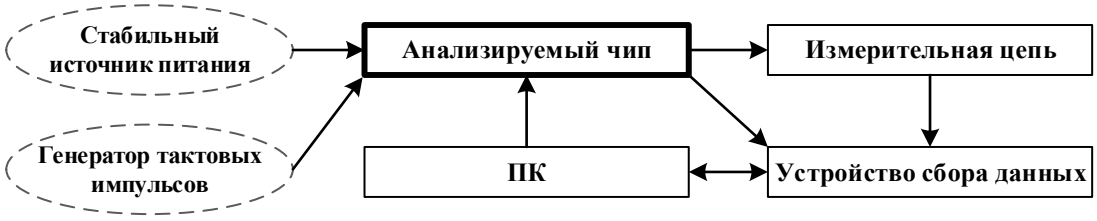


Рисунок 1.7 – Блок-схема измерительной установки (по материалам [10, с. 45])

Анализируемый чип – физическое устройство, из которого требуется извлечь секретный ключ. Анализируемый чип обладает интерфейсом связи с внешними устройствами (обычно через универсальный асинхронный приёмопередатчик), он используется для получения команд (в том числе инициирующих выполнение защитных преобразований), а также отправки ответов с результатом выполнения этих команд. Важной особенностью является то, что чип обычно тактируется внешним кварцевым резонатором или *генератором*, по той причине, что стабильность встроенной LC-цепочки, как-правило, слишком плоха для выполнения ответственных задач, что в частности может приводить к ошибкам в процессе обмена информацией между чипом и внешними устройствами, а также может увеличить джиттер на снимаемых формах сигнала (последнее будет объяснено во второй главе). Также необходимо учитывать то, что в процессе реализации атаки необходимо различать малейшие колебания энергопотребления, таким образом нестабильность или излишняя зашумлённость цепи питания могут привести к невозможности проведения атаки или чрезмерному увеличению требующегося количества данных об энергопотреблении. Поэтому чип необходимо запитывать от очень *стабильного источника питания* лабораторного уровня, в дополнение к этому, рекомендуется [10, с. 50; 11, с. 44-45] непосредственно перед его контактами питания, устанавливать несколько конденсаторов для дополнительной стабилизации напряжения.

Измерительная цепь – комплекс элементов функция которых заключается в том, чтобы обеспечивать электрическое соединение между анализируемым чипом и устройством сбора данных. При этом она должна оказывать минимальное влияние на чип, и передавать сигнал с максимально возможной достоверностью. Обычно в качестве измерительной цепи используется резистор (рисунок 1.8), вставленный в разрыв контакта питания или земли чипа, к которому подсоединяется осциллографический пробник – он снимает падение напряжения на резисторе, которое, по закону Ома, прямо пропорционально току, потребляемому чипом в процессе своей работы [10, с. 50; 11, с. 44-45]. Если разрешающей способности использующегося устройства сбора данных или усилителя пробника недостаточно для распознавания колебаний напряжения – между резистором и пробником, то может быть дополнительно установлен широкополосный усилитель.

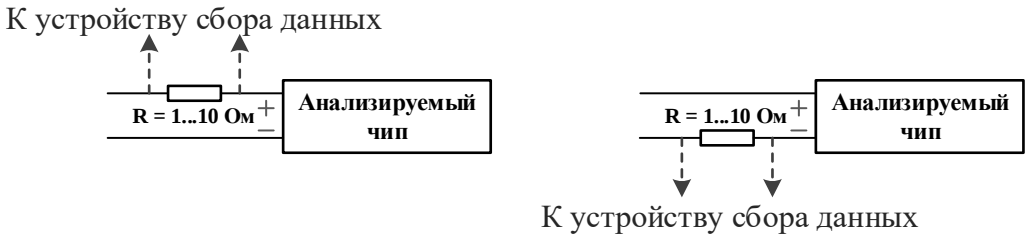


Рисунок 1.8 – Измерительная цепь установки

Вместо установки резистора может быть использован бесконтактный способ измерения при помощи электромагнитного зонда, располагаемого над чипом, улавливающем электромагнитное излучение во время выполнения защитных преобразований [11; 15; 59].

Устройство сбора данных – прибор, преобразующий аналоговый сигнал, полученный от измерительной цепи в цифровой вид, для его последующего сохранения на носителе информации, роль которого обычно играет жёсткий диск ПК. Как правило в качестве устройства сбора данных используется широкополосный осциллограф [10, с. 45-49; 11, с. 42-44; 41; 44; 57; 59], причём стоимость такого прибора может превышать $10 000.

Персональный компьютер – направляет на анализируемый чип команды, инициирующие выполнение защитных операций, кроме того ПК принимает и сохраняет данные об энергопотреблении, получаемые от устройства сбора данных.

Описанная измерительная установка функционирует по следующему алгоритму:

1. ПК переводит устройство сбора данных в режим ожидания триггерного сигнала.
2. ПК отправляет на чип данные, при необходимости сопровождая его служебными командами, инициирующими операции защитного преобразования, а также отправляет триггерный сигнал на устройство сбора данных.
3. Анализируемый чип, получив от ПК необходимые данные, выполняет заложенный алгоритм, после чего возвращается в режим ожидания следующей команды от ПК.
4. Устройство сбора данных, получив, триггерный сигнал, оцифровывает и записывает во внутренний буфер данные от измерительной цепи, эти данные характеризуют энергопотребление чипа, во время проведения защитных преобразований.
5. Устройство сбора данных пересылает записанные данные из буфера к ПК.
6. ПК записывает принятые данные в файл.
7. Шаги 1-6 повторяются столько раз, сколько форм сигнала в итоге необходимо получить.

1.5.2 Требования к характеристикам измерительной установки

Характеристики, требуемой для реализации побочных атак по цепям электропитания, измерительной установки определяются параметрами измеряемого сигнала, наиболее важные из них это [10, с. 48-49; 11, с. 42-44;]: полоса частот, чувствительность, ёмкость памяти.

Количество требуемой буферной памяти устройства сбора данных определяется длительностью выполнения защитного преобразования. Например, если защитное преобразование выполняется чипом за 10 мс, то при использовании осциллографа с частотой дискретизации 100 Мвыб/с, ему потребуется объём буферной памяти, равный 1 Мвыб (это около 1 Мбайт, если предположить, что каждая выборка состоит из восьми бит).

Определение требуемой полосы пропускания – весьма нетривиальная задача, авторы известных публикаций [10, с. 51; 11, с. 44; 41; 44; 57; 59] использовали осциллографы с полосой пропускания от 250 МГц до 1 ГГц, не давая чётких рекомендаций относительно данного параметра. На первый взгляд, одним из важнейших параметров, определяющих требуемую полосу пропускания измерительной установки, является тактовая частота работы чипа, данная информация как-правило является открытой. Известно, что, частота тактового сигнала чипа определяет время между активными переходами сигнала из одного логического состояния в другое. По приходу переднего фронта тактового импульса, в зависимости от выполняемой операции и обрабатываемых данных, задействованные логические вентили переключаются (не одновременно, а с некоторой задержкой друг относительно друга), в этот момент возникает скачок энергопотребления, после чего логические вентили ожидают следующего тактового сигнала, и данная операция повторяется пока выполняется программа. Здесь необходимо отметить, что к примеру, при перемещении байта данных из одного регистра в другой (или в ОЗУ), переключаются не только логические вентили, образующие регистр назначения, но и множество других «системных» логических вентилей, например, вентили, образующие регистр счётчика команд, регистр статуса и другие. Согласно теореме Котельникова, частота дискретизации аналогового сигнала должна быть в 2 раза выше наиболее высокочастотной компоненты этого сигнала, которая в свою очередь определяется временным интервалом $t_{КЗ}$ в течение которого в вентилях протекает ток короткого замыкания, а также временным интервалом t_{C_L} в течение которого происходит заряд паразитных ёмкостей вентилей. Очевидно, что время $t_{КЗ}$ и t_{C_L} для различного типа вентилей будет различно. То есть от числа транзисторов, из которых состоит вентиль, зависит интервал времени $t_{КЗ}$, в течении которого все транзисторы оказываются открытыми, и величина паразитной ёмкости, определяющая интервал времени t_{C_L}. Кроме этого, для каждой конкретной модели чипа, в зависимости от технологического процесса,

по которому она была произведена, значения $t_{КЗ}$ и t_{C_L} очевидно также будут отличаться. Однако, на практике нет смысла обеспечивать такую ширину полосы пропускания измерительной установки, которая позволила бы определять скачки энергопотребления, вызванные переключением каждого отдельного логического вентиля. Дело в том, что, во-первых, за счёт того, что логические вентили срабатывают не одновременно – всплеск энергопотребления сглаживается (то есть увеличивается его длительность и уменьшается амплитуда) данный эффект значительно снижает требуемую полосу пропускания измерительной установки. В работе [11, с. 43] даётся приблизительная оценка величины такого сглаживания, автор утверждает, что за счёт задержки в срабатывании вентилей, длительность всплеска энергопотребления увеличивается приблизительно в 10 раз. Во-вторых, между переключающимися логическими вентилями и контактами питания чипа обычно существует большое количество паразитных составляющих. Например, определённой паразитной ёмкостью и индуктивностью обладают дорожки питания, подходящие к чипу и разводящиеся в нём. Таким образом сигнал, характеризующий энергопотребление чипа, фильтруется ещё до подхода к измерительной цепи, полоса пропускания такого фильтра определяется величиной паразитных составляющих, а, следовательно, конкретной моделью чипа и платой, на которую данный чип смонтирован. Многие авторы отмечают [10, с. 48; 11, с. 42-44], что полоса пропускания этого фильтра значительно ниже частоты исходных сигналов, характеризующих энергопотребление отдельных логических вентилей, и лишь в несколько раз может превышать тактовую частоту чипа. Следовательно, даже при помощи сверхвысокочастотной измерительной установки, физически невозможно снять форму сигнала, которая характеризовала бы мощность, потребляемую отдельными логическими вентилями (переключающимися в течении нескольких наносекунд). Таким образом возможно измерить лишь суммарную мощность, потребляемую логическими вентилями за один такт работы чипа. Поэтому, теоретически, для взлома чипа, работающего на тактовой частоте 4 МГц, должно быть вполне достаточно полосы пропускания в 20-40 МГц, соответствующей осциллографу начального уровня.

Ранее, была дана формула, позволяющая оценить величину скачков энергопотребления, обусловленных зарядкой паразитной ёмкости логического вентиля. Если взять максимально возможную величину паразитной ёмкости C_L в 1 пФ, напряжение питания 5 В, тактовую частоту 4 МГц, вероятность перехода 0→1 равную 0.25, то потребляемая логическим вентилем мощность, обусловленная зарядкой паразитной ёмкости, составит $P = 10^{-12}\,\text{Ф} \cdot 25\text{В} \cdot 0.25 \cdot 4 \cdot 10^{6}\,\text{Гц} = 25 \cdot 10^{-6} = 10$ мкВт, следовательно сила тока составит 2 мкА. Что при установке в цепь питания, резистора на 10 Ом даст всплеск напряжения в 20 мкВ. Разрешающая способность подавляющего большинства осциллографов (не зависимо от полосы

пропускания) ограничивается восемью битами (обычно 2 мВ/дел), в комплекте с такими осциллографами идут пассивные пробники (состоящие из кабелей, разъёмов, резисторов и конденсаторов, в них отсутствуют активные компоненты, такие как транзисторы и интегральные схемы). Эти пробники очень дёшевы, а широкая полоса пропускания таких пробников (сравнимая с полосой пропускания осциллографа) достигается за счёт снижения чувствительности (обычно десятикратного, то есть до 20 мВ/дел). Пассивные пробники без коэффициента ослабления имеют очень низкую полосу пропускания как-правило порядка 7-15 МГц. К сожалению пассивные пробники за счёт плохого соотношения полоса пропускания/чувствительность не могут использоваться в побочных атаках по цепям электропитания. Вместо пассивных пробников обычно используются активные пробники (не ослабляющие сигнал и обладающие полосой пропускания в сотни МГц) совместно с широкополосным усилителем или активные дифференциальные пробники, в которые усилитель уже встроен. Данный подход позволяет повысить чувствительность осциллографа до значений 20÷200 мкВ/дел, что при обеспечении широкой полосы пропускания в сотни МГц, является более чем достаточным для осуществления успешной атаки по цепям питания [16]. Значительным недостатком использования подобной связки является высокая стоимость пробника, которая может достигать половины стоимости осциллографа, таким образом общая стоимость измерительной системы состоящей из связки осциллографа и совместимого с ним активного дифференциального пробника с усилителем достигает $15 000-20 000.

В данном разделе был описан способ получения данных об энергопотреблении чипа требуемого качества. Далее будут рассмотрены два типа побочных атак, позволяющие по полученным данным определить информацию, обрабатываемую чипом при выполнении защитного преобразования (в частном случае – его секретный ключ).

1.6 Теоретические основы построения побочных атак по цепям электропитания (простой анализ мощности)

Из описанного ранее понятно, что энергопотребление чипа будет зависеть от выполняемых им операций и данных, которые он на этих операциях обрабатывает. Наиболее тривиальной атакой, позволяющей использовать эту особенность работы современных чипов является простой анализ мощности (simple power analysis – SPA), описанный в 1999 году П. Кёхером [48]. Атака SPA предполагает прямую интерпретацию полученных данных об энергопотреблении чипа с целью определения как выполняемых им операций, так и обрабатываемых данных

[47; 51; 55]. В разделе 1.3 говорилось, что любой выполняемый чипом алгоритм (в том числе и алгоритм защитного преобразования) представляется в виде набора элементарных операций с определёнными данными, которые тот последовательно обрабатывает. При выполнении разных операций задействуются разные узлы чипа, следовательно, каждый вид операции, выполняемой чипом оставляет свой уникальный «отпечаток» на снятой форме сигнала. В качестве примера можно привести графики, полученные авторами [10, с. 103], в частности на рисунках 1.9 изображена форма сигнала, характеризующая энергопотребление микроконтроллера при выполнении шифра AES.

а) б)

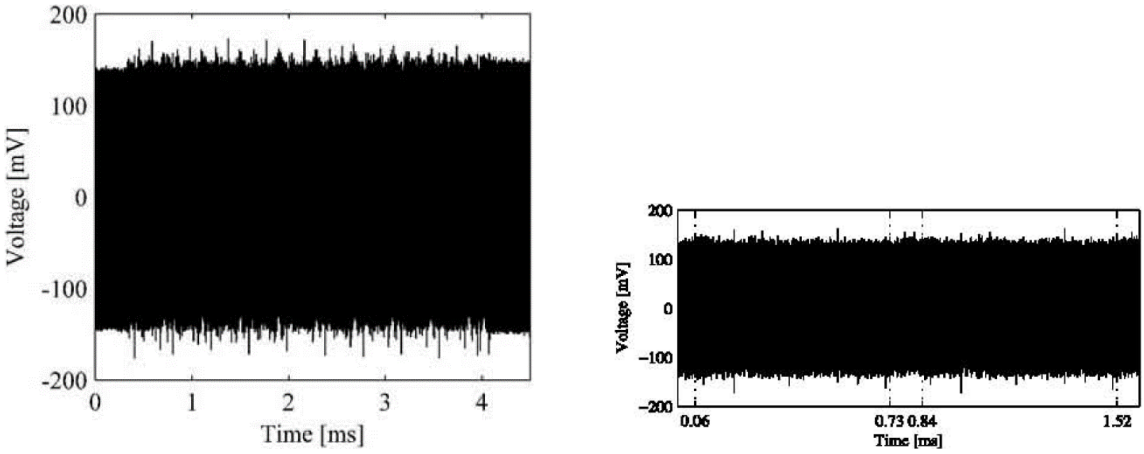


Рисунок 1.9 – Форма сигнала, характеризующая энергопотребление микроконтроллера при выполнении шифра AES: а) полная форма сигнала; б) участок, соответствующий обработке первых двух раундов (по материалам [10, с. 103])

По форме сигнала, изображенной на рисунком 1.9а, можно различить девять участков, характеризующих первые девять раундов шифра AES (обработке первого раунда соответствует участок между 0.06 и 0.73 мс, второго раунда – между 0.84 и 1.52 мс. и так далее), а также ещё один участок меньшей длительности (в районе четвёртой мс.), характеризующий обработку десятого раунда AES, который, согласно стандарту, является сокращённым (в нём отсутствует преобразование MixColumns). Увеличение масштаба области, соответствующей первому раунду (рисунок 1.10а) позволяет определить отдельные части выполняемого чипом алгоритма.

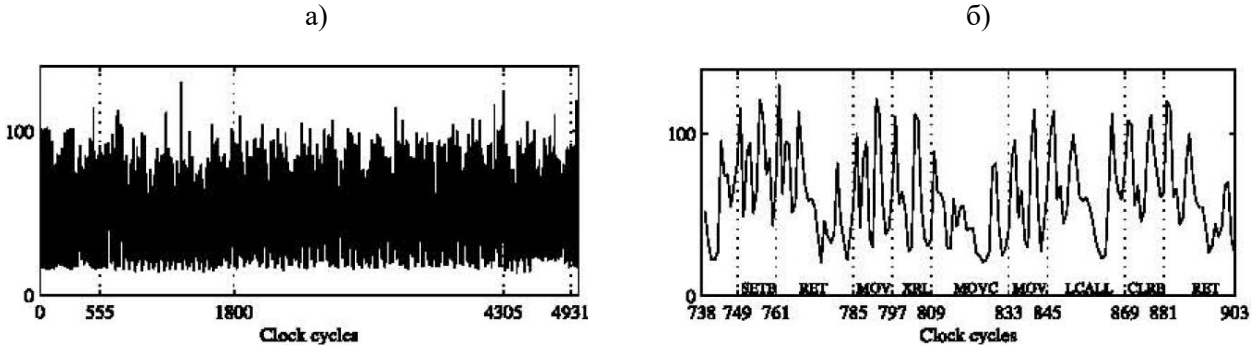


Рисунок 1.10 – Часть формы сигнала, характеризующей энергопотребление чипа при выполнении шифра DES: а) первый раунд (от 555-го по 4'305-й отсчёты); б) первый раунд, участок от 738-го по 903-й отсчёты (по материалам [10, с. 103])

Детальное знание алгоритма AES, а также сопоставление программного кода микроконтроллера, с полученной формой сигнала, позволяет разбить её на три области, в соответствии операциями, составляющими шифр:

- участок между 555 и 1'800 отсчётами соответствует выполнению преобразований AddRoundKey, SubBytes и ShiftRows;
- между 1'800 и 4'305: выполнению преобразования MixColumns;
- между 4'305 и 4'931: выработке ключа первого раунда.

Дальнейшее увеличение масштаба полученной формы сигнала позволяет сопоставить скачки энергопотребления с конкретными выполняемыми операциями (рисунок 1.10б).

После определения скачков напряжения, соответствующих выполнению конкретной операции защитного алгоритма, производится переход к следующему шагу – определению значения обрабатываемой комбинации. В качестве примера, положим, что получены формы сигнала, характеризующие энергопотребление чипа при выполнении алгоритма шифра «Кузнечик» [13]. На первом раунде этого алгоритма i-й блок сообщения $m(i)$ складывается по mod2 с блоком раундового ключа первого раунда $k(1)$ (оба блока в двоичном виде имеют длину 128 бит). При выполнении данного преобразования на восьми разрядном чипе, сложение этих комбинаций осуществляется частями (*подблоками*) по восемь бит за 16 подходов:

$$t_l(i,\ r) = m_l(i) \oplus k_l(r), \qquad (1.5)$$

где $l = 1, \ldots, 16$ – обрабатываемый подблок; $r = 1, \ldots, 10$ – номер раунда; $t_l(i,\ r)$ – результат выполнения операции сложения по mod2, подающийся затем на вход S-box.

Также положим, что после осуществления операции сложения по mod2 её результат пересылается на временное хранение из рабочего регистра в ячейку ОЗУ. На рисунках 1.11

изображены всплески энергопотребления характерные для выполнения этой операции на рассматриваемом микроконтроллере Atmel (более подробно процесс получения и анализ данных об энергопотреблении, будет рассмотрен во второй главе). При этом даны девять форм сигнала, по одной для каждого из весов Хэмминга пересылаемой комбинации (с целью уменьшения шумовой компоненты, для каждого веса Хэмминга было произведено статистическое усреднение по 1 000 измерений).

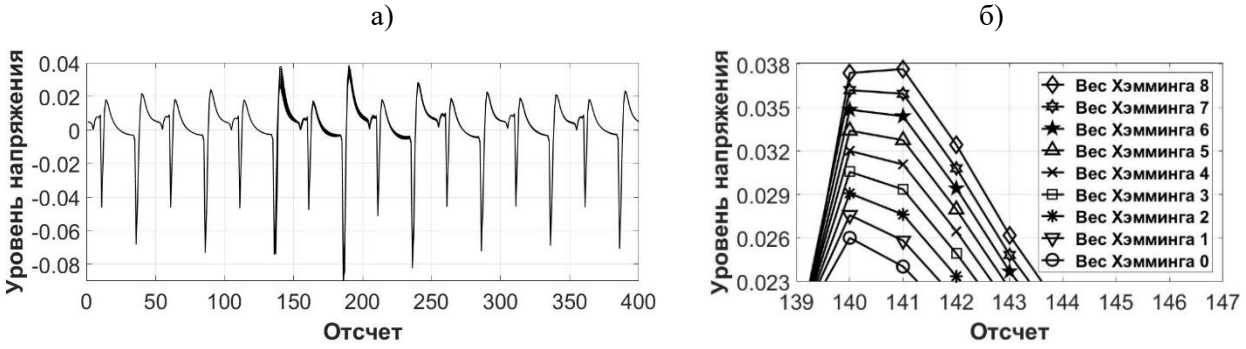


Рисунок 1.11 – Девять усреднённых (для минимизации шума) форм сигнала, полученных при пересылке байт с различными весами Хэмминга, из регистра в чистую ячейку ОЗУ: а) полная форма сигнала; б) увеличенный участок с максимальным всплеском

Из рисунка 1.11б видно, что амплитуда всплеска энергопотребления в диапазоне от 140-го по 143-й отсчёты изменяется пропорционально весу Хэмминга пересылаемой комбинации. Отсюда становится очевидной практическая возможность определения веса Хэмминга комбинации являющейся результатом выполнения промежуточной операции выполняемого защитного алгоритма (в том числе комбинации, зависящей от секретного ключа). Наиболее тривиальный подход к определению веса Хэмминга обрабатываемой комбинации носит название шаблонная атака (Template Attack), она была подобно описана зарубежными авторами для аппаратных реализаций шифров AES [51] и DES [55]. Шаблонная атака предполагает наличие у злоумышленника тестового чипа – чипа аналогичного тому на который предполагается реализация атаки. При этом крайне желательна возможность его свободного программирования, а также многократного инициирования операции защитного преобразования с различными входными данными. Далее такой чип будет называться тестовым, а чип, на который предполагается произвести практическую атаку – анализируемым чипом или просто чипом.

На предварительном этапе реализации шаблонной атаки определяются скачки энергопотребления характерные для выполнения отдельных операций – эта информация позволит локализовать области формы сигнала анализируемого чипа, характеризующие энергопотребление во время обработки зависящих от ключа данных. Затем, в этих областях форм сигнала выбираются отсчёты, на которых их значения наибольшим образом отличаются для

разных весов Хэмминга обрабатываемых данных (например, для форм сигнала, изображённых на рисунке 1.11б это 140-141-й отсчёты – на них разница между всплесками для разных весов Хэмминга достигает 2.6 мВ). По причине наличия шумовой составляющей на формах сигнала, значения амплитуд всплесков энергопотребления при выполнении той или иной операции можно считать случайной величиной с нормальным распределением. Поэтому по каждому весу Хэмминга обрабатываемых комбинаций, рассчитываются средние значения выбранных отсчётов и среднеквадратические отклонения (создаётся шаблон). Полученный шаблон можно представить в виде матрицы или диаграммы (рисунок 1.12), где пунктирная линия соответствует среднему значению 140-го отсчёта, для соответствующего веса Хэмминга пересылаемой комбинации ($\mu(140,\ hw)$, где hw – вес Хэмминга, $\mu(\cdot)$ – математическое ожидание), а серая область выше и ниже линии – указывает на область возможных значений отсчёта в соответствии с правилом трёх сигм ($3\cdot\sigma(140,\ hw)$, где $\sigma(\cdot)$ – среднеквадратическое отклонение). Очевидно, что если эти области не перекрываются, то по всплеску энергопотребления на отдельно взятой форме сигнала с вероятностью свыше 99% можно будет однозначно определить вес Хэмминга обрабатываемой комбинации. Чаще, по причине наличия большой шумовой составляющей на формах сигнала, области значений отсчётов перекрываются, тогда однозначное определение веса Хэмминга по шаблону окажется невозможным – в этом случае можно сделать вывод лишь о вероятности обработки комбинации с определённым весом Хэмминга.

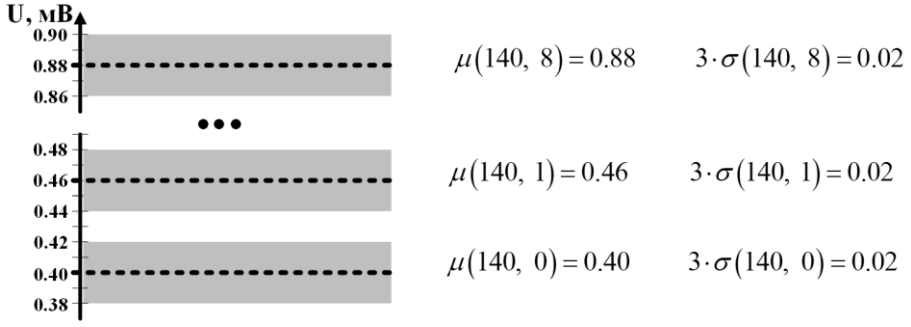


Рисунок 1.12 – Диаграмма математических ожиданий и СКО значений отсчёта для разных весов Хэмминга

Определение веса Хэмминга некоторой промежуточной операции защитного преобразования не приводит к однозначному определению секретного ключа, однако позволяет существенно сократить количество возможных ключей. Положим, что в (1.5) известен подблок сообщения $m_l(i)$ и определён вес Хэмминга $t_l(i,\ r)$ – результата его сложения по mod2 с ключом $k_l(r)$. Вес Хэмминга $HW(t_l(i,\ r))$ может принимать девять значений от 0 до 8, в зависимости от того в скольких разрядах отличаются комбинации $m_l(i)$ и $k_l(r)$. Нулевой вес Хэмминга

говорит об идентичности этих комбинаций, вес Хэмминга равный восьми говорит о том, что комбинации отличаются во всех разрядах, то есть комбинация $k_l(r)$ является инвертированной по отношению к $m_l(i)$. В этих случаях подблок ключа может быть определён однозначно. Однако, если к примеру значение $HW(t_l(i, r))$ равняется одному, то количество возможных вариантов подблока ключа увеличивается до восьми – так как неизвестно в каком именно разряде отличаются комбинации $k_l(r)$ и $m_l(i)$, строго аналогичная ситуация наблюдается при $HW(t_l(i, r))=7$. Количество возможных вариантов ключа $k_l(r)$ в зависимости от веса Хэмминга результата сложения $t_l(i, r)$ приведено в таблице 1.2. Из неё видно, что даже в наихудшем случае, при весе Хэмминга равным четырём, количество возможных вариантов ключа будет сокращено более чем в 3.6 раза.

Таблица 1.2 – Количество возможных вариантов подблоков ключа, в зависимости от значения $HW(t_l(i, r))$

Вес Хэмминга	0	1	2	3	4	5	6	7	8
Количество вариантов ключа	1	8	28	56	70	56	28	8	1

Таким образом для каждого из 16-ти подблоков раундового ключа первого раунда можно получить массив возможных значений, который имеет значительно меньший объём по сравнению со всеми 256-ю возможными значениями подблока ключа. Количество значений в массиве очевидно можно ещё сократить, если анализируется не одна форма сигнала, а несколько, полученные при преобразовании разных сообщений. Также количество значений можно сократить, анализируя веса Хэмминга последующих преобразований алгоритма, в случае шифра «Кузнечик» – это выходы S-box. Положим, что $m_l(i)=00001111$, $k_l(r)=00001110$, получим, что $t_l(i, r)=00000001$, отсюда $HW(t_l(i, r))=1$, злоумышленнику известны значения $m_l(i)$ и $HW(t_l(i, r))$, отсюда он делает вывод, что ключ $k_l(r)$ отличен от $m_l(i)$ всего в одном разряде – существует лишь восемь таких ключей. Однако после преобразования комбинации $t_l(i, r)=00000001$ на S-box получится комбинация $p_l(i, r)=11101110$ – её вес Хэмминга равен шести. Из восьми отобранных ранее вариантов подблока ключа всего шесть дают такой же вес Хэмминга выхода S-box на сообщении $m_l(i)=00001111$. Также необходимо отметить, что при использовании дополнительного анализа выходов S-box уже не два, а десять из 256-ти

возможных вариантов подблока ключа могут быть определены однозначно. При этом по остальным ключам количество возможных вариантов значительно сократится, например (таблица 1.3) существует 12 вариантов подблока ключа $k_l(r)$ при использовании в чипе которых злоумышленник (в случае реализации двухэтапной атаки), получит всего два кандидата на истинный вариант ключа.

Таблица 1.3 – Количество вариантов подблоков истинного ключа $k_l(r)$ (в первой строке), соответствующее количеству кандидатов на истинный вариант ключа, получаемому злоумышленником при двухэтапной атаке (во второй строке)

Количество истинных вариантов подблока ключа	12	6	8	25	36	16	18	24	13	15	32	20	21
Получаемое, злоумышленником количество вариантов подблока ключа	2	3	4	5	6	8	9	12	13	15	16	20	21

Количество возможных вариантов ключа может быть уменьшено ещё сильнее при анализе энергопотребления на последующих операциях, в том числе следующих раундах.

Суммируя приведённое выше можно сформировать ряд базовых условий, требующихся для реализации атаки SPA:

1. возможность подключения к цепям электропитания анализируемого чипа;
2. возможность совершения минимум одного защитного преобразования на анализируемом чипе с использованием неизвестного секретного ключа и известного входного сообщения;
3. наличие аппаратуры, позволяющей измерять энергопотребление чипа, с высокой точностью;
4. наличие у злоумышленника того же (или подобного ему) чипа, как чип, на который предполагается реализация атаки;
5. знание особенностей реализации алгоритма защитного преобразования, используемого чипом.

Первые три условия как-правило являются вполне выполнимыми на практике. Выполнение четвёртого условия конечно может быть затруднительным, но его также можно считать обязательным. В идеальном случае необходимо наличие возможности свободного программирования тестового чипа или, хотя бы, возможность выполнения защитного преобразования для задаваемых злоумышленником входных сообщений на известном ключе – это позволит сформировать шаблон, который будет использоваться для взлома реального чипа.

В случае отсутствия возможности свободного программирования чипа могут возникнуть сложности с идентификацией отдельных операций, выполняемых чипом. Для определения на форме сигнала всплесков энергопотребления, обусловленных выполнением определённой операции с известными данными можно производить расчёт корреляционного вектора. Последний представляет собой набор коэффициентов корреляции между обрабатываемыми чипом данными, на определённом этапе выполнения алгоритма, и отсчётами форм сигнал. В областях, соответствующих обработке рассматриваемых данных, на корреляционном векторе будут наблюдаться значительные всплески. На практике пятое условие также является обязательным – без знания ассемблерного кода как-правило может быть произведено определение отдельных раундов защитного преобразования, однако определение отдельных операций, выполняемых на этом раунде можно считать весьма сложной задачей, особенно учитывая зашумлённость снимаемых форм сигнала. Можно считать, что основная сложность реализации атаки SPA на незащищённые чипы заключается *в необходимости выполнения последних двух условий, что, применительно к реальному чипу может оказаться весьма труднореализуемым*. Необходимо отметить, что атака SPA является безальтернативной в том случае, если злоумышленнику доступно крайне ограниченное количество форм сигнала от анализируемого чипа – в предельном случае атака может быть реализована всего по одной форме сигнала.

Существует достаточно простой метод защиты от атак SPA – уменьшение соотношения сигнал-шум на снимаемых формах сигнала [48]. Это можно сделать двумя способами, принимаемыми по отдельности или совместно: уменьшение энергопотребления чипа и зашумление цепей электропитания чипа. В результате, на диаграмме, представленной на рисунке 1.12 значения отсчёта для всех весов Хэмминга будут взаимно перекрываться. Следовательно, имея небольшое количество форм сигнала (недостаточное для усреднения шумовой компоненты) злоумышленник не сможет с необходимой достоверностью определить вес Хэмминга обрабатываемых комбинаций по соответствующим всплескам. По этой причине SPA как отдельную атаку можно считать неприменимой к современным чипам, её чаще используют как вспомогательный инструмент при реализации значительно более мощной побочной атаки *дифференциального* (*разностного*) анализа мощности (DPA). Главным преимуществом атаки DPA над SPA является отсутствие необходимости наличия тестового чипа, а также детальных знаний об особенностях реализации алгоритма защитного преобразования (отметим, что хоть эти знания и не являются обязательными, однако их наличие может увеличить вероятность успешной реализации DPA). Недостаток DPA по сравнению с SPA – необходимость получения большего массива данных об энергопотреблении чипа (от сотен до тысяч форм сигнала). В следующем разделе будут более подробно описаны основы этой атаки.

1.7 Теоретические основы построения побочных атак по цепям электропитания (дифференциальный анализ мощности)

Если атака SPA предполагает непосредственный, визуальный анализ колебаний энергопотребления на снятых формах сигнала, интерпретируя которые, злоумышленник делает вывод о том какие операции, с какими данными выполняются на чипе, то атака DPA, предполагает статистический метод анализа энергопотребления чипа, что делает её значительно более эффективной чем SPA, в том случае, если злоумышленнику доступно большое количество данных об энергопотреблении чипа. За последние 20 лет, с момента первых публикаций, описывающих принципы реализации DPA его применимость, была доказана в отношении любых сколько-нибудь известных алгоритмов защитных преобразований, среди которых прежде всего блочные шифры, такие как DES [48; 55; 57; 59], AES [10, с. 119-165], ГОСТ [23, 31, 32], также алгоритмы с открытым ключом, например RSA [46]. Ниже будут приведены основные принципы реализации DPA на примере нового российского стандарта шифрования – «Кузнечик» [13]. Помимо описания известных подходов к реализации DPA, будет проиллюстрирована универсальность этой атаки. В частности, применительно к шифру «Кузнечик» она не имеет принципиальных отличий от известных реализаций в отношении других блочных шифров.

Обобщённый алгоритм DPA состоит из следующих основных шагов:

- Снимаются данные об энергопотреблении чипа при преобразовании известных сообщений на неизвестном ключе.
- Производятся оценки энергопотребления чипа при преобразовании тех же сообщений на всех возможных вариантах небольшой (переборной) части ключа (подблоке ключа).
- Полученные оценки сравниваются (статистическими методами) со снятыми данными об энергопотреблении чипа. На истинный подблок ключа будет указывать оценка, которая окажется наиболее приближенной к реальному энергопотреблению.
- Алгоритм повторяется для остальных подблоков ключа – в итоге ключ восстанавливается последовательно, по небольшим частям, с использованием дополнительной информации, утекающей по цепи электропитания чипа. То есть побочная атака DPA, в отличие от теоретических атак (то есть криптографических методов взлома), имеет полиномиальную (а не экспоненциальную) сложность в зависимости от длины ключа защитного преобразования.

В следующем подразделе каждый шаг этого алгоритма будет рассмотрен более подробно.

1.7.1 Обобщённый алгоритм атаки DPA

Со времён первых публикаций, принципы реализации DPA не претерпели существенных изменений, в обобщённом виде его алгоритм состоит из следующих пяти этапов [10, с. 119-124]:

На первом этапе реализации DPA, злоумышленник выбирает некоторое промежуточное значение алгоритма защитного преобразования $a_l(i) = f\left(m_l(i),\, k_l\right)$, зависящее от неких известных данных (обычно это подблок сообщения $m_l(i)$), и соответствующего подблока неизвестного секретного ключа k_l. Часто в качестве таких промежуточных значений выбираются выходы S-box шифра (для шифра «Кузнечик» они зависят от восьми бит раундового ключа).

На втором этапе злоумышленник производит снятие форм сигнала, характеризующих энергопотребление чипа, в процессе выполнения защитного преобразования. При этом злоумышленник должен знать, при преобразовании какого сообщения $m_l(i)$ получена соответствующая форма сигнала (обычно формы сигнала сохраняются на ПК в виде отдельных файлов, в название каждого из которых записывается соответствующее сообщение). Общую совокупность таких сообщений можно представить в виде вектора $\vec{m}_l = \left(m_l(1),\ ...,\ m_l(i),\ ...,\ m_l(N)\right)$, где N – общее количество обрабатываемых сообщений, и соответственно снятых форм сигнала. В свою очередь, каждую снятую форму сигнала можно представить в виде вектора $\vec{S} = \left(S(i,\,1),...,\ S(i,\,j),..,\ S(i,\,T)\right)$, где $S(i,\,j)$ – значение j-го отсчёта i-й формы сигнала, T – количество отсчётов i-й формы сигнала. Общую совокупность форм сигнала можно представить в виде матрицы $\mathbf{A}$ размерностью $N \times T$ (таблица 1.4). Также очень важно, чтобы значения энергопотребления для каждого столбца полученной матрицы были обусловлены выполнением одних и тех же операций, то есть необходимо чтобы они были «выравнены» (более подробно этот вопрос будет рассмотрен во второй главе).

Таблица 1.4 – Матрица значений отсчётов массива из N форм сигнала, каждая из которых состоит из T отсчётов

Номер формы сигнала	Отсчёт формы сигнала			
	1	2	…	T
1	$S(1,\,1)$	$S(1,\,2)$	…	$S(1,\,T)$
2	$S(2,\,1)$	$S(2,\,2)$	…	$S(2,\,T)$
…	…	…	…	…
N	$S(N,\,1)$	$S(N,\,2)$	…	$S(N,\,T)$

На третьем этапе составляется вектор вариантов перебираемого подблока ключа $\vec{k}'_l = \left(k'_l(1),\ ...,\ k'_l(d),\ ...,\ k'_l(K)\right)$, где K – общее количество возможных вариантов перебираемого подблока ключа. Например, если злоумышленник будет проводить атаку на шифр «Кузнечик» с использованием выходов S-box, то учитывая, что вход S-box определяется суммой восьми бит сообщения и восьми бит раундового ключа, следовательно $\vec{k}'_l = \left(0,\ 1,\ ...,\ 255\right)$. После этого на каждом варианте ключа $k'_l(d)$ с использованием известного вектора $\vec{m}_l$ (то есть совокупности сообщений), вычисляется вектор предполагаемых промежуточных значений (выбранных на первом шаге) – совокупность таких векторов для K вариантов перебираемого подблока ключа образует матрицу **B** (таблица 1.5) размерностью $N \times K$, где каждый элемент матрицы определяется следующим образом:

$$a'_l\left(i,\ k'_l(d)\right) = f\left(m_l(i),\ k'_l(d)\right), \tag{1.6}$$

где $d = 1,\ ...,\ K;\ i = 1,\ ...,\ N$

Таблица 1.5 – Матрица промежуточных значений $a'_l\left(i,\ k'_l(d)\right)$

Номер формы сигнала	Рассчитанное промежуточное значение			
	1	2	…	K
1	$a'_l(1,\ 1)$	$a'_l(1,\ 2)$	…	$a'_l(1,\ K)$
2	$a'_l(2,\ 1)$	$a'_l(2,\ 2)$	…	$a'_l(2,\ K)$
…	…	…	…	…
N	$a'_l(N,\ 1)$	$a'_l(N,\ 2)$	…	$a'_l(N,\ K)$

Каждый столбец матрицы **B** будет содержать промежуточные значения $a'_l\left(i,\ k'(d)\right)$, вычисленные на определённом варианте подблока ключа $k'(d)$ для всех обрабатываемых сообщений. С этого момента, целью атаки DPA является узнать значения из какого столбца полученной матрицы, обрабатывались чипом – зная это моментально определяется истинный вариант перебираемого подблока ключа.

На четвертом шаге, основываясь на своих знаниях об анализируемом чипе, злоумышленник выбирает модель утечки при помощи которой каждому рассчитанному промежуточному значению $a'_l\left(i,\ k'(d)\right)$, будет сопоставлен коэффициент, характеризующий предполагаемую величину энергопотребления – из таких величин образуется матрица **C** (имеющая точно такую же размерность, что и матрица **B**). К примеру если была выбрана простейшая модель веса

Хэмминга, то каждому значению $a'_l\left(i,\ k'\left(d\right)\right)$ будет сопоставлен его вес Хэмминга. Если, положим, $a'_l\left(3,\ 14\right)=34=00100010$, то в матрице $\mathbf{C}$ элемент с такими же координатами $h'_l\left(3,\ 14\right)$ будет равен $h'_l\left(3,\ 14\right)=2$, так как $\mathrm{HW}\left(00100010\right)=2$, а если $a'_l\left(3,\ 14\right)=63=11111111$, то данное значение будет соответствовать $h'_l\left(3,\ 14\right)=8$. Необходимо понимать, что идеальная модель утечки – это та модель, которая каждому рассчитанному промежуточному значению $a'_l\left(i,\ k'\left(d\right)\right)$, сопоставляла бы фактическую величину энергопотребления, однако такие модели слишком сложны, и требуют всесторонних знаний технических характеристик анализируемого чипа, которые как-правило недоступны. Поэтому обычно используют модель утечки, которая сопоставляет каждому отсчёту на снятой форме сигнала некий коэффициент (например вес Хэмминга значения $a'_l\left(i,\ k'\left(d\right)\right)$ или расстояние Хэмминга), являющийся числом, лишь косвенно характеризующим количество переключившихся логических вентилей (а следовательно и энергопотребление чипа во время обработки соответствующей комбинации).

На пятом этапе злоумышленник производит сравнение каждого отдельного столбца матрицы $\mathbf{C}$, с каждым столбцом матрицы $\mathbf{A}$, то есть вначале берётся первый столбец матрицы $\mathbf{C}$ (для первого варианта перебираемого подблока ключа), сравнивается с первым столбцом матрицы $\mathbf{A}$ (содержащим первые отсчёты N форм сигнала), затем со вторым столбцом матрицы $\mathbf{A}$ (характеризующим вторые отсчёты N форм сигнала), и так далее вплоть до T-го столбца матрицы $\mathbf{A}$.

Чаще всего [10, с. 80-83; 42; 55; 63] при сравнении элементов двух матриц применяют коэффициент линейной корреляции (в зарубежных источниках обычно применяют название «Корреляция Пирсона»):

$$c\left(k'_l\left(d\right),\ j\right)=\frac{\sum\limits_{i=1}^{N}\left[\left(h'_l\left(i,\ k'_l\left(d\right)\right)-\overline{h'_l\left(k'_l\left(d\right)\right)}\right)\cdot\left(S\left(i,\ j\right)-\overline{S\left(j\right)}\right)\right]}{\sqrt{\sum\limits_{i=1}^{N}\left(h'_l\left(i,\ k'_l\left(d\right)\right)-\overline{h'_l\left(k'_l\left(d\right)\right)}\right)^2\cdot\sum\limits_{i=1}^{N}\left(S\left(i,\ j\right)-\overline{S\left(j\right)}\right)^2}},\qquad(1.7)$$

где $h'_l\left(k'_l\left(d\right)\right)$ выборка из весов Хэмминга промежуточных значений $a'_l\left(i,\ k'_l\left(d\right)\right)$ рассчитанных на d-м варианте перебираемого подблока ключа по N сообщениям; $S\left(j\right)$ – выборка из значений j-го отсчёта по N формам сигнала; $\overline{X}$ – среднее значение выборки X.

В итоге образуется вектор из T коэффициентов корреляции $\vec{c}\left(k'_l\left(d\right)\right)=\left(c\left(k'_l\left(d\right),\ 1\right),\ c\left(k'_l\left(d\right),\ 2\right),\ ...,\ c\left(k'_l\left(d\right),\ T\right)\right)$, вначале для первого варианта

перебираемого подблока ключа, то есть определяется корреляция между оценками значений энергопотребления, рассчитанными на варианте перебираемого подблока ключа и реальными значениями отсчётов энергопотребления, на всех формах сигнала. Затем находятся такие же вектора для остальных $K-1$ вариантов подблока ключа.

При выборе истинного варианта перебираемого подблока ключа: $k_l'(d)=k_l$ значения весов Хэмминга будут рассчитаны верно: $h_l'(i,\ k_l'(d))=h_l(i)$, где $h_l(i)$ – вес Хэмминга выбранного промежуточного значения $a_l(i)$. Так как между весом Хэмминга обрабатываемых чипом данных $h_l(i)$ и амплитудой всплеска соответствующих отсчётов форм сигнала $S(i,\ j)$ существует прямо пропорциональная зависимость (рисунок 1.11б), то на корреляционном векторе будут возникать всплески в районе этих отсчётов. В районе других отсчётов столь значительных всплесков возникать не будет так как они не имеют (или имеют значительно меньшую) зависимость от $h_l(i)$. На других вариантах перебираемого подблока ключа: $k_l'(d)\neq k_l$ вес Хэмминга $h_l'(i,\ k_l'(d))$ для большей части сообщений будет рассчитан неверно, поэтому значительных всплесков на других корреляционных векторах возникать не будет.

Таким же образом, используя те же формы сигнала восстанавливаются остальные подблоки раундового ключа (задействующиеся при вычислении входов других S-box). Зная раундовый ключ первого раунда, для всех преобразуемых сообщений, злоумышленник сможет рассчитать вход второго раунда, атака на раундовый ключ которого осуществляется аналогично. Таким образом поэтапно можно восстановить раундовые ключи всех раундов, а, следовательно, и секретный ключ защитного преобразования.

1.7.2 Аспекты реализации атаки DPA с расчётом корреляционных векторов

При реализации DPA важным вопросом является выбор рассчитываемого промежуточного значения алгоритма защитного преобразования. Из приведённого в предыдущем подразделе понятно, что злоумышленник должен выбирать такое промежуточное значение $a_l(i)$, которое, при переборе вариантов подблока ключа давало бы наибольшие отличия во всплесках для истинного и ложных ключей. Например, в качестве рассчитываемого промежуточного значения можно выбрать некоторую часть результата сложения сообщения с раундовым ключом: $m(i)\oplus k$. Рассмотрим (рисунок 1.13) младший восьми битовый подблок шифра «Кузнечик»: $m_{0-7}(i)\oplus k_{0-7}$, в этом случае на истинном варианте перебираемого подблока ключа $k_{0-7}'(d)=k_{0-7}$

комбинация $a'_{0-7}\left(i,\ k'_{0-7}\left(d\right)\right)=m_{0-7}\left(i\right)\oplus k'_{0-7}\left(d\right)$ будет рассчитана верно для любых подблоков сообщения $m_{0-7}\left(i\right)$, что приведёт к значительному всплеску на корреляционном векторе. Если вариант подблока ключа будет угадан неверно: $k'_{0-7}\left(d\right)\neq k_{0-7}$, то значение $a'_{0-7}\left(i,\ k'_{0-7}\left(d\right)\right)=m_{0-7}\left(i\right)\oplus k'_{0-7}\left(d\right)$, для любых сообщений $m_{0-7}\left(i\right)$ будет отличаться от истинного минимум в одном разряде (учитывая однозначность операции сложения по mod2).

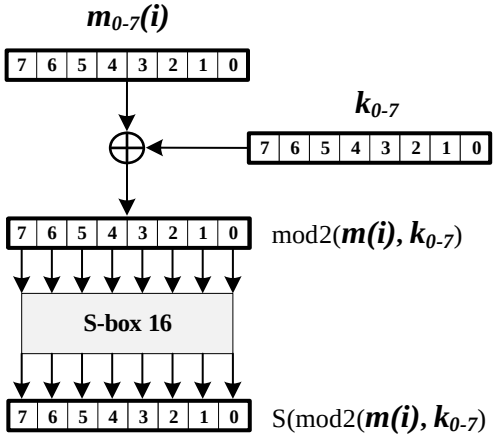


Рисунок 1.13 – Часть первого раунда шифра «Кузнечик»

Из [42] известна формула для оценки снижения корреляционного всплеска в этом случае:

$$c\left(h_l\left(k_l\right),\ h'_l\left(k'_l\left(d\right)\right)\right)=\frac{COV\left(h_l\left(k_l\right),\ h'_l\left(k'_l\left(d\right)\right)\right)}{\sigma\left(h_l\left(k_l\right)\right)\cdot\sigma\left(h'_l\left(k'_l\left(d\right)\right)\right)}=\frac{n-2e}{n},\qquad(1.8)$$

где $h_l\left(k_l\right)$ и $h'_l\left(k'_l\left(d\right)\right)$ массивы из N весов Хэмминга промежуточных значений $a'_l\left(i,\ k'_l\left(d\right)\right)$ рассчитанных соответственно на истинном и ложном вариантах перебираемого подблока ключа; n – длина комбинации $a'_l\left(i,\ k'_l\left(d\right)\right)$; e – количество бит, в которых различаются истинный и ложный подблоки ключа.

Из формулы (1.8) видно, что в наихудшем для злоумышленника случае, когда был выбран ложный вариант подблока ключа, отличающийся от истинного одним разрядом – рассчитанный в процессе реализации DPA всплеск на корреляционном векторе окажется на 25% ниже чем на истинном ключе, на остальных вариантах ключа всплеск окажется ещё ниже.

Если в качестве рассчитываемого промежуточного значения выбрать комбинацию выхода S-box «Кузнечика», в этом случае аналитическая оценка снижения корреляционного всплеска оказывается чрезвычайно сложна. Однако, моделирование показывает, что за счёт нелинейности преобразований на S-box ошибка даже в одном разряде перебираемого подблока ключа может привести к ошибке вычисления от одного до всех восьми бит выхода S-box. Хотя теоретически

веса Хэмминга полученных комбинаций и могут совпадать с весами Хэмминга комбинаций, полученных на истинном варианте перебираемого подблока ключа, однако моделирование с использованием всех возможных пар сообщение/ключ даёт вероятность такого совпадения не превышающую 30%, в то время как всплеск на корреляционном векторе, при наихудшем для злоумышленника ложном варианте подблока ключа не превысит 20.3% (при использовании восьми разрядной архитектуры чипа), что безусловно значительно лучше для злоумышленника, чем 75%-й всплеск, в случае использования входов S-box. По этой причине, при реализации атаки DPA в качестве рассчитываемого промежуточного значения защитного преобразования $a_l(i)$ обычно выбирают не результат сложения по mod2, а результат нелинейного преобразования.

Одним из ключевых преимуществ DPA по сравнению с SPA является возможность подавления практически любой шумовой компоненты на формах сигнала за счёт использования статистического усреднения. Ниже будет уточнено это качество DPA, однако сначала необходимо пояснить некоторые положения математической статистики. Одним из центральных понятий математической статистики в контексте рассматриваемой задачи является понятие корреляционной зависимости – это статистическая связь нескольких (далее будут рассматриваться только две) случайных величин, при которой изменение одной из них приводит к закономерному изменению другой, мерой корреляционной зависимости является ковариация [2, с. 358-400]:

$$COV(A,\ B)=\frac{1}{N}\sum_{i=1}^{N}\left(A(i)-\overline{A}\right)\left(B(i)-\overline{B}\right), \qquad (1.9)$$

где A, B – выборки двух случайных величин, наличие зависимости которых необходимо определить (объём каждой выборки равен N); $A(i)$, $B(i)$ – i-е значения случайных величин, $i=1,\ ...,\ N$; $\overline{A}$, $\overline{B}$ – средние значения (математические ожидания) случайных величин $A(i)$, $B(i)$.

Рассчитанное по (1.9) значение ковариации будет являться числом, характеризующим силу линейной зависимости между выборками A и B. Это число может принимать как положительные, так и отрицательные значения. В первом случае говорят о прямой зависимости, во втором – об обратной. Из (1.9) видно, что значение ковариации определяется не только силой зависимости случайных величин, но и размерностью значений этих величин (то есть единицами измерения). Это значение будет лежать в пределах $COV(A,\ B)\le\pm\sigma(A)\cdot\sigma(B)$, при этом, в случае отсутствия линейной зависимости, оно оказывается равным нулю.

Для устранения зависимости значения коэффициента ковариации от размерностей значений аргументов используется его нормировка, то есть деление коэффициента ковариации

на произведение СКО соответствующих выборок – результат представляет собой коэффициент корреляции (1.10). В итоге каждый отсчёт ковариационного вектора приводится к единой размерности. Таким образом всплески корреляционного вектора не будут зависеть от размерностей выборок, а определяются лишь степенью зависимости их значений. Значение коэффициента корреляции, всегда будет находится в интервале от −1 (для идеальной обратной зависимости) до 1 (для идеальной прямой зависимости), а в случае отсутствия зависимости, также, как и для коэффициента ковариации, значение корреляции устремится к нулю:

$$c\left(k_l'(r),\, j\right) = \frac{COV\left(h_l'\left(k_l'(d)\right),\, S(j)\right)}{\sigma\left(h_l'\left(k_l'(d)\right)\right)\cdot\sigma\left(S(j)\right)}, \tag{1.10}$$

где $c\left(k_l'(r),\, j\right)$ – коэффициент корреляции для j-го отсчёта на $k_l'(d)$-м варианте перебираемого подблока ключа; $\sigma(\cdot)$ – среднеквадратическое отклонение; $h_l'\left(k_l'(r)\right)$ – выборка значений весов Хэмминга выхода S-box, рассчитанная на варианте перебираемого подблока ключа $k_l'(d)$; $S(j)$ – выборка значений j-го отсчёта по N формам сигнала.

Положим, что в атаке DPA рассчитывается промежуточное значение в виде выхода S-box, отсчёт характеризующий энергопотребление при выработке этой комбинации обозначим как v. Значение отсчёта v, в свою очередь определяется тремя составляющими: полезной, шумовой и некоторой постоянной подложкой:

$$S(i,\, v) = b_{\text{ПОСТ}} + HW\left(a(i,\, s)\right)\cdot a_1 + \xi(i,\, v), \tag{1.11}$$

где $S(i,\, v)$ – значение v-го отсчёта i-й формы сигнала; $a(i,\, s)$ – комбинация, представляющая собой выход s-го S-box при преобразовании i-го сообщения; $HW(\cdot)$ – вес Хэмминга; a_1 – энергопотребление, затрачиваемое на выработку каждой единицы комбинации $a(i,\, s)$ (предполагается, что энергопотребление затрачиваемое на выработку логического нуля пренебрежимо мало); $\xi(i,\, v)$ – шумовая компонента, накладывающаяся на v-й отсчёт i-й формы сигнала, является случайной величиной, характеризуемой гауссовским распределением с нулевой средней; $b_{\text{ПОСТ}}$ – постоянная составляющая сигнала («подложка»), независящая от обрабатываемых данных и шумовой компоненты.

Тогда формулу для ковариации можно преобразовать к следующему виду:

$$COV\left(h_l'\left(k_l'(d)\right),\ S(v)\right)=\frac{1}{N}\sum_{i=1}^{N}\left(h_l'\left(i,\ k_l'(d)\right)-\overline{h_l'\left(k_l'(d)\right)}\right)\cdot\left(S(i,\ v)-\overline{S(v)}\right)=$$
$$=\frac{1}{N}\sum_{i=1}^{N}\left(h_l'\left(i,\ k_l'(d)\right)-4\right)\cdot\left(S(i,\ v)-\overline{S(v)}\right)=$$
$$=\frac{1}{N}\sum_{i=1}^{N}\left(h_l'\left(i,\ k_l'(d)\right)\cdot S(i,\ v)-4\cdot S(i,\ v)-h_l'\left(i,\ k_l'(d)\right)\cdot\overline{S(v)}+4\cdot\overline{S(v)}\right)= \quad , \quad (1.12)$$
$$=\frac{1}{N}\sum_{i=1}^{N}\left(h_l'\left(i,\ k_l'(d)\right)\cdot S(i,\ v)\right)-4\cdot\frac{1}{N}\sum_{i=1}^{N}\left(S(i,\ v)\right)-\overline{S(v)}\cdot\frac{1}{N}\sum_{i=1}^{N}\left(h_l'\left(i,\ k_l'(d)\right)\right)+$$
$$+4\cdot\overline{S(v)}$$

Очевидно, что значения $\frac{1}{N}\sum_{i=1}^{N}\left(S(i,\ v)\right)$ и $\frac{1}{N}\sum_{i=1}^{N}\left(h_l'\left(i,\ k_l'(d)\right)\right)$ являются математическими ожиданиями соответственно значений отсчёта v и значений рассчитанного веса Хэмминга выхода соответствующего S-box. Независимо от истинности варианта перебираемого подблока ключа, в случае преобразования случайных сообщений рассчитываемые комбинации выхода S-box будут принимать все возможные восьми битовые варианты с равной вероятностью. Следовательно, математическое ожидание их веса Хэмминга будет равно четырём. Отсюда математическое ожидание значений отсчёта v при увеличении количества обрабатываемых форм сигнала устремится к константе $b_{\text{ПОСТ}}+4\cdot a_1$:

$$\frac{1}{N}\sum_{i=1}^{N}\left(S(i,\ v)\right)=\frac{1}{N}\sum_{i=1}^{N}\left(b_{\text{ПОСТ}}+HW\left(a(i,\ s)\right)\cdot a_1+\xi(i,\ v)\right)=$$
$$=\frac{1}{N}\sum_{i=1}^{N}\left(b_{\text{ПОСТ}}+HW\left(a(i,\ s)\right)\cdot a_1\right)+\frac{1}{N}\sum_{i=1}^{N}\left(\xi(i,\ v)\right)\xrightarrow[N\to\infty]{} \quad , \quad (1.13)$$
$$\xrightarrow[N\to\infty]{} b_{\text{ПОСТ}}+\frac{1}{N}\sum_{i=1}^{N}\left(HW\left(a(i,\ s)\right)\cdot a_1\right)=b_{\text{ПОСТ}}+4\cdot a_1$$

Следовательно, формулу (1.12) можно привести к:

$$COV\left(h_l'\left(k_l'(d)\right),\ S(v)\right)\xrightarrow[N\to\infty]{}\frac{1}{N}\sum_{i=1}^{N}\left(h_l'\left(i,\ k_l'(d)\right)\cdot S(i,\ v)\right)-4\cdot\left(b_{\text{ПОСТ}}+4\cdot a_1\right)-$$
$$-\cancel{4\cdot\left(b_{\text{ПОСТ}}+4\cdot a_1\right)}+\cancel{\left(b_{\text{ПОСТ}}+4\cdot a_1\right)\cdot 4}= \quad , \quad (1.14)$$
$$=\left[\frac{1}{N}\sum_{i=1}^{N}\left(h_l'\left(i,\ k_l'(d)\right)\cdot S(i,\ v)\right)\right]-4\cdot b_{\text{ПОСТ}}-16\cdot a_1$$

Левую часть (1.14) представить следующим образом:

$$
\begin{aligned}
&\frac{1}{N}\sum_{i=1}^{N}\left(h_l'\left(i,\ k_l'(d)\right)\cdot S(i,\ v)\right)= \\
&=\frac{1}{N}\sum_{i=1}^{N}\left(h_l'\left(i,\ k_l'(d)\right)\cdot\left[b_{\text{ПОСТ}}+HW\left(a(i,\ s)\right)\cdot a_1+\xi(i,\ v)\right]\right)= \\
&=\frac{1}{N}\sum_{i=1}^{N}\left[h_l'\left(i,\ k_l'(d)\right)\cdot\left(b_{\text{ПОСТ}}+HW\left(a(i,\ s)\right)\cdot a_1\right)\right]+ \\
&+\frac{1}{N}\sum_{i=1}^{N}\left[h_l'\left(i,\ k_l'(d)\right)\cdot\xi(i,\ v)\right]\xrightarrow[N\to\infty]{} \\
&\xrightarrow[N\to\infty]{}\frac{1}{N}\sum_{i=1}^{N}\left[h_l'\left(i,\ k_l'(d)\right)\cdot\left(b_{\text{ПОСТ}}+HW\left(a(i,\ s)\right)\cdot a_1\right)\right]= \\
&=\frac{1}{N}\sum_{i=1}^{N}\left[h_l'\left(i,\ k_l'(d)\right)\cdot b_{\text{ПОСТ}}\right]+\frac{1}{N}\sum_{i=1}^{N}\left[h_l'\left(i,\ k_l'(d)\right)\cdot HW\left(a(i,\ s)\right)\cdot a_1\right]= \\
&=b_{\text{ПОСТ}}\cdot\frac{1}{N}\sum_{i=1}^{N}h_l'\left(i,\ k_l'(d)\right)+\frac{1}{N}\sum_{i=1}^{N}\left[h_l'\left(i,\ k_l'(d)\right)\cdot HW\left(a(i,\ s)\right)\cdot a_1\right]= \\
&=b_{\text{ПОСТ}}\cdot 4+\frac{1}{N}\sum_{i=1}^{N}\left[h_l'\left(i,\ k_l'(d)\right)\cdot HW\left(a(i,\ s)\right)\cdot a_1\right]
\end{aligned}
\qquad (1.15)
$$

В итоге получаем, что увеличением количества обрабатываемых форм сигнала можно свести влияние шума на результат расчёта ковариационного вектора к пренебрежительно малым значениям, кроме этого постоянная составляющая отсчёта, не оказывает никакого влияния на результат расчёта ковариации:

$$
\begin{aligned}
&COV\left(h_l'\left(k_l'(d)\right),\ S(v)\right)\xrightarrow[N\to\infty]{} \\
&\xrightarrow[N\to\infty]{} \cancel{b_{\text{ПОСТ}}\cdot 4}+\frac{1}{N}\sum_{i=1}^{N}\left(h_l'\left(i,\ k_l'(d)\right)\cdot HW\left(a(i,\ s)\right)\cdot a_1\right)-\cancel{b_{\text{ПОСТ}}\cdot 4}-16\cdot a_1= \\
&=\frac{1}{N}\sum_{i=1}^{N}\left(h_l'\left(i,\ k_l'(d)\right)\cdot HW\left(a(i,\ s)\right)\cdot a_1\right)-16\cdot a_1
\end{aligned}
\quad , \qquad (1.16)
$$

В знаменателе формулы (1.10) элементы выборки $h_l'\left(k_l'(d)\right)$ не зависят от шумовой компоненты, а элементы выборки $S(j)$ зависят, однако их влияние на успешность реализации атаки определяется рядом факторов, которые будут более подробно рассмотрены в третьей главе.

Важно подчеркнуть, что исторически первый алгоритм DPA описанный в работе П. Кёхера [48] и более подробно формализованный Т.Мессерджесом [55], предполагал в качестве промежуточного значения защитного преобразования выбирать всего один бит выхода S-box. Такой подход в настоящее время носит название «1-Bit DPA» [42, 55]. Он вычислял некоторую целевую функцию, конечный смысл которой сводился к расчёту ковариации, между рассчитанным битом и отсчётами на формах сигнала. Очевидно, что при выборе истинного

варианта перебираемого подблока ключа каждый бит выхода S-box «Кузнечика» будет рассчитан верно, а, следовательно, будет наблюдаться сильная зависимость рассчитанного значения с соответствующим отсчётом на формах сигнала. Из таблицы 1.6 видно, что при выборе ложного варианта ключа вероятность совпадения любого бита выхода S-box с истинным значением будет заметно ниже, следовательно, соответствующие всплески на ковариационном или корреляционном векторе также окажутся значительно меньше.

Таблица 1.6 – Вероятности совпадения бит выхода S-box «Кузнечика», рассчитанных на ложных вариантах перебираемого подблока ключа, с соответствующими битами, рассчитанными на истинном

	Вероятность совпадения с истинным значением	
Разряд выхода S-box	От, %	До, %
0	39.0625	62.5000
1	35.9375	62.5000
2	37.5000	60.9375
3	40.6250	59.3750
4	37.5000	60.9375
5	45.3125	53.1250
6	39.0625	59.3750
7	37.5000	62.5000

Недостаток 1-Bit DPA описанный во многих работах, в частности в [42] заключается в том, что чип обрабатывает комбинации не побитово, а подблоками, длины равной соответствующей разрядности чипа. Следовательно, скачок энергопотребления, вызванный обработкой, к примеру, выхода S-box «Кузнечика» на восьми разрядном чипе будет характеризовать значения восьми бит выхода S-box в целом. Следовательно, часть этого скачка, характеризующая энергопотребление, затрачиваемое на обработку остальных (семи неучтённых) бит будет представлять для злоумышленника дополнительный (так называемый алгоритмический) шум. Этот шум в купе с электронным шумом приводит к значительному снижению соотношения сигнал/шум, что оказывает крайне негативное влияние на успешность реализации атаки, приводя к возникновению значительных всплесков на ложных вариантах ключа. Поэтому во всех современных публикациях, авторами используется модель Multiple Bit DPA [55], предполагающая в качестве промежуточного значения выбирать комбинацию, равную разрядности чипа.

1.8 Краткая характеристика основных методов защиты от DPA

Как было показано в разделе 1.7, DPA является весьма эффективной побочной атакой, при помощи которой можно восстановить ключ шифров, считающихся устойчивыми ко всем известным криптографическим атакам. Безусловный интерес представляют предлагаемые методы защиты от атак данного типа. Анализ множества статей зарубежных авторов, а также двух монографий [10; 11] полностью посвящённых описанию и анализу атак по цепям электропитания позволяет разделить существующие методы защиты от DPA на три группы:

1. Защита на уровне программного обеспечения.
2. Защита на аппаратном уровне.
3. Защита на логическом уровне.

Защиту на уровне ПО можно считать наиболее дешёвой потому что её реализация не требует перепроектирования чипа (изменения архитектуры, периферийного оборудования и тому подобное), а предполагает внесение изменений лишь в выполняющийся программный алгоритм. Данный подход имеет всего один недостаток – увеличение требований к ресурсам чипа (вычислительная мощность, объём ОЗУ, ПЗУ). Основная идея заключается в модификации реализации алгоритма защитного преобразования таким образом, чтобы устранить зависимость энергопотребления от секретной информации, обрабатываемой чипом в процессе выполнения защитных преобразований. Существуют различные специализированные программные решения, позволяющие добиться этого (для конкретных архитектур чипов или алгоритмов защитных преобразований). Три наиболее эффективных, известных и универсальных из них это: метод маскировки, метод программной балансировки и рандомизация выполняемых чипом команд:

- Метод маскировки [10, с. 223-244; 24; 25; 35; 36; 38; 49; 52] предполагает сложение промежуточных данных, появляющихся в процессе выполнения защитного преобразования, с некоторой случайной комбинацией (маской) вырабатываемой встроенным генератором случайных чисел, и снятием данной маски после проведения всех раундов преобразования (при этом существуют такие модификации метода маскировки, которые предполагают изменение маски на каждом раунде). Для расчёта каких-либо промежуточных бит защитного преобразования, злоумышленнику кроме постоянной (и переборной) части секретного ключа придётся угадывать ещё и случайную маску. По той причине, что маска генерируется случайно и изменяется с каждым преобразуемым сообщением, её угадывание или перебор бессмысленны. Таким образом без знания маски нападающий перестаёт иметь возможность, с необходимой точностью, рассчитывать какие-либо промежуточные значения защитного преобразования, а, следовательно, исключается возможность сопоставления их с соответствующими отсчётами на формах сигнала.

- Метод программной балансировки является относительно новым и весьма перспективным подходом (первая публикация [40] появилась в 2013 году). Предлагается модификация программной реализации защитного преобразования, таким образом, чтобы комбинации, обрабатываемые на каждом такте работы чипа, состояли из одинакового количества нулей и единиц (имели одинаковый вес Хэмминга). Это достигается за счёт одновременной обработки вместе с «полезной» комбинацией, комбинации инвертированной к ней. Более подробно данный метод защиты будет рассмотрен в четвёртой главе.
- Рандомизация выполняемых процессором команд [53] основывается на идее, что в процессе выполнения чипом программы, практически всегда существует набор инструкций, которые могут быть выполнены независимо друг от друга. Инструкцию можно считать выполнимой, если она не требует результата, выполнения других инструкций, который ещё не получен, и она не изменяет данных, используемых другими инструкциями, которые пока ещё не выполнены. Каждый раз из этого набора инструкций случайным образом выбирается одна, которая и выполняется. Данный шаг делает бесполезным сопоставление рассчитанных на перебираемом ключе промежуточных бит защитного преобразования с соответствующими отсчётами на формах сигнала, по той причине, что на различных формах сигнала эти отсчёты будут случайным образом менять своё положение. Таким образом становится невозможным угадать в каком месте, на конкретной форме сигнала, находится отсчёт, характеризующий выработку рассчитываемого промежуточного значения.

Основная идея методов защиты на аппаратном уровне заключается в том, чтобы добавить в физическую реализацию чипа некоторые компоненты, влияющие на формы сигнала, и делающие крайне затруднительным их сопоставление с данными, рассчитываемыми на перебираемом ключе. К этой группе можно отнести минимум три метода защиты: зашумление, дополнительная стабилизация напряжения, десинхронизация и поддельные такты

- Зашумление – наиболее тривиальная идея [48], заключается в том, чтобы включить в состав чипа устройство, вводящее зашумление цепей питания. Добавляемый шум должен иметь амплитуду, значительно превышающую колебания полезного сигнала. Данный метод защиты лишь затрудняет реализацию DPA, не обеспечивая абсолютной защищённости. Несмотря на это, при умелой реализации, он может привести к необходимости обработки нереализуемо большого количества форм сигнала для совершения успешного взлома чипа.
- Дополнительная стабилизация напряжения также является весьма тривиальным методом защиты чипов от анализа энергопотребления [60]. Идея заключается в том, чтобы поместить в структуру чипа стабилизатор напряжения, например, выполненный в виде цепи обратной связи. Данное устройство в идеальном случае сможет обеспечить один и тот же уровень

энергопотребления независимо от обрабатываемых данных, однако на практике абсолютной независимости получить невозможно. Данный метод, по аналогии с зашумлением, лишь уменьшает амплитуду сигнальной компоненты, по сравнению с шумовой.

- По своей сути метод десинхронизации и вставки поддельных тактов [37] аналогичен методу рандомизации выполняемых процессором команд. Более того, он может быть реализован как на аппаратном уровне, так и программном. Идея заключается в том, чтобы в процессе выполнения защитного преобразования обрабатывать поддельные такты или вставлять случайные прерывания. Данные шаги позволят случайным образом изменять местоположение интересующего отсчёта на различных формах сигнала, что очевидно приведёт к невозможности реализации DPA. Однако существуют методы (так называемая пересинхронизация [43; 59; 64]) позволяющие в некоторых случаях «обойти» этот метод защиты.

Основная идея методов защиты на логическом уровне заключается в устранении первопричины утечки информации по цепи электропитания – различной мощности, потребляемой логическими вентилями в зависимости от обрабатываемых бит. В данном случае можно выделить три основных направления защиты: снижение энергопотребления, изменение структуры логических вентилей, аппаратная балансировка обрабатываемых чипом данных:

- Снижение энергопотребления предполагает использование при разработке и производстве чипа технологий, позволяющих значительно снизить его энергопотребление. В качестве примера можно привести технологию «Кремний на изоляторе» [58]. Она предполагает производство транзисторов, использующих подложку, выполненную по принципу «бутерброда», где вместо чисто кремниевой подложки используется кремний-изолятор-кремниевая. Данный шаг позволяет многократно уменьшить (но не устранить полностью) паразитную ёмкость транзисторов (следовательно, и время её зарядки) обеспечив таким образом заметное снижение энергопотребления логического вентиля во время переключения, и возрастание скорости переключения транзисторов (что влечёт увеличение скорости работы чипа). Главные недостатки данной технологии: высокая стоимость и невозможность полного устранения зависимости энергопотребления от обрабатываемых данных.
- Изменение структуры логических вентилей [50; 62] предполагает модификацию логических вентилей с целью внесения дополнительных паразитных ёмкостей, которые провоцировали бы всплески энергопотребления (примерно одинаковые) на любых четырёх переходах: $0 \to 0$, $0 \to 1$, $1 \to 0$, $1 \to 1$. На практике, данный метод защиты значительно снижает различия энергопотребления во время переключений между различными состояниями, но

не устраняет их полностью. Однако этого шага может быть достаточно, чтобы затруднить реализацию DPA до такой степени, чтобы чип, защищённый данным методом, можно было бы считать устойчивым к анализу.

- Метод аппаратной балансировки [39], по своей сути, аналогичен программной балансировке. Наиболее совершенная модификация предполагает реализацию защитного преобразования на многопроцессорной (в частности двухпроцессорной) вычислительной системе. При этом один процессор в обычном режиме выполняет операции, предусмотренные алгоритмом защитного преобразования, в то время как второй процессор параллельно выполняет другой алгоритм, построенный таким образом, чтобы поддерживать одинаковое энергопотребление чипа на каждом такте работы. Это реализуется путём обработки двоичных комбинаций, инвертированных относительно, комбинаций, обрабатываемых при выполнении алгоритма защитного преобразования. Таким образом в любой момент времени вычислительная система обрабатывает комбинации, состоящие из одинакового количества нулей и единиц.

Можно предложить ещё одно, более радикальное, направление к защите от побочных атак по цепям электропитания – использование бесключевых защитных преобразований (не требующих распределения ключей) наподобие системы Дина-Голдсмита [27; 29]. По причине того, что такая система основана на новом подходе к построению защитных преобразований, не требующем хранения в памяти ключа, реализация в её отношении побочных атак по цепям электропитания оказывается принципиально невозможной. Однако в ряде последующих работ были выявлены значительные уязвимости предлагаемой системы. Также её особенностью является то, что она предназначена только для выполнения функции защищённой передачи информации, но не для её хранения. Поэтому подобную систему пока можно рассматривать лишь как перспективную альтернативу «классическим» защитным преобразованиям защищённым описанными выше методами.

Из приведённой краткой характеристики методов защиты от DPA видно, что лишь методы маскировки, программной балансировки и аппаратной балансировки способны обеспечить строго доказанный уровень защиты. При этом метод маскировки может быть реализован сравнительно просто и с минимальными затратами лишь для небольшого количества шифров (в частности AES [52]). Адаптация маскировки к шифру «Магма» оказывается чрезвычайно сложной и требует значительных вычислительных ресурсов [25] (аналогичная ситуация будет наблюдаться и для шифра «Кузнечик»). Реализация метода программной балансировки пока существует лишь для шифра AES (и то с рядом оговорок), а вопрос возможности её адаптации к шифрам DES, «Магма» и «Кузнечик» ещё не решён, и с высокой вероятностью решён не будет. Реализацию метода аппаратной балансировки можно считать затратной и неприменимой к

дешёвым массовым продуктам, по причине значительного усложнения архитектуры чипа. Остальные, более простые и дешёвые для конечного пользователя методы, лишь затрудняют анализ чипа и основываются на следующих весьма рискованных допущениях:

- доступное злоумышленнику измерительное оборудование, имеет весьма ограниченную чувствительность;
- злоумышленник не способен значительно снизить шумовую компоненту на снимаемых формах сигнала;
- злоумышленник неспособен компенсировать рассинхронизацию отсчётов на снимаемых формах сигнала.

Эти методы защиты конечно могут применяться на практике, в отношении дешёвых чипов, в различных комбинациях, но понятно, что крайне рискованно полагаться на отсутствие у злоумышленника дорогостоящего измерительного оборудования или на его низкую квалификацию. В качестве примера можно привести статью [59], в которой авторы продемонстрировали успешную реализацию атаки на достаточно популярную смарт-карту Mifare DESFire MF3ICD40. После этого стало понятно, что высококвалифицированный злоумышленник, обладающий даже весьма средним измерительным оборудованием (более того не знающий какие именно методы защиты применяются в анализируемом чипе), способен обойти приведённые допущения и восстановить ключ защитного преобразования за небольшой промежуток времени. (Более того известны ещё как минимум три успешные атаки на коммерческие чипы: [41; 44; 57].)

1.9 Выводы по главе

В настоящей главе были систематизированы основные известные сведения о побочных атаках по цепям электропитания. В частности, было отмечено, что:

1. Зависимость энергопотребления от обрабатываемых данных вытекает из особенностей построения современных чипов и принципиально не устранима без использования специализированных, целенаправленных средств.
2. Архитектура современных чипов позволяет предложить модель их энергопотребления, при обработке тех или иных данных на отдельных узлах и при выполнении интересующих операций.
3. С использованием относительно недорогого оборудования (стоимость которого можно оценить в $15 000-20 000) возможно собрать измерительную установку, позволяющую

получить весьма качественные данные об энергопотреблении чипа, пригодные для последующей реализации побочной атаки.

4. Наиболее известны два типа побочных атак по цепям электропитания, позволяющие использовать зависимость энергопотребления чипа от обрабатываемых данных для извлечения этих данных: простая (SPA) и дифференциальная (DPA). Атака DPA является значительно более мощной, но она применима лишь в том случае, если злоумышленнику доступен большой объём данных об энергопотреблении чипа.
5. Большинство известных методов защиты от DPA лишь затрудняют реализацию атаки. Методы, позволяющие обеспечить доказанный теоретически уровень защиты оказываются весьма дорогостоящими и/или применимыми лишь в отношении ограниченного количества зарубежных шифров (например, таких как AES).

В следующей главе будут приведены результаты применения атаки DPA на реальный чип. В частности, описан комплекс технических средств, требующихся для снятия качественных данных об энергопотреблении чипа – измерительная установка, в основе которой лежит не осциллограф с активным дифференциальным пробником, а на порядок более дешёвая плата с аналого-цифровым преобразователем. Будет детально описана зависимость энергопотребления интегрального чипа от выполняемой им операции и обрабатываемых данных. Исследована эффективность реализации DPA с использованием данных, обрабатываемых на различных операциях. Раскрыт целый ряд особенностей практической реализации атак данного типа не описанных ранее в известных источниках.

Настоящая глава является обзорной, большая часть её содержания основывается на работах, других авторов, наиболее важные из них: [10; 11; 21; 48; 55].

ГЛАВА 2. ЭКСПЕРИМЕНТАЛЬНОЕ ИССЛЕДОВАНИЕ ПОБОЧНЫХ АТАК ПО ЦЕПЯМ ЭЛЕКТРОПИТАНИЯ

2.1 Описание предлагаемой измерительной установки

2.1.1 Обоснование выбора и описание устройства сбора данных

В разделе 1.5 приводилась классическая структура и параметры измерительной установки, предназначенной для реализации побочных атак по цепям электропитания. В частности, в известных атаках [10; 11; 44; 57; 59] использовались весьма дорогостоящие измерительные установки, в основе которых лежал осциллограф с активным дифференциальным пробником. Стоимость подобной установки, составляющая $10 000-15 000 можно считать вполне приемлемой по сравнению со стоимостью оборудования, требующегося для реализации других типов побочных атак (например, агрессивных атак). Но как научные исследования этих атак, так их практическое применение, имели бы значительно б**о**льшее распространение если бы удалось существенно сократить требующиеся затраты на измерительное оборудование (к примеру, до вполне разумной, по любым меркам, суммы в $1 000-2 000). Из раздела 1.5 понятно, что в контексте реализации атаки измерительная установка должна получать аналоговые данные от цепи электропитания чипа (в частности падение напряжения на резисторе, включённым в разрыв линии питания или земли), преобразовывать их в цифровой вид, а затем отправлять на ПК. При этом необходимо обеспечить соответствующую полосу пропускания и чувствительность. Сфера применения современных осциллографов весьма широка, и они не лучшим образом подходят для измерения малых сигналов [16]. Осциллограф в купе с активным дифференциальным пробником можно считать весьма избыточным оборудованием для реализации столь узкой задачи, а затраты на него – не оправданными. Поэтому в текущем эксперименте автором было принято решение использовать значительно более дешёвую измерительную установку основой которой является плата с аналого-цифровым преобразователем (АЦП). АЦП – устройство, преобразующее непрерывный аналоговый сигнал (как-правило – напряжение) в дискретный (как-правило двоичный цифровой) код, то есть последовательность чисел, отнесенных к неким фиксированным моментам времени. Для преобразования аналогового сигнала в цифровой вид АЦП выполняют две операции: дискретизацию и квантование.

Дискретизация [8, с. 18-26] – это представление непрерывного аналогового сигнала в виде последовательности отсчетов. Значения непрерывного во времени сигнала, выбираются через промежутки времени Δt (периоды дискретизации). В результате получается последовательность значений сигнала через интервал Δt. Период дискретизации очевидно определяет такой параметр измерительной установки как полоса пропускания, и характеризует способность достоверно измерять высокочастотные компоненты сигнала.

На этапе квантования [8, с. 26-29], диапазон значений сигнала разбивается на отрезки с шагом квантования δ, после чего, каждое значение отсчётов сигнала, взятых с интервалом дискретизации Δt, приводится («притягивается») к ряду дискретных уровней, количество уровней, характеризует один из основных параметров любого АЦП – разрядность, которая очевидно определяет его чувствительность. Шаг квантования определяет неустранимую погрешность присущую всем АЦП – ошибку квантования [8, с. 29-32], заключающуюся в том, что, значение сигнала, находящееся между двумя уровнями, округляются до одного из них.

Также большое значение для побочных атак по цепям электропитания имеет джиттер – в идеальном случае квантуемые отсчёты должны браться через равные промежутки времени, однако генератор, тактирующий АЦП не идеален, следовательно, из-за дрожания фронта тактового сигнала, интервал дискретизации Δt подвержен флуктуациям. В простейшем случае джиттер ведёт к искажению формы оцифрованного сигнала.

Существует множество типов АЦП, простейший из них – параллельные АЦП (АЦП прямого преобразования), они применяются в большинстве высокоскоростных осциллографов (описание принципа работы параллельных АЦП представлено в приложении А). Параллельные АЦП как-правило имеют разрешение 8-10 разрядов ($2^8 \div 2^{10}$ уровней квантования). АЦП данного типа осуществляют квантование сигнала одновременно с помощью набора компараторов, включенных параллельно источнику входного сигнала [8, с. 61-63]. Главное достоинство параллельных АЦП заключается в том, что за счёт параллельной работы всех его компараторов, время, требуемое на квантование каждого отсчёта, равно времени задержки в одном компараторе и времени задержки в преобразователе кодов, таким образом данный тип АЦП способен оцифровывать сигнал с частотой до единиц (некоторые даже до десятка) ГГц. Недостатком данных АЦП является чувствительность – для получения N разрядов, необходимо $2^N - 1$ компараторов, а следовательно для получения восьми разрядного АЦП (256 уровней квантования) необходимо 255 компараторов, а для 16-ти разрядов уже – 65 535 компараторов.

На основе параллельных АЦП строятся одни из самых популярных на сегодняшний день АЦП конвейерного типа [8, с. 63-68], упрощённо, представляющие собой сборку из двух и более последовательно подключённых секций параллельных АЦП (описание принципа работы АЦП конвейерного типа представлено в приложении Б). Частота квантования таких АЦП оказывается

ниже, чем параллельных, однако на момент проведения исследования лишь АЦП конвейерного типа, обладали частотой дискретизации выше 100 МГц при разрешающей способности 14-16 бит, то есть это единственный тип АЦП, подходивший под задачи анализа потребляемой КМОП чипами мощности.

Блок АЦП является основой любого современного осциллографа, поскольку именно это устройство дискретизирует аналоговый сигнал, переводя его в цифровой вид, который в дальнейшем выводится на дисплей осциллографа и может быть сохранён на ПК. Но ввиду того, что измерение сигналов с уровнем десятков микровольт является крайне специфической и невостребованной, подавляющим большинством пользователей, задачей, в осциллографы устанавливаются «классические» восьми разрядные АЦП, обеспечивающие необходимую большинству пользователей широкую полосу пропускания (существуют весьма редкие модели осциллографов «высокого разрешения» с АЦП на 10 или 12 разрядов, однако их стоимость оказывается чрезвычайно высокой). В контексте решения задачи анализа энергопотребления чипа, АЦП на 16 разрядов обеспечит чувствительность значительно выше любого осциллографа, построенного на основе 8-12 разрядного АЦП. Причём не потребуется предусиления сигнала, что избавляет от дополнительных искажений, вносимых усилителем, при этом обеспечивается приемлемая полоса пропускания. АЦП AD9652 фирмы Analog Devices, на момент проведения исследования имел уникальное соотношение разрешающей способности и частоты дискретизации, он обладает 16-ю уровнями квантования (в динамическом диапазоне от -1.8 В до 1.8 В, теоретически, его разрешение по напряжению составляет $U_{ШАГ} = \frac{3.6}{2^{16}} = \frac{3.6}{65\,536} = 0.0000549 = 54.9$ мкВ), при этом обеспечивается частота дискретизации до 310 МГц. Физически АЦП представляет собой интегральную схему в корпусе для поверхностного монтажа, размером 1х1 см. (рисунок 2.1).

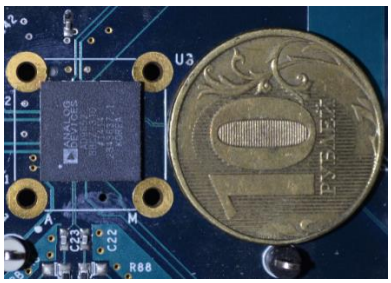


Рисунок 2.1 – Фото АЦП AD9652

Для использования АЦП в качестве высокоточной системы сбора данных его необходимо смонтировать на плату, и произвести разводку каждого контакта в соответствии с техническим руководством, что является весьма нетривиальной задачей, поэтому был приобретён АЦП уже

смонтированный на фирменную оценочную плату (AD9652-310EBZ Evaluation Board, стоимостью $420), а также плата сопряжения с ПК (HSC-ADC-EVALCZ FPGA-Based Data Capture Kit, стоимостью $620) имеющая буферную память на 64 536 отсчётов, реализацию функции триггера и интерфейс связи с ПК (USB). В комплекте с оборудованием производитель поставляет программное обеспечение «VisualAnalog», предоставляющее различные функции обработки данных, получаемых от платы согласования. Для работы АЦП требуется подключение внешнего источника тактового сигнала, при этом производитель рекомендует использовать генератор с ультранизким джиттером (Rohde & Schwarz SMA 100), стоимостью $25 000. Ввиду его высокой цены, был использован значительно менее стабильный кварцевый генератор EPSON с частотой 100 МГц, стоимость такого генератора оказывается менее $20.

2.1.2 Описание принципов монтажа анализируемого чипа

В качестве анализируемого чипа был выбран весьма популярный микроконтроллер Atmel ATmega16L-8AU в корпусе типоразмера DIP-40 [20]. Этот чип имеет достаточно высокое энергопотребление, кроме того он обладает широким диапазоном рабочих напряжений и частот, как и большинство защищённых чипов он основан на восьми разрядной архитектуре, а формат корпуса DIP-40 упрощает модификацию схемы и обеспечивает высокую доступность отдельных ножек (краткое описание принципа работы этого микроконтроллера приведено в приложении В). Для того чтобы снизить вероятность возникновения электромагнитных наводок и помех, на цепи электропитания микроконтроллера, при его монтаже на макетную плату был выдержан принцип минимальной достаточности – монтировались исключительно необходимые элементы (рисунок 2.2).

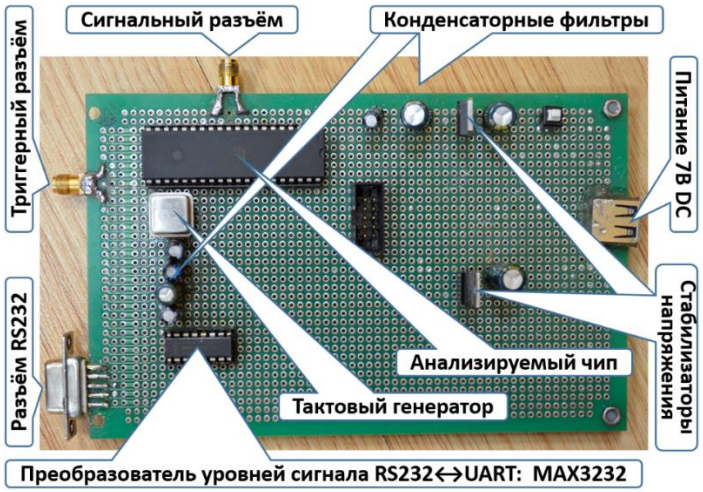


Рисунок 2.2 – Макетная плата с анализируемым микроконтроллером

Питание платы (7.5 В) осуществлялось от стабилизированного источника питания: Robiton SN1000S через USB разъём (рисунок 2.2, правая сторона платы), между которым и контактами питания микроконтроллера были смонтированы два конденсаторных фильтра (при этом последний конденсатор ставится не более чем в полусантиметре от контакта питания V_{DD} микроконтроллера), в промежутке между фильтрами установлен стабилизатор напряжения LM7805CV на котором напряжение понижается до 5 вольт (стабилизатор применяется для дополнительной защиты от просадок напряжения, в случае нестабильной работы бытовой сети). Отдельный стабилизатор напряжения, и два конденсаторных фильтра были установлены также перед контактами питания кварцевого генератора тактового сигнала и преобразователя уровней сигнала RS232↔UART. (В представленной схеме, для тактирования микроконтроллера применяется внешний кварцевый генератор, так как встроенная в чип LC цепочка имеет значительно худшую стабильность. В свою очередь ПК связан с микроконтроллером через интерфейс RS-232 со стороны ПК и блок универсального асинхронного приёмопередатчика (UART) со стороны микроконтроллера. Эти интерфейсы электрически не совместимы, поэтому для согласования сигналов, использовался преобразователь уровней сигнала MAX3232.)

Помимо этого, на плате смонтированы:

- Триггерный разъём («SMA») – он соединён с ножкой микроконтроллера, на которую, перед началом выполнения защитного преобразования, выставляется логическая единица. Этот разъём, соединяется с триггерным входом платы согласования АЦП с ПК – это необходимо для запуска сбора данных точно в тот момент, когда микроконтроллер выполняет исследуемые операции. Такой приём сильно упрощает анализ энергопотребления микроконтроллера, так как одно из основных требований, обуславливающих возможность реализации побочных атак по цепям электропитания заключается в том, что на получаемых формах сигнала, скачки напряжения, соответствующие выполнению какой-либо операции, должны располагаться в одном и том же месте на временной шкале. Такое упрощение обычно возможно лишь при использовании устройств, программируемых злоумышленником. При реализации атаки на устройство, программа которого не поддаётся изменению может применяться подход, предусматривающий запуск устройства сбора данных через определённый интервал, после отправки от ПК на чип команды, инициализирующей операции защитного преобразования. Однако этот интервал может быть не постоянным, в этом случае снятые формы сигнала, вручную (или при помощи программного алгоритма) выравниваются злоумышленником «на глаз», такая операция занимает большой объём времени и не всегда возможна.

- Сигнальный разъём («SMA») – он соединён с контактом земли микроконтроллера через резистор 10 Ом, как это показано на рисунке 1.8 и рисунке 2.3. Именно через этот разъём на плату с АЦП поступает сигнал, характеризующий энергопотребление чипа. От его качества прямо зависит успешность проводимой атаки. Сигнальный разъём через переходник непосредственно подключается к сигнальному входу платы АЦП. Данный подход исключает использование длинных соединительных проводов, которые, за счёт ёмкостной составляющей, сужали бы полосу пропускания. Такая схема включения предполагает, что на АЦП будет поступать сигнал, характеризующий падение напряжения на резисторе, включённом в разрыв линии земли чипа. По закону Ома оно будет пропорционально силе тока, протекающего по ножке GND микроконтроллера: $U = R \cdot I = 10 \text{ Ом} \cdot I$. Сила тока, в свою очередь, будет определяться энергопотреблением микроконтроллера в двух режимах: статическом (токи утечки в транзисторах) и динамическом (ток короткого замыкания, ток зарядки паразитных ёмкостей).

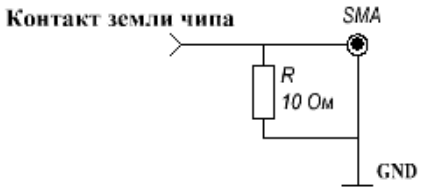


Рисунок 2.3 – Схема подключения сигнального SMA разъёма к контакту земли чипа

2.2 Оценка качества измерительной установки

Имея в распоряжении измерительную установку, обладающую достаточной чувствительностью и полосой пропускания, злоумышленник получает возможность анализировать скачки энергопотребления чипа и по их амплитуде делать выводы об обрабатываемых данных. Однако при любых реальных измерениях, на снятых формах сигнала всегда будут присутствовать шумы, которые могут значительно затруднять анализ. Их можно разделить на две большие группы [10, с. 53-58] *электронные* и *алгоритмические*.

2.2.1 Электронные шумы

Положим при проведении некоторых логических операций, измеряется энергопотребление чипа, в результате будет получена некоторая форма сигнала с определённой постоянной составляющей и присутствующими на ней скачками напряжения, характеризующими энергопотребление логических вентилей за каждый такт работы чипа. Если, не меняя обрабатываемые чипом данные, повторить эксперимент и снять вторую форму сигнала, то она будет немного отличаться от первой – данные отличия являются следствием воздействия электронного шума. При проведении реальных измерений избавиться от электронного шума невозможно, однако можно минимизировать влияние некоторых причин его возникновения.

Можно выделить следующие основные виды электронных шумов [10, с. 53-58]:

- *Шум источника питания* – наиболее очевидный вид шума, заключается в нестабильности напряжения подаваемого на анализируемый чип, что прямо отражается на зашумлённости снимаемых форм сигнала. Если запитывать чип, к примеру, от USB порта ПК, то обычные для него шумовые колебания напряжения в диапазоне 10-50 мВ не позволят различить небольшие колебания «полезного» сигнала в сотни микровольт или единицы милливольт, такой уровень шума будет очень сложно снизить до приемлемых значений даже после обработки очень большого количества форм сигнала. Для минимизации данного эффекта необходимо использовать стабильный источник питания, и дополнительно, перед контактами питания чипа устанавливать конденсаторный фильтр. Приведённая измерительная установка, сверх того, была дополнена стабилизатором напряжения.
- *Шум генератора тактового сигнала* – то есть нестабильность тактовой частоты и зашумление тактовых импульсов. Приводит к тому, что даже при обработке одних тех же данных, от измерения к измерению значения сигнала будут смещаться на временной шкале, что приведёт к искажению всплесков напряжения. Для минимизации этих эффектов рекомендуется применять качественные кварцевые генераторы, или значительно более дорогие высокоточные генераторы произвольной формы сигнала. В принципе плохое «выравнивание» форм сигнала как-правило не может привести к невозможности проведения атаки, но способно значительно увеличить требуемое количество обрабатываемых форм сигнала. Далее будет показано, что незначительные временные сдвиги можно компенсировать, специальными алгоритмами обработки снятых форм сигнала.
- *Шум питания периферийных устройств* – если от линии питания микроконтроллера, запитываются другие, смонтированные на плату устройства (генератор тактового сигнала, преобразователь уровней интерфейса RS232 и подобные узлы), то все эти компоненты могут влиять на снимаемую форму сигнала, которая помимо операций, выполняемых

непосредственно микроконтроллером будет зависеть от операций, выполняемых другими устройствами. Для минимизации данного эффекта рекомендуется использовать две линии питания – одна непосредственно для анализируемого чипа, вторая для периферийных устройств, в идеальном случае две эти линии должны быть гальванически развязаны. В эксперименте, вместо гальванической развязки использовались 2 стабилизатора с фиксированным выходным напряжением и несколько конденсаторных фильтров, что должно было свести до ничтожного уровня влияние двух линий питания друг на друга.

- *Шум электромагнитного излучения* – шум вызываемый Э/М наводками на чип как от периферийных устройств, смонтированных на ту же плату, так и от сторонних электрических приборов, включая ПК и устройство сбора данных. Данный вид шума может быть минимизирован экранированием чипа и его цепей питания, а также проведением измерений в спокойной Э/М среде.

2.2.2 Алгоритмический шум

В процессе выполнения, на анализируемом чипе, любых операций, составляющих защитный алгоритм, злоумышленника интересует энергопотребление лишь тех логических вентилей, которые переключаются при обработке данных, зависящих от секретного ключа. Однако на каждом такте одновременно с ними происходит переключение большого числа вентилей, образующих различные «системные» узлы, наиболее тривиальные из них: регистр счётчика команд (являющийся адресом следующей команды, его значение увеличивается после каждой выполненной команды), регистр команд (в который загружается кодовое обозначение следующей по порядку команды), регистр статуса (который изменяет своё значение в зависимости от ряда параметров результата, полученного, при выполнении очередной команды) и подобные.

Как говорилось ранее, при выполнении реальных измерений скачки энергопотребления чипа, обусловленные переключением множества логических вентилей в рамках одного такта его работы, как-правило, сливаются в один-два «общих» всплеска [10, с. 48; 11, с. 43-44]. Их амплитуда, помимо части, обусловленной переключением интересующих злоумышленника ЛВ, будет складываться и из энергопотребления, затрачиваемого на переключение «системных» вентилей. Мощность, потребляемая такими ЛВ не представляет для злоумышленника никакого интереса – более того она по сути является для него шумом. Этот шум определяется архитектурой анализируемого чипа, поэтому бороться с ним крайне сложно. Часть, таких алгоритмических шумовых составляющих, которая не зависит от обрабатываемых данных можно минимизировать

увеличением количества обрабатываемых форм сигнала (например, от обрабатываемых данных не зависит состояние регистра команд и счётчика команд). Другую часть алгоритмического шума, которая зависит от обрабатываемых данных (например, часть содержимого регистра статуса) к сожалению, минимизировать нельзя, что может в итоге приводить к ошибкам при восстановлении секретного ключа чипа.

2.3 Анализ характеристик снимаемых форм сигнала

Формы сигнала, снятые измерительной установкой, характеризуют падение напряжения на резисторе, вставленном в разрыв линии земли микроконтроллера, как было описано ранее, это падение напряжения пропорционально потребляемой микроконтроллером мощности, согласно [10, с. 61-63] значение каждого отсчёта формы сигнала можно охарактеризовать пятью составляющими:

- Постоянная составляющая, обусловленная токами утечки: $U_{\text{ПОСТ. ШУМ}}$.
- Постоянная составляющая, обусловленная алгоритмическим шумом: $U_{\text{АЛГ. ШУМ}}$.
- Постоянная составляющая, обусловленная выполняемой операцией (командой) $U_{\text{ОП}}$.
- Шумовая составляющая обусловленная электронным шумом (она изменяется на каждой форме сигнала): $U_{\text{ЭЛ. ШУМ}}$.
- Составляющая, зависящая от обрабатываемых данных $U_{\text{ДАН}}$.

Все перечисленные составляющие сигнала являются уникальными для каждой конкретной модели чипа, а шумовая составляющая $U_{\text{ЭЛ. ШУМ}}$ дополнительно определяется характеристиками измерительной установки, а также электромагнитной зашумлённостью окружающей среды.

Полезной для злоумышленника является составляющая $U_{\text{ДАН}}$ так как она непосредственно зависит от обрабатываемых данных. Составляющая $U_{\text{ОП}}$ может использоваться для идентификации отдельных, выполняемых чипом, операций в случае отсутствия у злоумышленника подробной информации о выполняемом анализируемым чипом алгоритме. При статистическом анализе $U_{\text{ОП}}$ не влияет на успешность атаки, так как для конкретной операции она является константой. Постоянная составляющая $U_{\text{ПОСТ. ШУМ}}$ никак не влияет на успешность проводимой атаки, так как не зависит от обрабатываемых данных и на каждом такте работы чипа будет приблизительно одинаковой: $Var\left(U_{\text{ПОСТ. ШУМ}}\right)=0$, $E\left(U_{\text{ПОСТ. ШУМ}}\right)=const$, где $Var\left(\cdot\right)$ –

дисперсия; $E(\cdot)$ – математическое ожидание. Постоянная составляющая $U_{\text{АЛГ. ШУМ}}$ влияет на успешность проводимой атаки, Так как во-первых является дополнительной шумовой компонентой (для её минимизации потребуется обработка большего количества форм сигнала), во-вторых отдельные её компоненты могут иметь зависимость от обрабатываемых данных, в этом случае её полное устранение усреднением будет невозможно. Шумовая составляющая $U_{\text{ЭЛ. ШУМ}}$ также очевидно влияет на успешность проводимой атаки. Ранее уже выдвигалась гипотеза, что шумовая составляющая $U_{\text{ЭЛ. ШУМ}}$, как-правило имеет гауссовское распределение с нулевым средним и её влияние на результат атаки может быть минимизировано путём усреднения форм сигнала. Для проверки этой гипотезы микроконтроллер был запрограммирован на многократное выполнение команды «nop» – холостая (пустая) команда, данная команда выполняется за один такт работы, при этом содержимое регистров данных, ОЗУ, ПЗУ, шины данных не изменяются. Перед выполнением этих операций, микроконтроллер отсылал на АЦП триггерный сигнал (логическую «1»), который запускал сбор данных, и последующую выгрузку полученной формы сигнала на ПК. Всего было снято 102 400 форм сигнала. Пример одной из них представлен на рисунке 2.4.

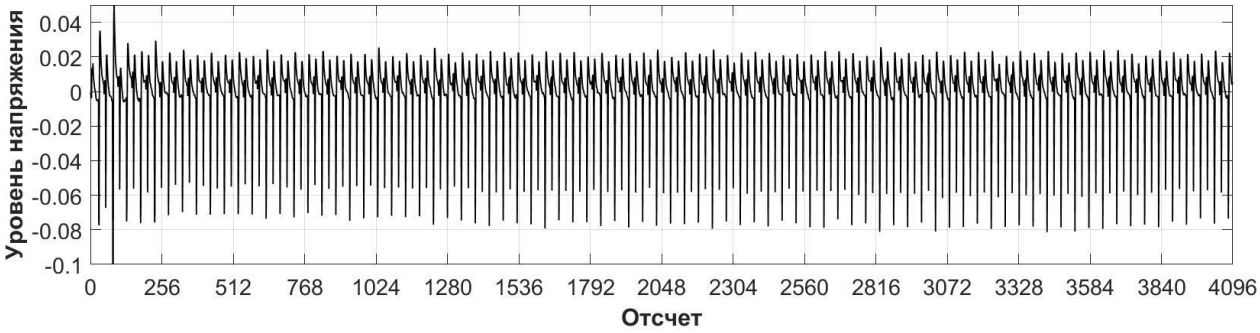


Рисунок 2.4 – Форма сигнала, характеризующая энергопотребление микроконтроллера Atmel при выполнении операций «nop»

Необходимо пояснить некоторые особенности получаемых от АЦП форм сигнала:

- Входной тракт АЦП удаляет постоянную низкочастотную составляющую сигнала, однако, на эффективность реализации побочных атак по цепям электропитания эта особенность не окажет влияния, так как для них важны лишь относительные, высокочастотные изменения сигнала.
- На оси ординат и оси абсцисс откладываются не абсолютные значения напряжения и времени, а относительные. По оси ординат отложены уровни напряжения (привести которые к абсолютным значениям можно соотнеся их с динамическим диапазоном АЦП от -1.8 В до 1.8 В. (Например, значение на оси ординат, равное одному уровню напряжения

будет эквивалентно 1.8 В, значение -0.5 – будет эквивалентно -0.9 В.) По оси абсцисс отложены отсчёты, интервал между двумя соседними отсчётами эквивалентен временному интервалу, обратному частоте дискретизации тактового генератора АЦП, то есть $1/100\ \text{МГц} = 10\ \text{нс}$. Также необходимо заметить, что по скачкам напряжения можно весьма просто определить протекающий ток и потребляемую мощность. Учитывая, что формы сигнала характеризуют падение напряжения на резисторе сопротивление которого составляет $R_{\text{ИЗМ}} = 10\ \text{Ом}$, то по закону Ома мгновенное значение протекающего через него тока определяется следующим образом:

$$I_{\text{ОБЩ}}(t) = \frac{U_{\text{ОБЩ}}(t)}{R_{\text{ИЗМ}}} = \frac{U_{\text{ОБЩ}}(t)}{10\ \text{Ом}}, \tag{2.1}$$

где $U_{\text{ОБЩ}}(t)$ – падение напряжения на резисторе снятое в момент времени t; $I_{\text{ОБЩ}}$ – сила тока, протекающего по цепи электропитания микроконтроллера в тот же момент времени.

Потребляемая мощность определяется как:

$$P_{\text{ОБЩ}}(t) = I_{\text{ОБЩ}}(t) \cdot U_{\text{ПИТ}} = \frac{U_{\text{ОБЩ}}(t)}{R_{\text{ИЗМ}}} \cdot U_{\text{ПИТ}} = \frac{U_{\text{ОБЩ}}(t)}{10\ \text{Ом}} \cdot 5\ \text{В}, \tag{2.2}$$

где $U_{\text{ПИТ}} = 5\ \text{В}$ – напряжение питания чипа.

Таким образом, мгновенное значение потребляемой чипом мощности можно получить делением соответствующего значения напряжения (определяемого по снятым формам сигнала) на коэффициент 0.5.

В данном эксперименте, для тактирования микроконтроллера использовался кварцевый генератор с частотой 2 МГц, поэтому длительность одного такта работы микроконтроллера составляет порядка $1/2\ 000\ 000\ \text{Гц} = 500\ \text{нс}$, что соответствует 50-ти отсчётам на снятых формах сигнала. На рисунке 2.5, каждые 2 всплеска (например, с 1933 по 1983 отсчёты) как-раз имеют длительность в 50 отсчётов, отсюда логично допустить, что каждая пара всплесков характеризует энергопотребление микроконтроллера на одном такте работы.

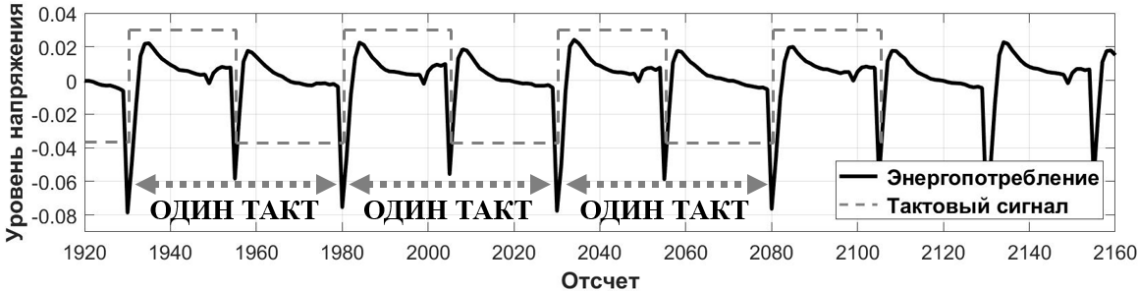


Рисунок 2.5 – Участок формы сигнала, полученной при выполнении команды «nop»

Учитывая работу [45] можно предположить, что отрицательный всплеск, в середине такта (рисунок 2.6), и пропорциональные его амплитуде части всплесков на концах такта обусловлены током короткого замыкания $I_{КЗ}$. Токами зарядки паразитных ёмкостей I_C обусловлены небольшие части всплесков на концах такта. Необходимо подчеркнуть, что приведённые на рисунке 2.6 всплески характеризуют энергопотребление всех ЛВ составляющих чип в соответствующий момент времени. (Среди них находятся ЛВ, обрабатывающие информацию представляющую интерес для злоумышленника).

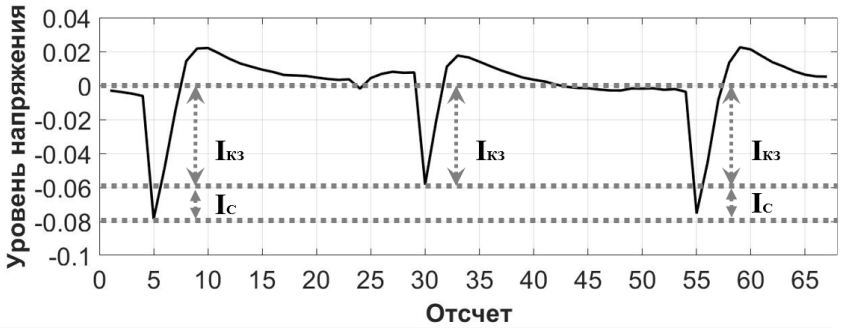


Рисунок 2.6 – Участок формы сигнала, характеризующий энергопотребление на одном такте работы чипа

За счёт нестабильности тактовых генераторов микроконтроллера и АЦП, всплески на полученных формах сигнала от измерения к измерению будут смещаться по временной шкале на 5-7 отсчётов, то есть 50-70 нс. (рисунок 2.7а). Очевидно, что данный эффект будет серьёзно затруднять машинный анализ энергопотребления при выполнении какой-либо команды, следовательно, требуется выравнивание флуктуаций форм сигнала (рисунок 2.7б). Выравнивание предполагает совмещение всплесков напряжения в границах рассматриваемого такта. Если рассматривать две формы сигнала, изображённые на рисунке 2.7а, то необходимо совместить всплеск на 1938-м отсчёте первой формы сигнала со всплеском на 1934-м отсчёте второй формы сигнала, это можно сделать, например сдвинув вторую форму сигнала на 5 отсчётов вправо. Вдобавок к этому, для снижения времени обработки и объёма памяти, требующегося для хранения форм сигнала, их можно обрезать справа и слева, оставив лишь отсчёты, характеризующие рассматриваемые такты работы микроконтроллера.

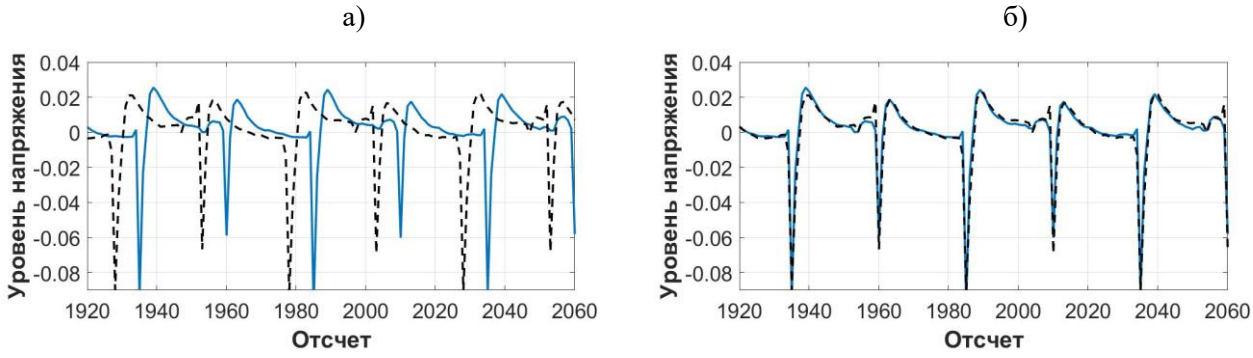


Рисунок 2.7 – Две формы сигнала: а) до; б) после выравнивания флуктуаций

Технически, процесс выравнивания заключается в задании «окна» в котором находится некий ярко выраженный всплеск, в приводимом примере это колебания напряжения между 1920-м и 1945-м отсчётами. В этом окне, для каждой формы сигнала производится поиск максимального отрицательного всплеска (то есть минимального значения). Затем, считываются 15 отсчётов до этого значения и 125 отсчётов после. Таким образом происходит некая «привязка» к отрицательному всплеску, позволяющая весьма качественно выровнять все 102 400 форм сигнала.

В этом случае гистограмма распределения, к примеру 1938-го отсчёта (максимальный положительный всплеск на первом такте рисунка 2.7б) примет классическую гауссовскую форму колокола (рисунок 2.8б), в отличие от распределения того же отсчёта до выравнивания форм сигнала (рисунок 2.8а).

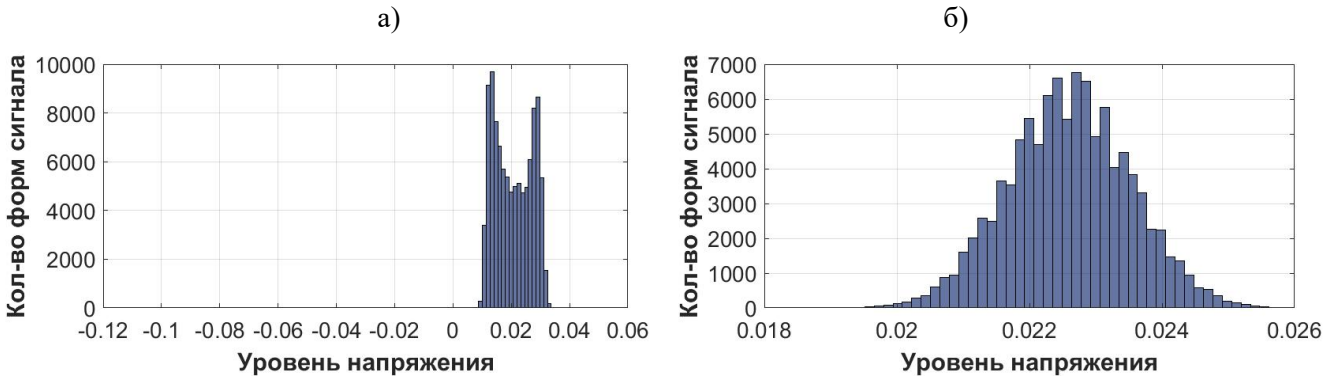


Рисунок 2.8 – Гистограммы распределения значений 1938-го отсчёта, а) до; б) после выравнивания форм сигнала

Математическое ожидание рассматриваемого отсчёта составляет порядка 0.0226 (что эквивалентно 41 мВ). Учитывая правило трёх сигм, которое гласит, что 0.9973% значений случайной величины лежит в диапазоне $(x-3\sigma;\ x+3\sigma)$, где σ – среднеквадратическое отклонение, получается, что СКО значений отсчёта составляет порядка 0.00094 (что

эквивалентно 1.7 мВ). Значения отсчёта на около 0.0027% (264) формах сигнала будут выходить за границы $\pm 3\sigma \left(= 0.00282 \sim 5.1 \text{ мВ} \right)$.

Минимальное значение, принимаемое 1938-м отсчётом, составляет порядка 0.004 (7.2 мВ), максимальное – порядка 0.04 (72 мВ), на рисунке 2.9 приведены две формы сигнала (крайние случаи), на которых 1938-й отсчёт принимает указанные значения, отсюда видно, что даже в таких «крайних» случаях алгоритм выравнивания выполняется корректно.

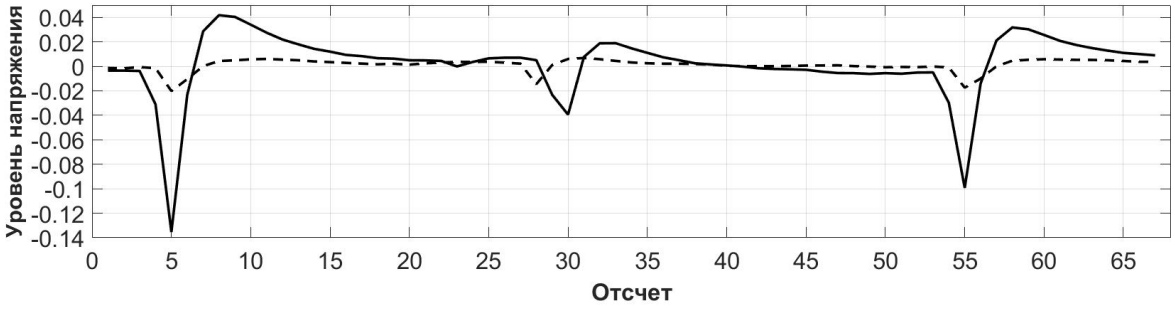


Рисунок 2.9 – Две формы сигнала, на которых 1938-й отсчёт имеет максимальное (сплошная линия) и минимальное (пунктирная линия) значения; формы сигнала являются выравненными и, для оптимизации процесса их последующей обработки, обрезанными с права и слева, в результате чего, 1938-й отсчёт сместился на место пятого отсчёта.

Так как в данном примере микроконтроллер выполнял одну и ту же операцию, и наблюдение за энергопотреблением производилось в одном и том же отсчёте снятых форм сигнала, следовательно $Var\left(U_{\text{ПОСТ. ШУМ}}\right) = Var\left(U_{\text{АЛГ. ШУМ}}\right) = Var\left(U_{\text{ОП}}\right) = Var\left(U_{ДАН}\right) = 0$, $E\left(U_{\text{ПОСТ. ШУМ}}\right) = const$, $E\left(U_{\text{АЛГ. ШУМ}}\right) = const$, $E\left(U_{\text{ОП}}\right) = const$, $E\left(U_{\text{ДАН}}\right) = const$. Это означает, что электронный шум в используемой измерительной установке имеет СКО равное порядка 5.1 мВ, при нулевом математическом ожидании (после выполнения процедуры минимизации шума тактового генератора).

2.4 Исследование зависимости энергопотребления микроконтроллера от выполняемых операций и обрабатываемых данных

Выше были описаны шумовые характеристики снятых форм сигнала, когда микроконтроллер не выполняет операций с какими-либо данными. В текущем разделе будет проанализирована зависимость значений форм сигнала от выполняемых операций и

обрабатываемых на этих операциях данных. К наиболее часто используемым при реализации защитных алгоритмов, командам можно отнести: запись в ОЗУ, чтение из ОЗУ, чтение из ПЗУ, копирование данных между регистрами, сложение по mod2 (XOR). Ячейки назначения могут иметь предварительное заполнение данными – в этом случае происходит перезапись старых данных новыми, следовательно, согласно теоретическим доводам, описанным в разделе 1.4, необходимо применять модель расстояния Хэмминга. Так же они могут быть чистыми (не содержащими никаких данных, то есть заполненные нулями или единицами), в этом случае, модель расстояния Хэмминга будет эквивалентна модели веса Хэмминга.

2.4.1 Зависимость энергопотребления от обрабатываемых данных при выполнении операции записи в чистую ячейку ОЗУ

На первом этапе микроконтроллер был запрограммирован на запись случайной восьми битовой комбинации (поступившей в его рабочий регистр от ПК) в чистую ячейку внутренней ОЗУ. Для каждого возможного варианта восьми битовой двоичной комбинации было снято около 200 форм сигнала, характеризующих энергопотребление чипа (то есть всего $256 \cdot 200 = 51\,200$ форм сигнала). Снятые формы сигнала оказываются весьма похожими, на формы сигнала, полученные при выполнении команд «nop», кроме одной особенности – на всех формах сигнала в районе 1 300-го отсчёта (рисунок 2.10) заметен скачок напряжения (всплеск напряжения от 0-го до 300-го отсчёта вызван энергопотреблением, затрачиваемым на посылку триггерного сигнала и от обрабатываемых данных не зависит, поэтому здесь и в дальнейшем он учитываться не будет).

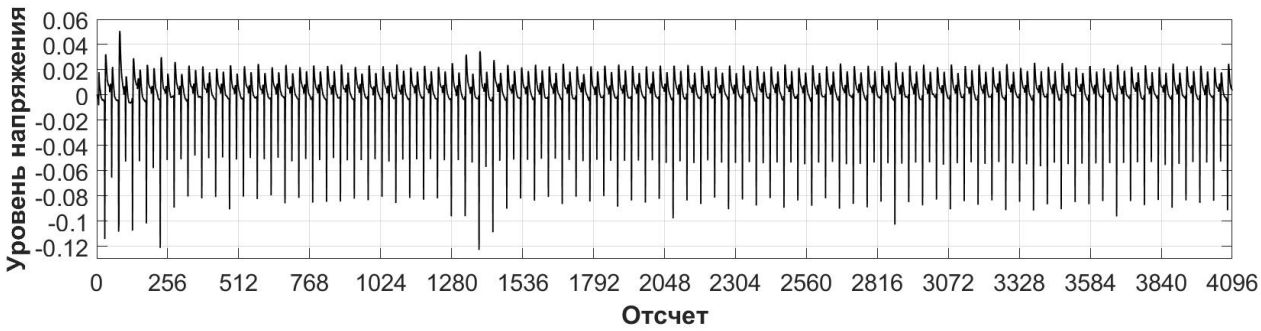


Рисунок 2.10 – Форма сигнала, характеризующая энергопотребление микроконтроллера при выполнении операции записи в чистую ячейку ОЗУ

Увеличение масштаба участка, на котором происходит скачок напряжения (рисунок 2.11а), позволяет сделать вывод, что его длительность составляет порядка 3-ёх тактов или 150 отсчётов

(с 1328-го по 1478-й отсчёты). Вместе с этим из технической документации на микроконтроллер [20] известно, что операция записи байта данных в ОЗУ занимает 2 такта. Получается, что скачок энергопотребления, обусловленный выполнением рассматриваемой операции, происходит не только на тех тактах, на которых данная операция выполняется, но и затрагивает соседние такты. Можно допустить, что причиной этого явления являются задержки в распространении сигнала и ёмкостные составляющие контактных дорожек, которым необходимо некоторое время на разряд, более подробно этот эффект будет описан ниже. Факт того, что данный временной интервал в действительности соответствует операции записи в ОЗУ, подтверждается и анализом программного кода микроконтроллера – задержка между посылкой триггерного сигнала и операцией записи в ОЗУ составляет 25 тактов, что соответствует приблизительно 1 250 отсчётам на форме сигнала.

Снятые формы сигнала, ожидаемо оказались рассинхронизированными и нуждались в выравнивании. Эмпирически было выяснено, что наиболее надёжная привязка осуществляется по максимальному значению, находящемуся в окне между 1360-м и 1410-м отсчётами. При выполнении операции выравнивания из форм сигнала вырезался интервал в 500 отсчётов (включая 2.5 такта до выполнения рассматриваемой операции и пять тактов после), что позволило сократить время их дальнейшей машинной обработки (рисунок 2.11б).

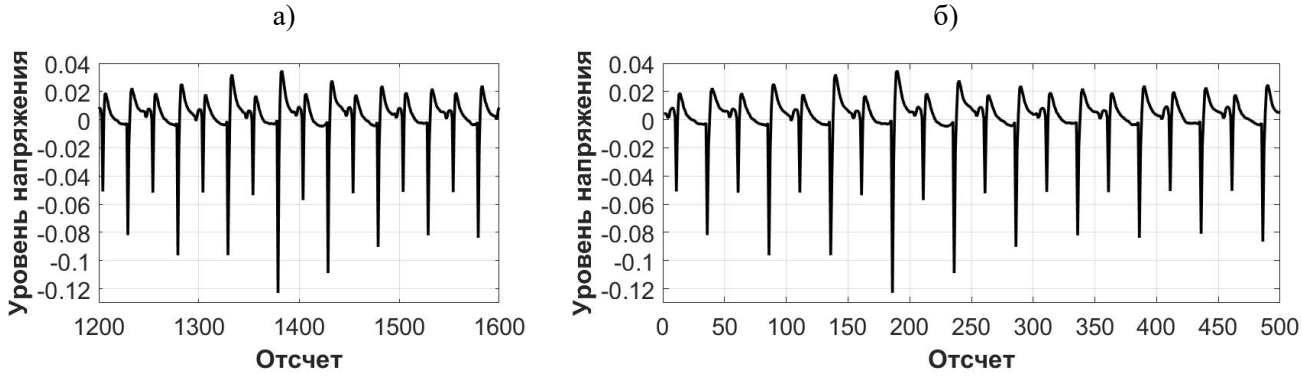


Рисунок 2.11 – Увеличенный участок формы сигнала, характреизующей энергопотребление микроконтроллера при выполнении операции записи в чистую ячейку ОЗУ: а) до выравнивания; б) после выравнивания форм сигнала и вырезания рассматриваемого участка

Наибольший интерес представляют такты, которым соответствуют два максимальных всплеска на 140-м и 190-м отсчётах форм сигнала (пример одной из которых изображён на рисунке 2.11б). На рисунках 2.12 построены гистограммы распределения их значений (далее рассматриваемый, на форме сигнала отсчёт, будет называться целевым отсчётом). При их сравнении с гистограммой, представленной на рисунке 2.8б, видно, что для 140-го отсчёта математическое ожидание сместилось в сторону б**о**льших значений приблизительно на 0.0085

уровня напряжения (что эквивалентно 16.5 мВ): с 0.0226 (41 мВ) до 0.0311 (57.5 мВ), а для 190-го отсчёта на 22.9 мВ, то есть до 0.0355 (63.9 мВ), СКО также увеличилось более чем в 2 раза с 0.00094 (1.7 мВ) до 0.00246 (4.4 мВ) для 140-го отсчёта и до 0.00199 (3.6 мВ) для 190-го отсчёта.

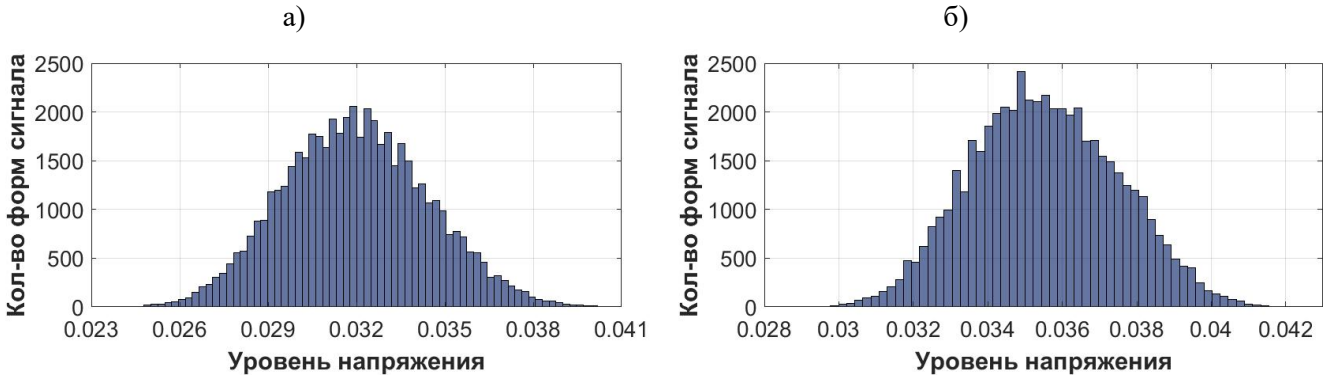


Рисунок 2.12 – Гистограммы распределения значений: а) 140-го; б) 190-го отсчётов.

Логично допустить, что смещение математического ожидания было вызвано выполняемой операцией с обрабатываемыми данными. Для проверки этой гипотезы были построены отдельные гистограммы распределения значений 140-го отсчёта для массивов форм сигнала соответствующих обработке каждого из девяти возможных весов Хэмминга. Вначале были отобраны формы сигнала, полученные при сохранении в ОЗУ комбинаций с весом Хэмминга равным «0», затем с весом Хэмминга «1» и так далее вплоть до веса Хэмминга равному восьми. После чего, для каждого составленного массива форм сигнала, была построена отдельная гистограмма распределения значений 140-го отсчёта, и такие же действия были реализованы в отношении 190-го отсчёта. Огибающие гистограмм для всех девяти весов Хэмминга сведены в едином масштабе на рисунках 2.13, из чего отчётливо видно, что для каждого веса Хэмминга, значения целевого отсчёта имеют нормальное распределение с приблизительно одинаковым СКО и различным средним значением (то есть математическим ожиданием). Данные гистограммы являются экспериментальным подтверждением выдвинутой ранее гипотезы, согласно которой, энергопотребление микроконтроллера пропорционально зависит от количества переключившихся логических вентилей, а, следовательно, от веса Хэмминга обрабатываемых комбинаций.

а) б)

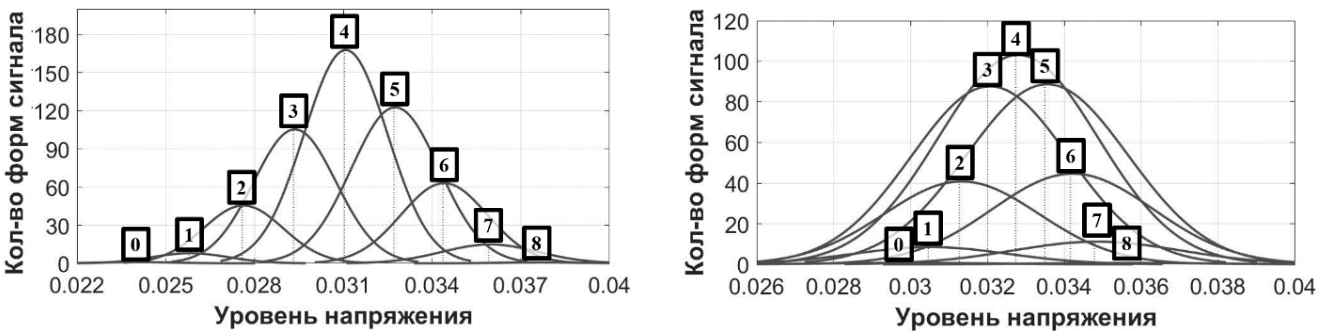


Рисунок 2.13 – Огибающие гистограмм распределения значений: а) 140-го; б) 190-го отсчётов, в зависимости от веса Хэмминга обрабатываемых байт (указан в квадратах)

В таблице Г.1 приложения Г для каждого веса Хэмминга приведено математическое ожидание и СКО значений для 140-го и 190-го отсчётов. Из них видно, что каждое увеличение на единицу веса Хэмминга обрабатываемых комбинаций, приводит к увеличению среднего значения целевого отсчёта примерно на 2.3 мВ для 140-го отсчёта и на 1.2 мВ для 190-го отсчёта. СКО шума незначительно увеличивается с увеличением веса Хэмминга, в среднем на 0.2 мВ для 140-го отсчёта и на 0.05 мВ для 190-го отсчёта, его усреднённое значение составляет 2.6 мВ для 140-го отсчёта и 2.9 мВ для 190-го отсчёта.

Из полученных экспериментальных результатов можно сделать вывод, что на формах сигнала всегда присутствует электронный шум $U_{\text{ЭЛ. ШУМ}}$, который зависит от обрабатываемых данных. Вместе с этим СКО шума оказывается соизмеримо или больше отличий математических ожиданий целевого отсчёта для соседних весов Хэмминга. Используя правило трёх сигм, можно определить, что для 140-го отсчёта, для порядка 60% форм сигнала шумовая компонента не будет выходить за границы $(-2.3; +2.3)$ мВ, следовательно для этих форм сигнала возможно однозначное определение веса Хэмминга по значению отсчёта. Для остальных 40% форм сигнала, определение веса Хэмминга, сохраняемых в ОЗУ данных, по значению отсчёта на отдельной форме сигнала, окажется практически невозможным, по причине наложения на данный отсчёт шумовой компоненты, превосходящей отличия математических ожиданий целевого отсчёта для соседних весов Хэмминга. Возможность однозначного определения веса Хэмминга обрабатываемой комбинации по значению 190-го отсчёта будет существовать для порядка 40% форм сигнала, за счёт уменьшения соотношения сигнал/шум.

Значение целевого отсчёта можно описать следующей математической моделью:

$$U_{\text{ОБЩ}} = U_{\text{СИГН}} + U_{\text{ЭЛ. ШУМ}}, \tag{2.3}$$

где $U_{\text{СИГН}} = U_{\text{ДАН}} + U_{\text{ОП}} + U_{\text{ПОСТ. ШУМ}} + U_{\text{АЛГ. ШУМ}}$ – скачок напряжения, который определяется выполняемой микроконтроллером операцией и весом Хэмминга обрабатываемых данных, его среднее значение окажется сдвинутым за счёт постоянного и алгоритмического шума.

Важно заметить, что, при рассмотрении на формах сигнала отсчёта, соответствующего выполнению одной и той же операции, с одними и теми же данными, происходящей в одно и тоже время, компоненты $U_{\text{ДАН}}$, $U_{\text{ОП}}$, $U_{\text{ПОСТ. ШУМ}}$ и $U_{\text{АЛГ. ШУМ}}$ будут постоянными.

Для приведённых выше данных на 140-м отсчёте $E\left(U_{\text{ДАН}}\right) = 2.3$ мВ, на 190-м отсчёте второго такта $E\left(U_{\text{ДАН}}\right) = 1.2$ мВ. Для определения $E\left(U_{\text{ОП}}\right)$, необходимо знать математическое ожидание наибольшего всплеска напряжения на такте, на котором микроконтроллер не выполняет операций (выполняет пустую операцию), из предыдущего раздела известно, что оно составляет 41 мВ. Следовательно, учитывая то, что при сохранении в ОЗУ комбинации с нулевым весом Хэмминга (в процессе чего, предположительно, не происходит переключения логических вентилей ОЗУ) математическое ожидание наибольшего всплеска (а именно 190-й отсчёт на втором такте) составляет 52.8 мВ, тогда $E\left(U_{\text{ОП}}\right) = 58.2 - 41 = 17.2$ мВ, и соответственно $E\left(U_{\text{ПОСТ. ШУМ}}\right) + E\left(U_{\text{АЛГ. ШУМ}}\right) = 41$ мВ.

$U_{\text{ЭЛ. ШУМ}}$ – представляет собой шумовую составляющую, с нулевой средней $E\left(U_{\text{ЭЛ. ШУМ}}\right) = 0$ и гауссовским распределением, характеризуемым некоторой дисперсией $Var\left(U_{\text{ЭЛ. ШУМ}}\right) = \sigma^2_{\text{ЭЛ. ШУМ}}$. Для 140-го отсчёта она составляет $2.6^2 = 6.76$, а для 190-го отсчёта – $2.9^2 = 8.41$.

Можно показать, что усреднение большого (в идеальном случае бесконечного) количества форм сигнала для каждого веса Хэмминга, приведёт к тому, что будет найдено математическое ожидание каждого (в том числе целевого) отсчёта:

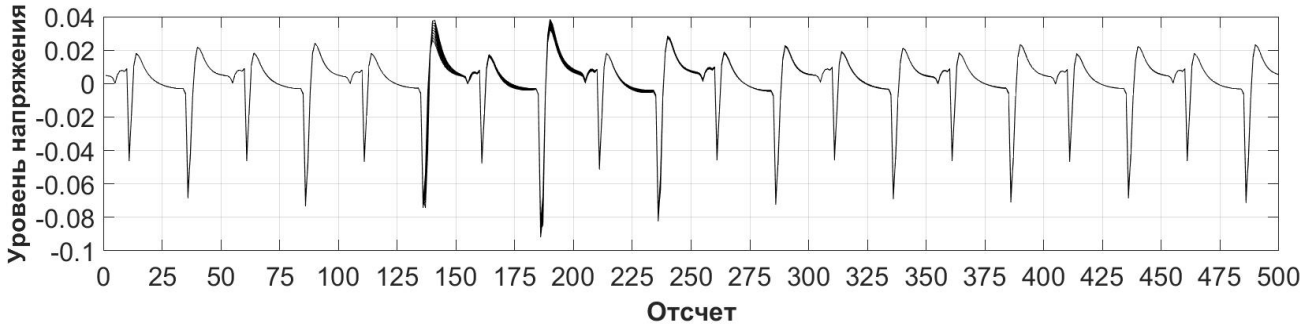
$$E(v) = \frac{1}{N}\sum_{1}^{N} U_{\text{ОБЩ}} = \frac{1}{N}\sum_{1}^{N}\left(U_{\text{СИГН}} + U_{\text{ЭЛ. ШУМ}}\right) = \frac{1}{N}\sum_{1}^{N} U_{\text{СИГН}} + \frac{1}{N}\sum_{1}^{N} U_{\text{ЭЛ. ШУМ}} \xrightarrow[N\to\infty]{} \xrightarrow[N\to\infty]{} U_{\text{СИГН}} + \cancel{\frac{1}{N}\sum_{1}^{N} U_{\text{ЭЛ. ШУМ}}}, \quad (2.4)$$

где $E(v)$ – математическое ожидание v-го отсчёта.

Поэтому на усреднённой форме сигнала шумовая составляющая целевого отсчёта устремится к нулю (так как $E\left(U_{\text{ЭЛ. ШУМ}}\right) = 0$), и значение отсчёта устремится к $U_{\text{СИГН}}$ (так как $E\left(U_{\text{СИГН}}\right) = \text{const}$).

Для проверки этой гипотезы на практике, выровненные 51 200 форм сигнала, были усреднены для каждого из девяти весов Хэмминга, то есть совокупность форм сигнала была разбита на девять массивов в зависимости от веса Хэмминга обрабатываемого байта, после чего по каждому массиву была рассчитана усреднённая форма сигнала. Девять усреднённых форм сигнала для различных весов Хэмминга представлены на рисунках 2.14.

а)



б)

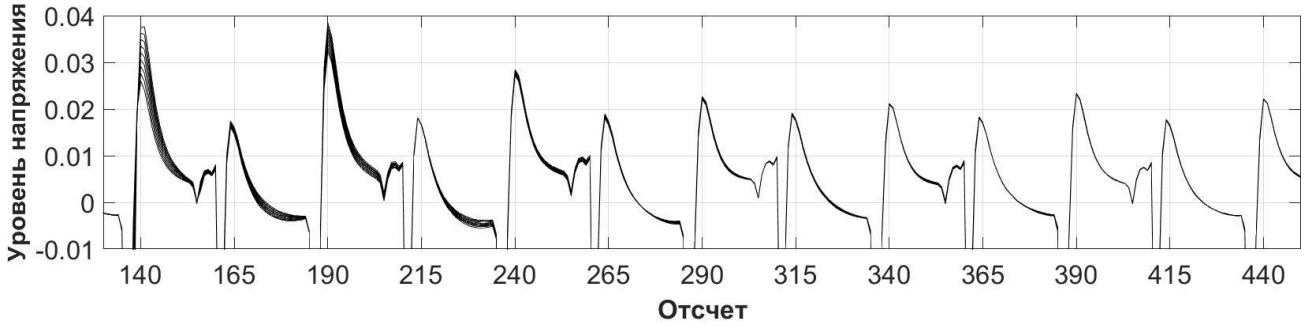


Рисунок 2.14 – Девять усреднённых форм сигнала для различных весов Хэмминга при выполнении операции записи в чистую ячейку ОЗУ

Увеличение масштаба представленной на рисунке 2.14а формы сигнала позволяет сделать вывод, что зависимость энергопотребления чипа от обрабатываемых данных существует на шести тактах работы микроконтроллера (увеличенные области усреднённых форм сигнала представлены на рисунках 2.14б и Д.1 приложения Д). При этом сила зависимости уменьшается от такта к такту, непостоянным является и характер зависимости – в некоторых случаях она прямая, в других – обратная. На рисунке 2.15 приведена увеличенная область формы сигнала, характеризующая энергопотребление чипа на первом такте с 136 по 186 отсчёты. Зависимость от данных проявляется по всей области формы сигнала. Для упрощения характеристики зависимостей, каждый такт был разбит на четыре области: первая и третья – максимальные отрицательные всплески на такте (включая область справа, в которой они нарастают), вторая и четвёртая – максимальные положительные (включая область справа, в которой они спадают).

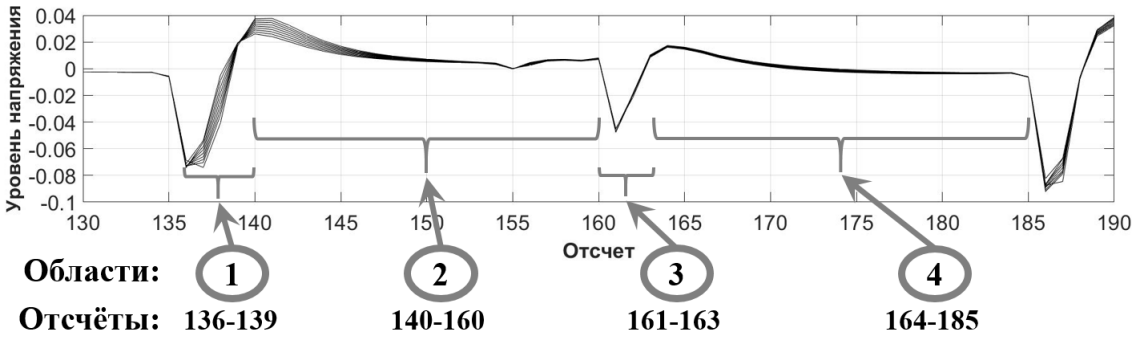


Рисунок 2.15 – Девять усреднённых форм сигнала для различных весов Хэмминга, установление границ областей рассматриваемых тактов

Максимальная разница в энергопотреблении при обработке данных с разными весами Хэмминга возникает в первой и второй областях первого такта (соответственно до $U_{\text{ДАН}} = 7.2$ мВ и $U_{\text{ДАН}} = 2.7$ мВ для каждого увеличения веса Хэмминга на единицу), зависимость в третьей области намного слабее (до $U_{\text{ДАН}} = 0.46$ мВ), зависимость в четвёртой области также весьма слабая (до $U_{\text{ДАН}} = 0.44$ мВ). Также необходимо обратить внимание, что наибольшая зависимость от обрабатываемых данных проявляется не в пиковых значениях форм сигнала, а в областях, расположенных правее (областях нарастания и спада сигнала). Такая же структура зависимости наблюдается и на последующих тактах. Также интересно отметить, что в разных областях такта направление зависимости произвольным образом изменяется от прямой на обратную.

Очевидно, что чем больше значение $U_{\text{ДАН}}$ по отношению к $U_{\text{ЭЛ. ШУМ}}$ (чем сильнее сигнал, превышает шум), тем меньшее количество форм сигнала будет необходимо усреднить, для однозначного определения веса Хэмминга обрабатываемой микроконтроллером комбинации на определённом такте. Обычно для характеристики отношения сигнала к шуму используют так называемый коэффициент *сигнал/шум* (*SNR*), который определяется по известной формуле $SNR = \dfrac{Var(Сигнал)}{Var(Шум)}$, под $Var(Сигнал)$ необходимо понимать дисперсию $U_{\text{ДАН}}$ для различных весов Хэмминга, под $Var(Шум)$ – дисперсию шума. В контексте побочных атак по цепям электропитания можно, сказать, что чем больше значение SNR, тем больше информации «утекает» по цепи электропитания.

Значения $U_{\text{ДАН}}$, СКО и SNR для каждой из четырёх областей каждого из шести тактов, на которых существует зависимость энергопотребления от веса Хэмминга обрабатываемых данных приведены в таблице Е.1 приложения Е. При этом в границах каждой области выбирался один отсчёт на котором значение $U_{\text{ДАН}}$ является максимальным (максимальная зависимость энергопотребления от обрабатываемых данных). К примеру, во второй области первого такта

(140-160-й отсчёты) максимальное значение $U_{\text{ДАН}}$ достигается на 141-м отсчёте – в таблице приведены значения $U_{\text{ДАН}}$, СКО и SNR для данного отсчёта, также указано направление зависимости – прямая или обратная. В графе «Особые отметки» проставлялся знак «*» в том случае если в указанном отсчёте зависимость была нарушена для одного веса Хэмминга. Если во *всех* отсчётах соответствующей области зависимость нарушалась для двух или более весов Хэмминга, то напротив номера соответствующей области проставлялась отметка «Зависимость сильно нарушена». Необходимо отметить, что максимальное значение $U_{\text{ДАН}}$ в границах рассматриваемой области не всегда означает, что в данном отсчёте будет наблюдаться максимальный SNR . Значение СКО шума может отличаться в 2-3 раза для разных отсчётов, даже в областях, на которых не происходит резких спадов или нарастания сигнала (как в первой и третьей областях). В свою очередь в первой и третьей областях наблюдаются резкие спады и нарастания сигнала (рисунок 2.15 и рисунок Д.1к приложения Д) в этих областях $U_{\text{ДАН}}$ достигают весьма больших значений, но вместе с тем наблюдается и значительная дисперсия шума. Это явление можно объяснить ограниченностью частоты дискретизации используемой измерительной установки. Оно весьма ясно иллюстрируется формами сигнала, изображёнными на рисунке 2.9 – даже при обработке одних и тех же данных на одной и той же операции некоторые снятые формы сигнала могут весьма сильно искажаться, причём не составляет труда автоматически отбраковывать такие формы сигнала. Для случаев, когда максимальные значения $U_{\text{ДАН}}$ не соответствуют максимальным значениям SNR в таблице Е.1 добавлены строки имеющие серый фон, в них указаны отсчёты в той же области соответствующего такта, в которых SNR является максимальным. На основе результатов измерений, приведённых в таблице Е.1 приложения Е, а также описанного выше, можно сделать следующе основные выводы, об особенностях энергопотребления анализируемого чипа при выполнении операции записи байта данных в чистую ячейку ОЗУ:

- Энергопотребление микроконтроллера имеет весьма существенную зависимость от обрабатываемых данных. При этом выполнением самой операции обусловлено увеличение амплитуды скачков напряжения на снятых формах сигнала до 17.2 мВ (почти на 42%, или 8.6 мВт), по сравнению с выполнением чипом пустой команды. Кроме того, энергопотребление определяется весом Хэмминга обрабатываемых данных, причём эта «добавка» может достигать 7.2 мВ (17.6%, или 3.6 мВт) для каждого увеличения веса Хэмминга на единицу.
- Зависимость энергопотребления от обрабатываемых данных, затрагивает минимум четыре последующих такта. (Далее это явление будет называться *остаточной корреляцией*.) Энергопотребление на этих тактах также будет зависеть от веса Хэмминга обрабатываемой

комбинации. Эта зависимость спадает от такта к такту, причём можно считать, что через пять тактов она сходит на нет.

- На каждом из шести тактов, наибольший SNR наблюдается во второй и четвёртой областях, к тому же не в пиковых значениях, а правее (областях спада сигнала).
- Характер зависимости может меняться несколько раз от прямой на обратную даже в рамках одной области такта.

2.4.2 Зависимость энергопотребления от обрабатываемых данных при выполнении операции записи в предварительно заполненную ячейку ОЗУ

Описанный эксперимент был повторён для операции записи данных в ячейку ОЗУ, содержащую случайную комбинацию (эта операция выполняется микроконтроллером также за два такта), однако в данном случае рассчитывался не вес Хэмминга сохраняемого байта, а расстояние Хэмминга между записываемыми в ОЗУ данными и теми данными, которые они перезаписывают. Девять усреднённых форм сигнала представлены на рисунке 2.16.

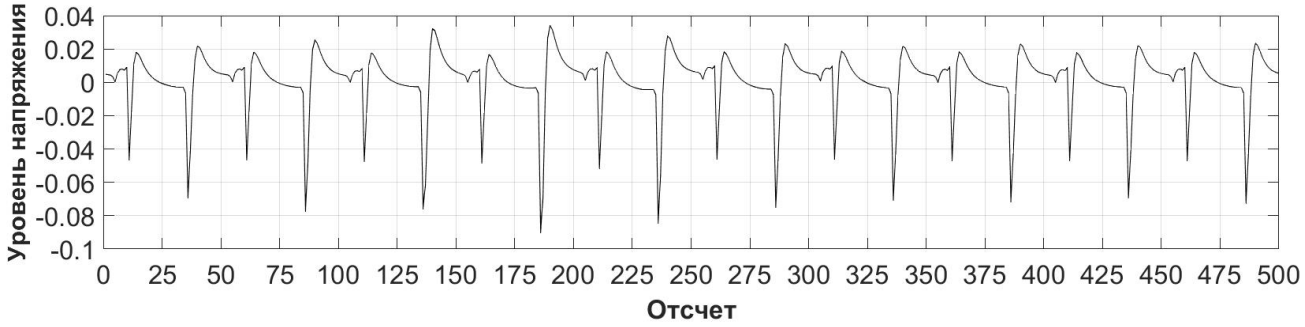


Рисунок 2.16 – Девять усреднённых форм сигнала для различных расстояний Хэмминга при выполнении операции записи в предварительно заполненную ячейку ОЗУ

Увеличение масштаба рисунка 2.16 показывает, что лишь на двух первых тактах существует зависимость энергопотребления микроконтроллера от расстояния Хэмминга между записываемыми и перезаписываемыми в ОЗУ данными (увеличенные области формы сигнала представлены на рисунках Д.2 приложения Д). При этом зависимость является весьма слабой и для некоторых расстояний Хэмминга нарушается. Поэтому были построены (рисунок 2.17, рисунки Д.3 приложения Д) девять усреднённых форм сигнала без учёта наличия в ячейке ОЗУ каких-либо данных (с использованием модели веса Хэмминга).

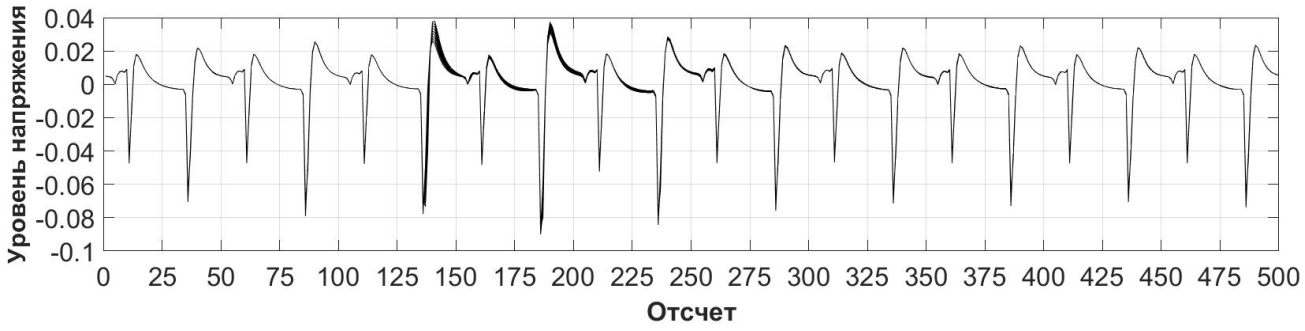


Рисунок 2.17 – Девять усреднённых форм сигнала для различных весов Хэмминга при выполнении операции записи в предварительно заполненную ячейку ОЗУ

Гистограммы, а также, характеристики форм сигнала при использовании модели веса Хэмминга практически полностью повторяют описанные ранее, для операции записи в чистую ячейку ОЗУ (в том числе и явление остаточной корреляции на четырёх последующих тактах). Аналогично операции записи в чистую ячейку ОЗУ, для операции записи в предварительно заполненную ячейку, каждое увеличение на единицу веса Хэмминга обрабатываемых комбинаций, приводит к увеличению среднего значения 140-го отсчёта примерно на 2.29 мВ, а 190-го на 1.3 мВ. СКО шума также незначительно увеличивается с увеличением веса Хэмминга, и в среднем оно составляет 2.44 мВ и 1.76 мВ соответственно. SNR для 140-го отсчёта составит 2.26, а для 190-го – 1.34, значение $E\left(U_{\text{ОП}}\right)=55.3-41=14.3$ мВ. Значения $U_{\text{ДАН}}$, СКО и SNR для участков шести тактов, на которых существует зависимость энергопотребления от веса Хэмминга обрабатываемых данных приведены в таблице Е.2 приложения Е. Из таблицы видно, что отсутствуют сколько-нибудь значительные отличия в энергопотреблении по сравнению со случаем записи данных в чистую ячейку ОЗУ.

Из полученных данных можно сделать вывод, что энергопотребление микроконтроллера практически не зависит от количества изменившихся бит в предварительно заполненной ячейке ОЗУ. Отсюда логично предположить, что ячейки ОЗУ перед записью новых данных затираются или операция записи в ОЗУ, требует пренебрежимо малых затрат энергии. В этом случае, полученные при использовании модели веса Хэмминга всплески, обусловлены только энергией, затрачиваемой на чтение данных из регистра в АЛУ и зарядку линий шины данных для их последующей пересылки в ОЗУ. В любом случае, независимо от того, осуществляется ли запись в чистую ячейку ОЗУ или предварительно заполненную, для определения записываемой комбинации, по снятым формам сигнала, необходимо использовать модель веса Хэмминга.

2.4.3 Зависимость энергопотребления от обрабатываемых данных при выполнении операции чтения из ячейки ОЗУ в чистую ячейку рабочего регистра

Описанный эксперимент был повторён для операции чтения данных из ячейки ОЗУ в чистую ячейку рабочего регистра (эта операция выполняется микроконтроллером также за два такта). Девять усреднённых форм сигнала представлены на рисунке 2.18.

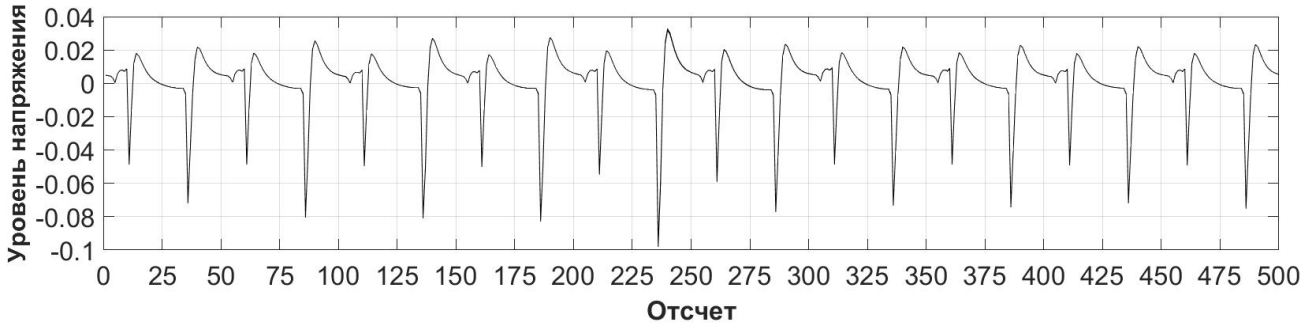


Рисунок 2.18 – Девять усреднённых форм сигнала для различных весов Хэмминга при выполнении операции чтения из ячейки ОЗУ в чистую ячейку рабочего регистра

Увеличение масштаба соответствующих участков показывает (рисунки Д.4 приложения Д), что зависимость между энергопотреблением микроконтроллера и весом Хэмминга, считываемых из ОЗУ данных, существует лишь на третьем такте (с 236-го по 284-й отсчёты), в частности, наибольшую зависимость имеет положительный всплеск во второй области, всплеск в четвёртой области имеет менее сильную, обратную зависимость (264-й отсчёт), которая сохраняется до конца такта. Область в середине такта, а также отрицательные всплески (первая и третья области) зависимости не имеют. На последующих тактах зависимость сильно нарушается. При этом значение $E\left(U_{\text{ОП}}\right)$ составляет порядка 16.1 мВ.

Значения $U_{\text{ДАН}}$, СКО и SNR для участков третьего такта приведены в таблице Е.3 приложения Е. Оттуда видно, что по сравнению с операцией записи в ОЗУ, разница математических ожиданий рассматриваемых отсчётов для каждого веса Хэмминга, уменьшается примерно на порядок, при приблизительно таком же уровне шума, что значительно влияет на соотношение сигнал/шум. Отсюда понятно, что окажется значительно труднее по форме сигнала (даже усреднённой) однозначно определить вес Хэмминга комбинации, считываемой из ОЗУ по сравнению с записываемой в ОЗУ.

2.4.4 Зависимость энергопотребления от обрабатываемых данных при выполнении операции чтения из ячейки ОЗУ в предварительно заполненную ячейку рабочего регистра

В том случае, если повторить эксперимент, но перед копированием данных, в регистр назначения записывать случайную комбинацию, таким образом, чтобы копируемые из ОЗУ данные, перезаписывали данные, находящиеся в регистре назначения – эта операция также будет выполняться микроконтроллером за два такта. В этом случае необходимо использовать модель расстояния Хэмминга, и усреднять формы сигнала в зависимости от количества изменившихся в регистре назначения бит. Девять усреднённых таким образом форм сигнала представлены на рисунке 2.19.

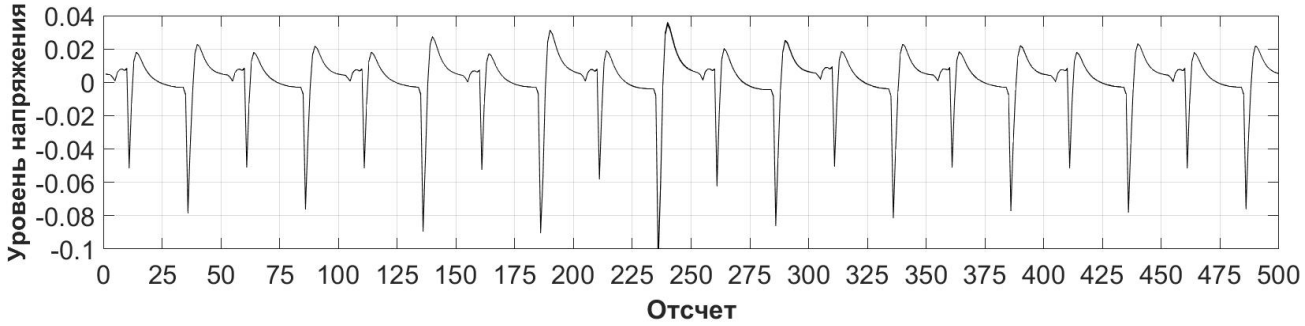


Рисунок 2.19 – Девять усреднённых форм сигнала для различных расстояний Хэмминга при выполнении операции чтения из ячейки ОЗУ в предварительно заполненную ячейку рабочего регистра

Из рисунка 2.19 видно, что наибольшие всплески энергопотребления возникают на четырёх тактах с 136-го по 336-й отсчёты.

Также были построены девять усреднённых форм сигнала без учёта наличия в регистре назначения каких-либо данных (с использованием модели веса Хэмминга; рисунок 2.20).

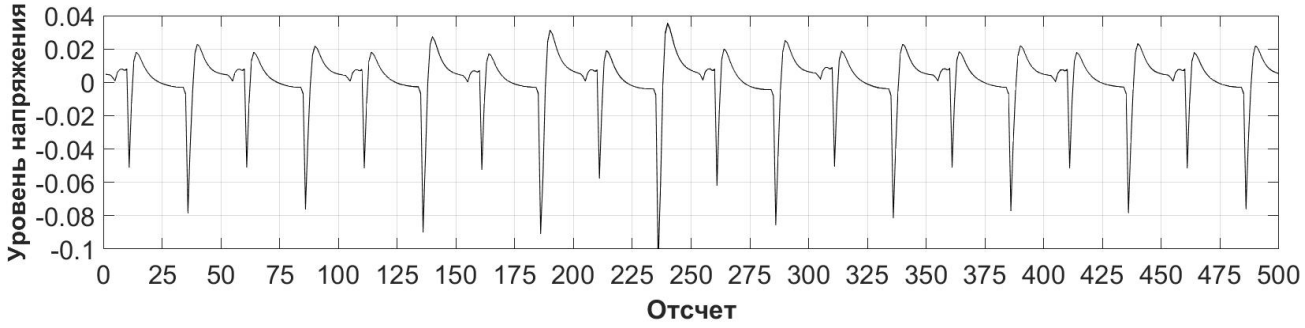


Рисунок 2.20 – Девять усреднённых форм сигнала для различных весов Хэмминга при выполнении операции чтения из ячейки ОЗУ в предварительно заполненную ячейку рабочего регистра

Увеличение масштаба соответствующих участков показывает (рисунки Д.5 приложения Д), что зависимость от веса Хэмминга, существует в четвёртой области второго такта (216-220-й отсчёты), а также в первом положительном всплеске третьего такта (вторая область, 240-й отсчёт). Зависимость от расстояния Хэмминга наблюдается на двух тактах: третьем и четвёртом. При этом значение $E(U_{\text{ОП}})$ возрастёт до 21.8 мВ.

Анализируя полученные значения $U_{\text{ДАН}}$, СКО и SNR для моделей веса Хэмминга (таблица 2.1) и расстояния Хэмминга (таблица 2.2) можно сделать вывод, что для случая, чтения из ОЗУ в предварительно заполненный регистр, лучше использовать модель расстояния Хэмминга, так как значения $U_{\text{ДАН}}$ и SNR в данном случае будут значительно больше чем при использовании модели веса Хэмминга.

Таблица 2.1 – Наилучшие для злоумышленника значения $U_{\text{ДАН}}$, СКО и SNR в областях форм сигнала, на которых существует зависимость энергопотребления от веса Хэмминга обрабатываемых данных, при выполнении операции чтения из ячейки ОЗУ в предварительно заполненную ячейку рабочего регистра

Такт	Область	Зависимость	Отсчёт	$U_{\text{ДАН}}$, мВ	СКО, мВ	SNR	Особые отметки
2	4	Прямая	216	0.063	2.70	0.0010	
3	2	Прямая	240	0.147	3.58	0.0029	

Таблица 2.2 – Наилучшие для злоумышленника значения $U_{\text{ДАН}}$, СКО и SNR в областях форм сигнала, на которых существует зависимость энергопотребления от расстояния Хэмминга обрабатываемых данных, при выполнении операции чтения из ячейки ОЗУ в предварительно заполненную ячейку рабочего регистра

Такт	Область	Зависимость	Отсчёт	$U_{\text{ДАН}}$, мВ	СКО, мВ	SNR	Особые отметки
3	1	Обратная	237	0.719	54.64	0.00036	*
	1	Прямая	239	0.265	10.21	0.00302	*
	2	Прямая	240	0.300	3.58	0.01893	
	3	Зависимость сильно нарушена					
	4	Обратная	272	0.046	1.95	0.00235	*
	4	Обратная	273	0.04	1.92	0.00246	*
4	2	Прямая	290	0.124	3.03	0.00445	
	2	Прямая	291	0.126	3.24	0.00260	

Сравнивая всплески энергопотребления микроконтроллера при чтении байта данных из ОЗУ в чистый и предварительно заполненный регистр, можно сделать вывод, что сложность определения обрабатываемой комбинации для этих двух случаев окажется приблизительно

одинаковой – операция чтения из ОЗУ приводит к значительно меньшей утечке чем запись в ОЗУ.

2.4.5 Зависимость энергопотребления от обрабатываемых данных при выполнении операции чтения из ячейки ПЗУ в чистую ячейку рабочего регистра

Описанный эксперимент был повторён для операции чтения данных из ячейки ПЗУ в чистую ячейку регистра (данная операция выполняется микроконтроллером также за два такта). Девять усреднённых форм сигнала представлены на рисунке 2.21.

Можно предположить, что выполнением операции чтения из ПЗУ обусловлены всплески напряжения на четырёх тактах – с 136-го по 336-й отсчёты.

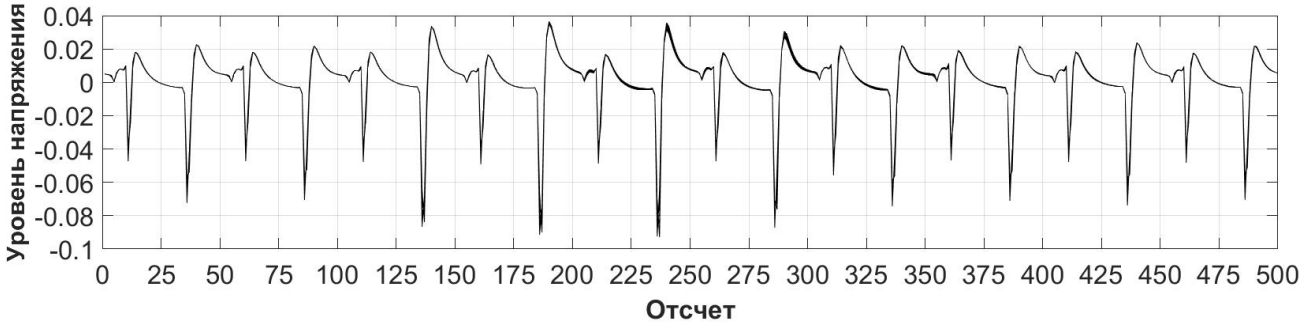


Рисунок 2.21 – Девять усреднённых форм сигнала для различных весов Хэмминга при выполнении операции чтения их ячейки ПЗУ в чистую ячейку рабочего регистра

На практике зависимость существует на всех тактах, представленных на рисунке 2.21 (в том числе на трёх тактах, характеризующих энергопотребление микроконтроллера до момента выполнения команды чтения данных из ПЗУ), объяснений причинам этого явления найти не удалось. Наибольшая зависимость проявляется на трёх тактах со 190-го по 336-й отсчёты, значения $U_{\text{ДАН}}$, СКО и SNR представлены в таблице Е.4 приложения Е. При этом в пиковых значениях положительных всплесков зависимость может весьма сильно нарушаться, однако в областях спада и нарастания сигнала она как-правило весьма стабильна (рисунки Д.6 приложения Д).

Наибольший SNR (равный 0.42) наблюдается на 290-м отсчёте, при этом значение $U_{\text{ДАН}}$ достигает значения 0.77 мВ, при СКО шума равным 2.48. При сравнении операция чтения из ПЗУ, с операцией записи в ОЗУ можно сделать однозначный вывод, что первая приводит к

меньшей «утечке» информации об обрабатываемых данных. Однако, по сравнению с операцией чтения из ОЗУ, «утечка» оказывается заметно больше.

Сравнивая рисунок 2.21 с аналогичными (для операций чтения из ОЗУ и записи в ОЗУ) можно сделать вывод, что операция чтения из ПЗУ в целом приводит к б**о**льшим всплескам энергопотребления так как можно заметить, что на четырёх тактах всплески будут превышать 0.03 уровня напряжения (что соответствует 54 мВ). Причём, на максимальном из них, на 190-м отсчёте $U_{ОП} = 62.9 - 41 = 21.9$ мВ.

2.4.6 Зависимость энергопотребления от обрабатываемых данных при выполнении операции чтения из ячейки ПЗУ в предварительно заполненную ячейку рабочего регистра

Описанный эксперимент был повторён для операции чтения данных из ячейки ПЗУ в предварительно заполненную ячейку регистра. Девять усреднённых форм сигнала представлены на рисунке 2.23.

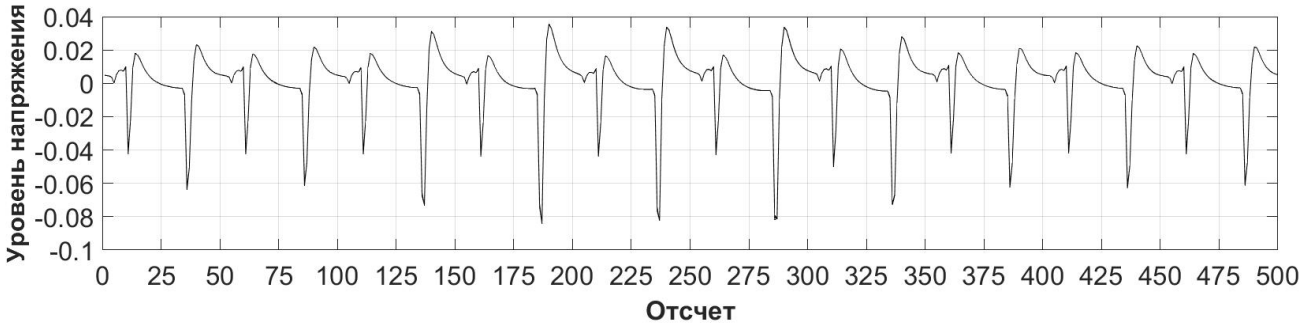


Рисунок 2.22 – Девять усреднённых форм сигнала для различных расстояний Хэмминга при выполнении операции чтения из ячейки ПЗУ в предварительно заполненную ячейку рабочего регистра

Увеличение масштаба (рисунок 2.23) показывает, что, нет никакой зависимости между энергопотреблением микроконтроллера и расстоянием Хэмминга между записываемыми и перезаписываемыми данными в регистре назначения. Поэтому также были построены девять усреднённых форм сигнала без учёта начального заполнения ячейки регистра (с использованием модели веса Хэмминга, рисунок 2.24).

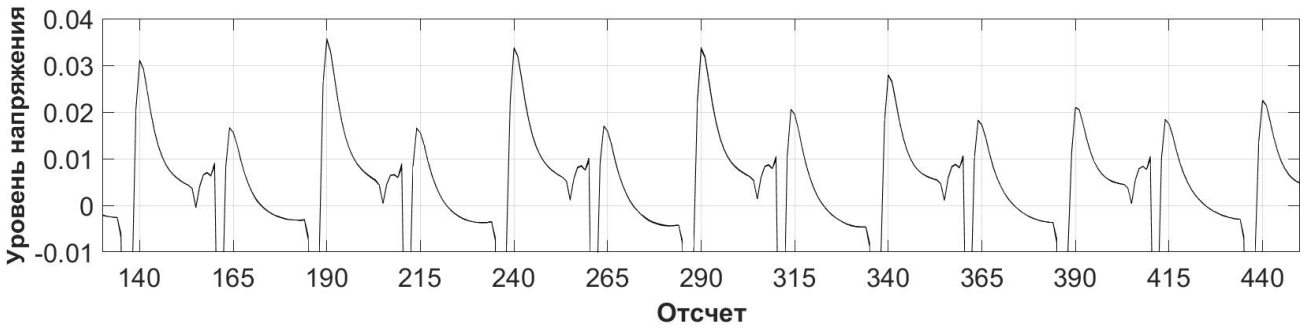


Рисунок 2.23 – Девять усреднённых форм сигнала для различных расстояний Хэмминга при выполнении операции чтения из ячейки ПЗУ в предварительно заполненную ячейку рабочего регистра (увеличенная область)

Увеличение соответствующих областей рисунка 2.24, а также анализ значений $U_{\text{ДАН}}$, СКО и SNR (рисунки Д.7 приложения Д и таблица Е.5 приложения Е) позволяют сделать вывод, о том, что зависимость энергопотребления от обрабатываемых данных оказывается слабее чем для случая копирования из ПЗУ в чистую ячейку регистра. Очень серьёзное падение SNR возникает на 290-м отсчёте, за счёт 50%-го увеличения СКО шума, при этом $U_{\text{ДАН}}$ находится на примерно том, же уровне. Значительные всплески превышающие 0.03 уровня напряжения (54 мВ) также возникают на четырёх тактах, при этом значение $U_{\text{ОП}}$, определяемая по максимальному всплеску (также происходящему на 290-м отсчёте) снижается незначительно (в пределах погрешности), составляя 21.6 мВ.

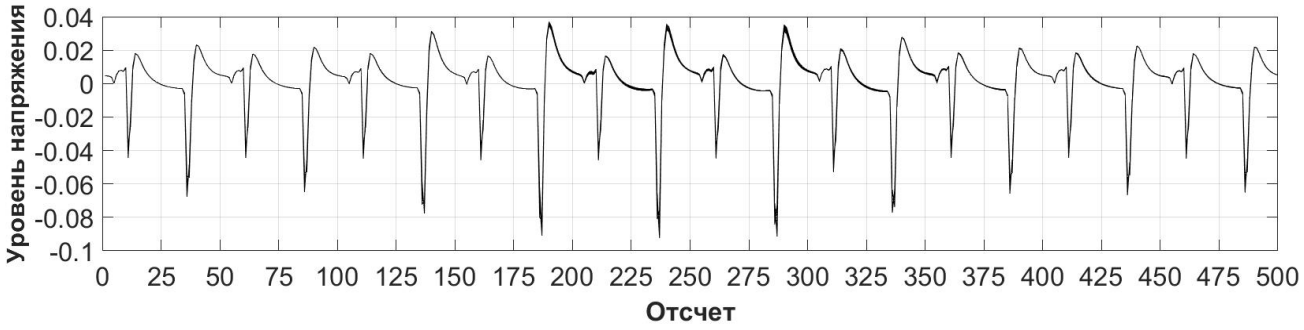


Рисунок 2.24 – Девять усреднённых форм сигнала для различных весов Хэмминга при выполнении операции чтения из ячейки ПЗУ в предварительно заполненную ячейку рабочего регистра

Из полученных результатов можно сделать вывод, что независимо от того, осуществляется ли запись из ПЗУ в чистую ячейку регистра, или предварительно заполненную – в обоих случаях, для определения считываемой из ПЗУ комбинации, по снятым формам сигнала, необходимо использовать модель веса Хэмминга.

2.4.7 Зависимость энергопотребления от обрабатываемых данных при выполнении операции копирования данных между рабочими регистрами

Описанный эксперимент был повторён для операции копирования данных из одного рабочего регистра в другой, предварительно очищенный регистр (данная операция выполняется микроконтроллером за один такт). Девять усреднённых форм сигнала представлены на рисунке 2.25.

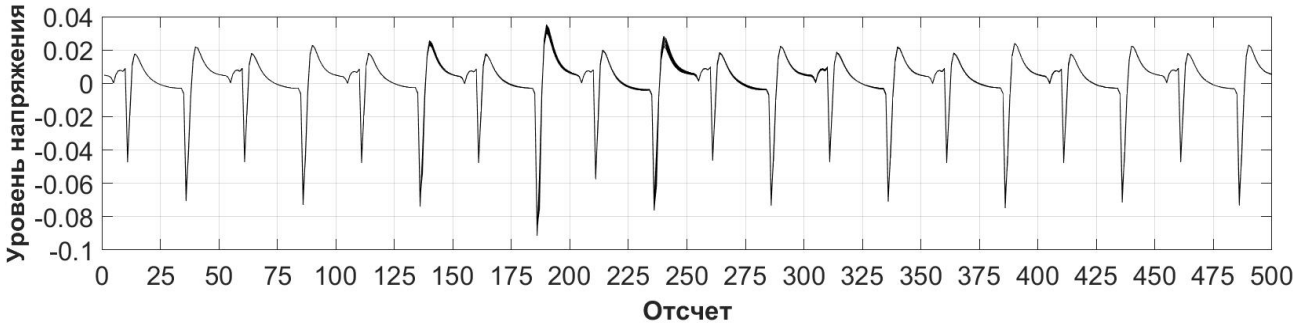


Рисунок 2.25 – Девять усреднённых форм сигнала для различных весов Хэмминга при выполнении операции копирования данных между рабочими регистрами

Увеличение масштаба соответствующих участков (рисунки Д.8 приложения Д и таблица Е.6 приложения Е) показывает, что выполнение самой операции копирования данных из регистра в регистр, приводит к весьма небольшому увеличению энергопотребления микроконтроллера до $U_{\text{ОП}} = 54.7 - 41 = 13.7$ мВ. При этом в отличие от рассмотренных ранее операций, существует лишь один скачок напряжения, превышающий 0.03 уровня напряжения. Однако существует весьма сильная зависимость от обрабатываемых данных как на такте, на котором выполняется операция, так и на пяти последующих тактах. При этом на втором и третьем тактах, соответственно в 190-м и 240-м отсчётах наблюдаются весьма большие значения SNR (около единицы), при этом значения $U_{\text{ДАН}}$ составят соответственно: 0.97 и 1.31 мВ.

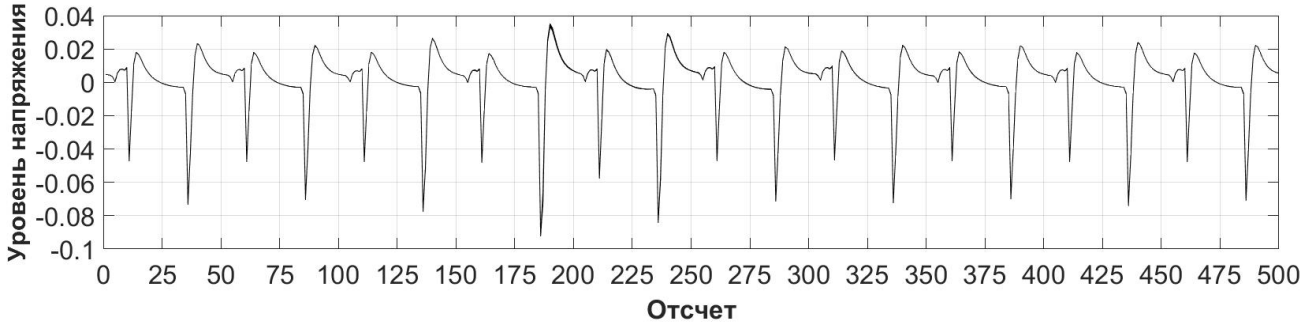
2.4.8 Зависимость энергопотребления от обрабатываемых данных при выполнении операции копирования данных из одного рабочего регистра в другой, предварительно заполненный регистр

В том случае, если повторить эксперимент, но перед копированием данных в регистр назначения записывать случайную комбинацию, таким образом, чтобы копируемые данные

перезаписывали данные находящиеся в регистре назначения – данная операция также будет выполняться микроконтроллером за один такт. В этом случае необходимо использовать модель расстояния Хэмминга, и усреднять формы сигнала в зависимости от количества изменившихся бит в регистре назначения. Девять усреднённых таким образом форм сигнала представлены на рисунке 2.26а. Девять усреднённых форм сигнала без учёта предварительного заполнения регистра (с использованием модели веса Хэмминга) приведены на рисунке 2.26б.

Увеличение масштаба соответствующих графиков, а также анализ данных по значениям $U_{\text{ДАН}}$, СКО и SNR для обоих случаев (рисунки Д.9 приложения Д и таблиц Е.7, Е.8 приложения Е) позволяет сделать вывод о значительном снижении утечки по сравнению со случаем копирования данных в чистый рабочий регистр.

а)



б)

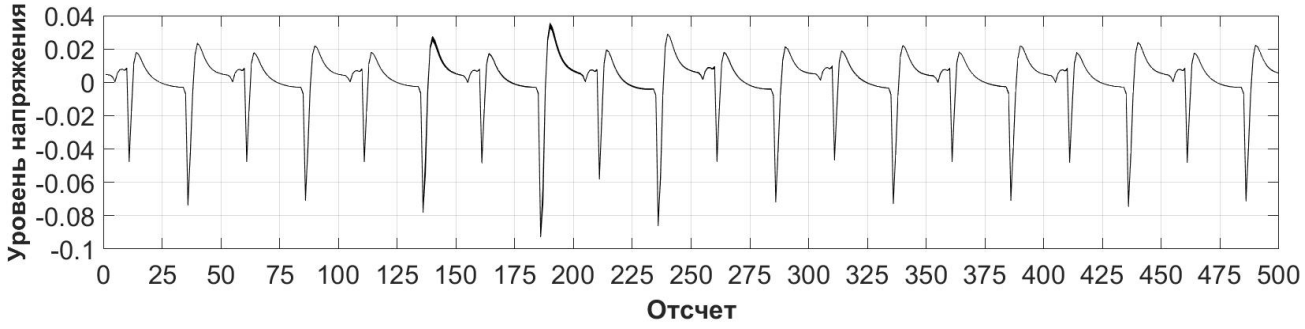


Рисунок 2.26 – Девять усреднённых форм сигнала для различных: а) расстояний Хэмминга; б) весов Хэмминга, при выполнении операции копирования данных из одного рабочего регистра в другой, предварительно заполненный регистр

При использовании модели расстояния Хэмминга зависимость наблюдается на четырёх тактах – от второго по пятый, при этом значения SNR не превысят 0.03, при $U_{\text{ДАН}} = 0.38$ мВ. При использовании модели веса Хэмминга (без учёта предварительного заполнения регистра назначения) зависимость будет существовать на 1-5 тактах, значения SNR будут принимать значительно б**о**льшие значения – до 0.18, при $U_{\text{ДАН}} = 0.52$ мВ, причём величина $U_{\text{ОП}}$ оказывается

равной 18.7 мВ . В целом можно заключить, что по сравнению со случаем копирования данных в чистую ячейку рабочего регистра утечка представляется весьма слабой.

Из полученных результатов можно сделать вывод, что независимо от того, осуществляется ли копирование данных в чистую ячейку регистра, или предварительно заполненную – в обоих случаях, лучше использовать модель веса Хэмминга.

2.4.9 Зависимость энергопотребления от обрабатываемых данных при выполнении операции XOR между комбинациями из двух рабочих регистров

Описанный эксперимент был повторён для операции сложения по mod2 (XOR) содержимого двух рабочих регистров. Данная операция выполняется микроконтроллером за один такт, результат помещается в один из регистров-слагаемых, перезаписывая его содержимое – соответствующее слагаемое.

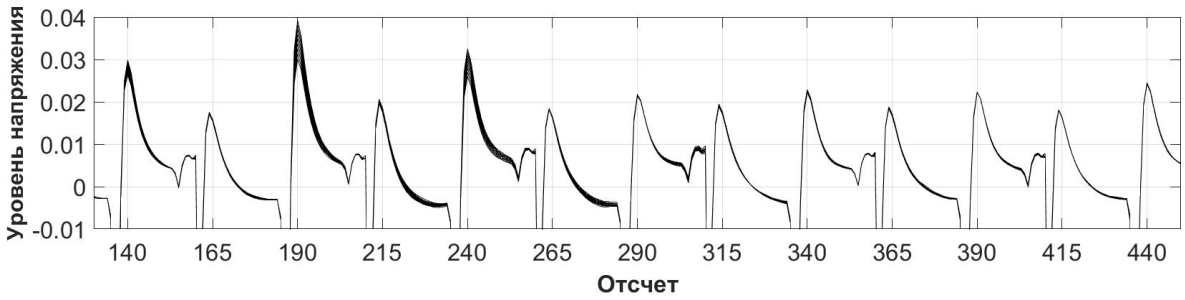
Выполнение операции XOR на микроконтроллере осуществляется за три шага:

- слагаемое A из первого регистра и слагаемое B из второго регистра поступают по шине данных в АЛУ, на котором непосредственно осуществляется операция сложения по mod2;
- результат сложения по шине данных отправляется в первый регистр (содержащий слагаемое A);
- результат перезаписывает содержимое регистра назначения.

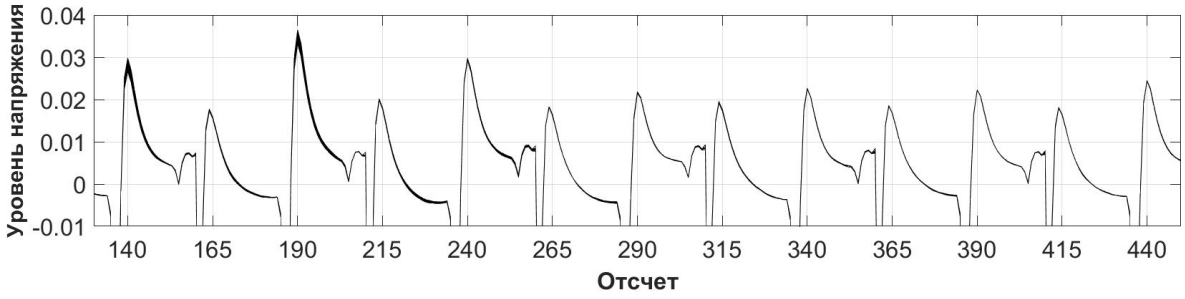
В данном случае представляет интерес возможность определения по формам сигнала веса Хэмминга обеих комбинаций участвующих в операции XOR, а также результата этой операции. Поэтому были построены три группы усреднённых форм сигнала:

- усреднение которых производилось в зависимости от веса Хэмминга комбинации из первого регистра (рисунок 2.27а);
- усреднение которых производилось в зависимости от веса Хэмминга комбинации из второго регистра (рисунок 2.27б);
- усреднение которых производилось в зависимости от веса Хэмминга суммы по mod2 комбинаций этих регистров (рисунок 2.27в);

а)



б)



в)

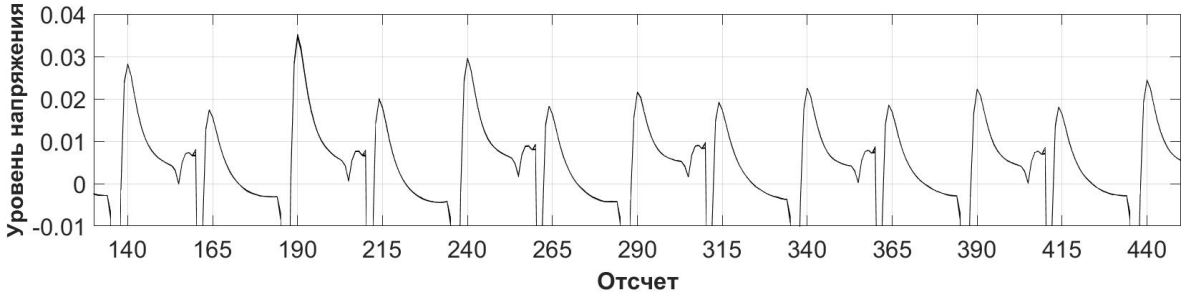


Рисунок 2.27 – Девять усреднённых форм сигнала, полученных при выполнении операции XOR комбинаций из двух рабочих регистров, и характеризующих зависимость энергопотребления от: а) веса Хэмминга комбинации из первого регистра; б) веса Хэмминга комбинации из второго регистра; в) веса Хэмминга суммы по mod2 комбинаций двух регистров

Анализируя формы сигнала для рассматриваемых случаев, а также значения $U_{\text{ДАН}}$, СКО и SNR (таблицы Е.9 - Е.11 приложения Е) можно видеть, что:

- Существует возможность определения по формам сигнала веса Хэмминга обеих комбинаций участвующих в операции XOR, при этом утечка от обработки разных слагаемых будет сильно отличаться. Для слагаемого из первого регистра зависимость существует на шести тактах, а наибольший SNR возникает на 190-м и 240-м отсчётах, составляя 1.12 и 0.72, при $U_{\text{ДАН}}$ равным 1.81 мВ и 1.29 мВ, а СКО шума равным 2.71 мВ и 2.39 мВ соответственно. В то время как для второго слагаемого зависимость существует на

пяти тактах, а наибольший SNR возникает на 140-м и 190-м отсчётах, составляя 0.17 и 0.07, при $U_{\text{ДАН}}$ равным 0.63 мВ и 0.65 мВ, а СКО шума равным 2.50 мВ и 3.83 мВ соответственно.

- Обработка результата сложения по mod2 приводит к крайне малой утечке и только на втором такте, при этом наибольший SNR возникает на 190-м отсчёте, составляя 0.004 при $U_{\text{ДАН}} = 0.18$ мВ, а СКО шума равным 3.95 мВ.
- Расстояние Хэмминга между результатом сложения по mod2 и перезаписываемой комбинацией будет представлять собой вес Хэмминга комбинации, содержащейся во втором регистре: $(A \oplus B) \oplus A = B$. Поэтому сложность атаки с использованием модели расстояния Хэмминга между результатом сложения и перезаписываемым слагаемым будет эквивалентна сложности реализации атаки с использованием слагаемого из второго регистра.
- Величина $U_{\text{ОП}}$ оказывается весьма значительной (20.7 мВ).

2.5 Оценка эффективности реализации DPA в отношении различных операций, с использованием экспериментальных данных

В этом разделе, описанные ранее теоретические основы DPA, будут подтверждены экспериментально. В том числе будет показано какие операции алгоритма, выполняемого анализируемым чипом, окажутся наиболее уязвимыми при реализации DPA. Также для каждой операции будут приведены данные о количестве форм сигнала, требующемся для успешной реализации атаки.

2.5.1 Описание методологии измерений

В разделе 2.4 уже было показано, что разные операции, выполняемые анализируемым микроконтроллером, приводят к различной утечке информации по цепи электропитания, поэтому, при исследовании практической реализации DPA, логично определить какие операции наиболее уязвимы к разностному анализу мощности. При реализации атаки на полноценный защитный алгоритм, результаты этих исследований помогут выбрать такое промежуточное

значение алгоритма $a_l(i) = f\left(m_l(i),\ k_l\right)$, которое позволит наиболее эффективно реализовать атаку DPA, то есть с использованием наименьшего количества форм сигнала.

Для этого на микроконтроллере были реализованы две простейшие операции, часто применяющиеся в алгоритмах защитных преобразований это сложение по mod2 и преобразование на S-box. В качестве последнего были взяты восьмой и седьмой S-box ГОСТ 28147-89 (использовались таблицы преобразований S-box описанные в книге [9], наиболее актуальными на сегодняшний день являются описанные в стандарте ГОСТ Р 34.12-2015 таблицы преобразования шифров «Магма» или «Кузнечик», но на момент проведения исследований этот стандарт ещё не был опубликован). Необходимо напомнить, что S-box ГОСТ 28147-89 совершают табличное преобразование четырёх бит поступающих на их вход в четырёх битовый выход. Напомним, что анализируемый микроконтроллер имеет восьми разрядную архитектуру, а, следовательно, при выполнении любых команд, он оперирует восемью разрядными комбинациями. Поэтому при реализации этого шифра на микроконтроллере нет оснований отдельными командами совершать преобразования на седьмом и восьмом S-Box – наиболее эффективно выполнять преобразование на двух S-box за один подход (оперируя не четырьмя, а восемью битами), подробное описание реализации такого преобразования представлено в приложении Ж.

Опишем реализацию выполняемого анализируемым чипом алгоритма:

1. На этапе программирования микроконтроллера в его Flash память последовательно, байт за байтом, записывается таблица преобразований седьмого и восьмого S-box ГОСТ, а также восьми битовый ключ $k_{7-8} = 1111\,1111$.
2. При выполнении программы ключ $k_{7-8} = 1111\,1111$ считывается из Flash памяти в 18-й рабочий регистр.
3. ПК отсылает на микроконтроллер случайную двоичную восьми битовую комбинацию $m_{7-8}(i)$, являющуюся условным сообщением, данное сообщение сохраняется в ОЗУ, после чего перемещается в 17-й рабочий регистр.
4. Производится сложение по модулю 2 содержимого 17-го и 18-го регистров, с сохранением результата в 17-й регистр (то есть результат $S_{7-8}^{IN}(i) = m_{7-8}(i) \oplus k_{7-8}$ перезаписывает комбинацию $m_{7-8}(i)$). Подчеркнём, что так как от ПК поступают случайные восьми битовые комбинации $m_{7-8}(i)$, то результат сложения по mod2 будет принимать любое из 256-ти возможных значений (от 0 до 255-ти) с равной вероятностью.

5. В 31-32 регистры загружается адрес ячейки Flash памяти, в которой содержится выход 7-8 S-box, соответствующий входу «0000 0000» (по этому адресу находится первый элемент таблицы преобразований).
6. Комбинация $S_{7-8}^{IN}(i)$, складывается с содержимым 31-32 регистров, результат помещается в 31-32 регистры.
7. По адресу Flash памяти, находящемуся в 31-32 регистрах происходит чтение байта (так называемое чтение со смещением, которое подробно описано в приложении Ж), результат сохраняется в 19-й регистр. Результат данной операции – байт $S_{7-8}^{OUT}(i)$, представляющий собой комбинацию $S_{7-8}^{IN}(i)$, преобразованную на седьмом и восьмом S-box ГОСТ.

   Все описанные выше операции выполняются до того, как микроконтроллер отошлёт на АЦП триггерный сигнал, и скачки энергопотребления в процессе их выполнения не будут отображены на снятых формах сигнала.

8. На порт ввода-вывода микроконтроллера выставляется логическая «1», незамедлительно после этого, она сбрасывается на логический «0». Таким образом на триггерный вход АЦП приходит короткий положительный импульс, запускающий запись формы сигнала, характеризующей энергопотребление микроконтроллера. После этого делается пауза на 25 тактов микроконтроллера (она необходима для того, чтобы избежать влияния скачка энергопотребления, вызванного отправкой импульса триггерного сигнала на исследуемую на девятом шаге операцию). (Длина снимаемой формы сигнала – 4 096 отсчётов, при частоте дискретизации в 100 млн. отсчётов в секунду, при этом, тактовая частота микроконтроллера составляет 2 МГц (2 млн. тактов в секунду).)
9. На последнем шаге выполняются манипуляции с комбинацией $S_{7-8}^{OUT}(i)$, в частности операции, особенности энергопотребления при выполнении которых описывались в разделе 2.4. Следовательно, будет исследована эффективность реализации DPA в случае выбора в качестве рассчитываемого промежуточного значения $a_l(i) = f(m_l(i), k_l)$ результата выполнения каждой рассматриваемой операции.

2.5.2 Реализация DPA для операции записи данных в ячейку ОЗУ

Вначале была исследована эффективность реализации DPA, для случая выбора в качестве промежуточного значения комбинации, пересылаемой из рабочего регистра в ячейку ОЗУ. В подразделе 2.4.1, приводились особенности энергопотребления анализируемого микроконтроллера при выполнении этой операции – она выполняется за 2 такта (100 отсчётов на форме сигнала), и приводит к значительной утечке информации по цепи электропитания. С использованием измерительной установки, были реализованы 1-9 шаги, приведённого в подразделе 2.5.1 алгоритма. При этом 3-9 шаги повторялись 51 200 раз, каждый из которых, байт сообщения $m_{7-8}(i)$, $i = 1,....,51\,200$ генерировался случайным образом. В итоге было снято 51 200 форм сигнала, каждая из них сохранялась в отдельном файле, в названии которого содержалось значение соответствующего $m_{7-8}(i)$.

Полученные формы сигнала в точности повторяли представленные в подразделе 2.4.1. При помощи программной среды MatLab, все формы сигнала загружались из соответствующих файлов в матрицу $\mathbf{A}$ размерностью 51 200x4 096. Каждая из 51 200 строк которой содержала 4 096 отсчётов соответствующей формы сигнала.

Помимо содержимого файлов, из заголовка каждого из них, в вектор $\vec{m}_{7-8} = \left(m_{7-8}(1),\ ...,\ m_{7-8}(51\,200)\right)$, считывался байт сообщения, на котором соответствующая форма сигнала была получена.

Затем выполнялся второй шаг алгоритма DPA, на котором составлялся вектор вариантов перебираемого ключа $\vec{k}'_{7-8} = \left(k'_{7-8}(1),\ ...,\ k'_{7-8}(256)\right) = (0,\ 1,\ ...,\ 255)$. Для каждого из них, используя элементы вектора $\vec{m}_{7-8}$, в среде MatLab рассчитывался выход 7-8 S-box: $S_{7-8}^{OUT}\left(i,\ k'_{7-8}(d)\right) = S_{7-8}\left(m_{7-8}(i) \oplus k'_{7-8}(d)\right)$, где $S_{7-8}(\cdot)$ – преобразование на 7-8 S-box. Таким образом получалась матрица $\mathbf{B}$ размерностью 51 200x256.

Далее каждому элементу $S_{7-8}^{OUT}\left(i,\ k'_{7-8}(d)\right)$ матрицы $\mathbf{B}$ сопоставлялся коэффициент, характеризующий величину энергопотребления микроконтроллера при обработке такой комбинации. При этом использовалась модель веса Хэмминга. Из значений полученных коэффициентов создавалась матрица $\mathbf{C}$ той же размерности, что и матрица $\mathbf{B}$, в которой каждому значению $S_{7-8}^{OUT}\left(i,\ k'_{7-8}(d)\right)$ был сопоставлен его вес Хэмминга.

Затем, отдельно для каждого столбца матрицы $\mathbf{C}$ рассчитывался коэффициент корреляции с каждым столбцом матрицы $\mathbf{A}$. (То есть корреляцию первого столбца матрицы $\mathbf{C}$, содержащего рассчитанные коэффициенты, характеризующие величину энергопотребления

микроконтроллера, с первым, затем вторым, и так далее до 4 096-го столбцами матрицы $\mathbf{A}$, содержащих отсчёты, фактического энергопотребления микроконтроллера. Затем тоже самое повторялось для последующих 255-ти столбцов матрицы $\mathbf{C}$.) Таким образом для каждого из 256-ти вариантов перебираемого ключа $k'_{7-8}(d)$, вычисляется вектор $\vec{c}\left(k'_{7-8}(d)\right)=\left(c\left(k'_{7-8}(d),\ 1\right),\ ...,\ c\left(k'_{7-8}(d),\ 4\ 096\right)\right)$, каждый элемент которого характеризует зависимость между рассчитанными на перебираемом ключе значениями $S_{7-8}^{OUT}\left(i,\ k'_{7-8}(d)\right)$ и соответствующим отсчётом снятого массива форм сигнала.

Согласно приведённым в разделе 1.7 теоретическим доводам, на истинном варианте перебираемого ключа, должен наблюдаться всплеск значений корреляционного вектора, в областях соответствующих отсчётам, на снятых формах сигнала, характеризующих энергопотребление микроконтроллера в моменты пересылки байта $S_{7-8}^{OUT}\left(i,\ k'_{7-8}(d)\right)$ в ОЗУ. На других отсчётах или других вариантах перебираемого ключа всплеск также возможен, но должен иметь заметно меньшую амплитуду.

На рисунках 2.28 приведены корреляционные вектора $\vec{c}\left(k'_{7-8}(d)\right)$ для истинного и наихудшего ложного вариантов перебираемого ключа. Из рисунков 2.28 видно, что всплески на ложных ключах будут иметь как минимум на 25% меньшую амплитуду. Отсюда очевидна возможность однозначного определения истинного ключа путём визуального или машинного анализа амплитуды всплесков корреляционного вектора.

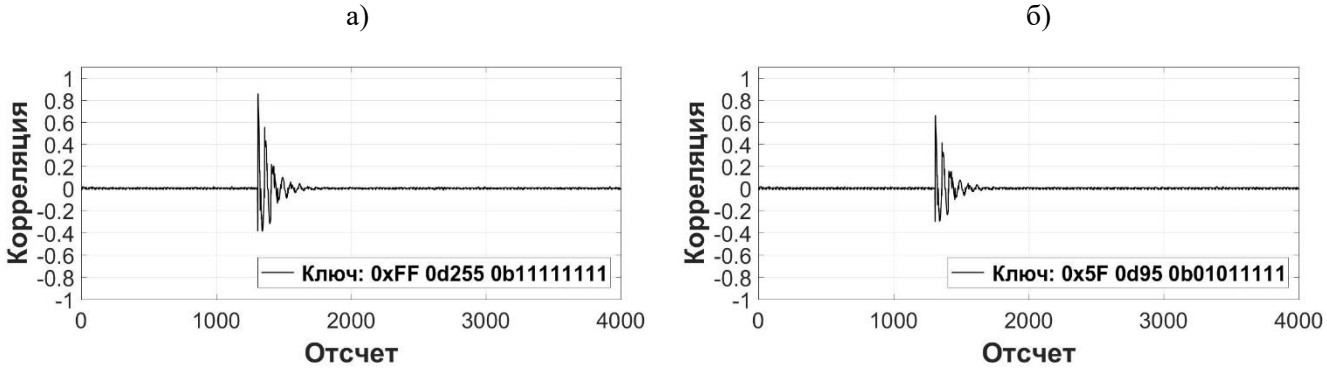


Рисунок 2.28 – Корреляционные вектора, полученные при использовании в качестве рассчитываемого промежуточного значения, комбинации $S_{7-8}^{OUT}\left(i,\ k'_{7-8}(d)\right)$, записываемой в чистую ячейку ОЗУ , при обработке 50 000 форм сигнала на: а) истинном; б) «наихудшем» для злоумышленника ложном вариантах ключа

Представляет интерес экспериментальная проверка доводов, представленных в формулах (1.11)-(1.16), сводящихся к тому, что, при увеличении количества обрабатываемых

форм сигнала, шумовая компонента корреляционного вектора устремляется к нулю. На рисунках 2.29 представлены корреляционные вектора, рассчитанные при использовании 2 000, 1 000, 100 и 30-ти форм сигнала.

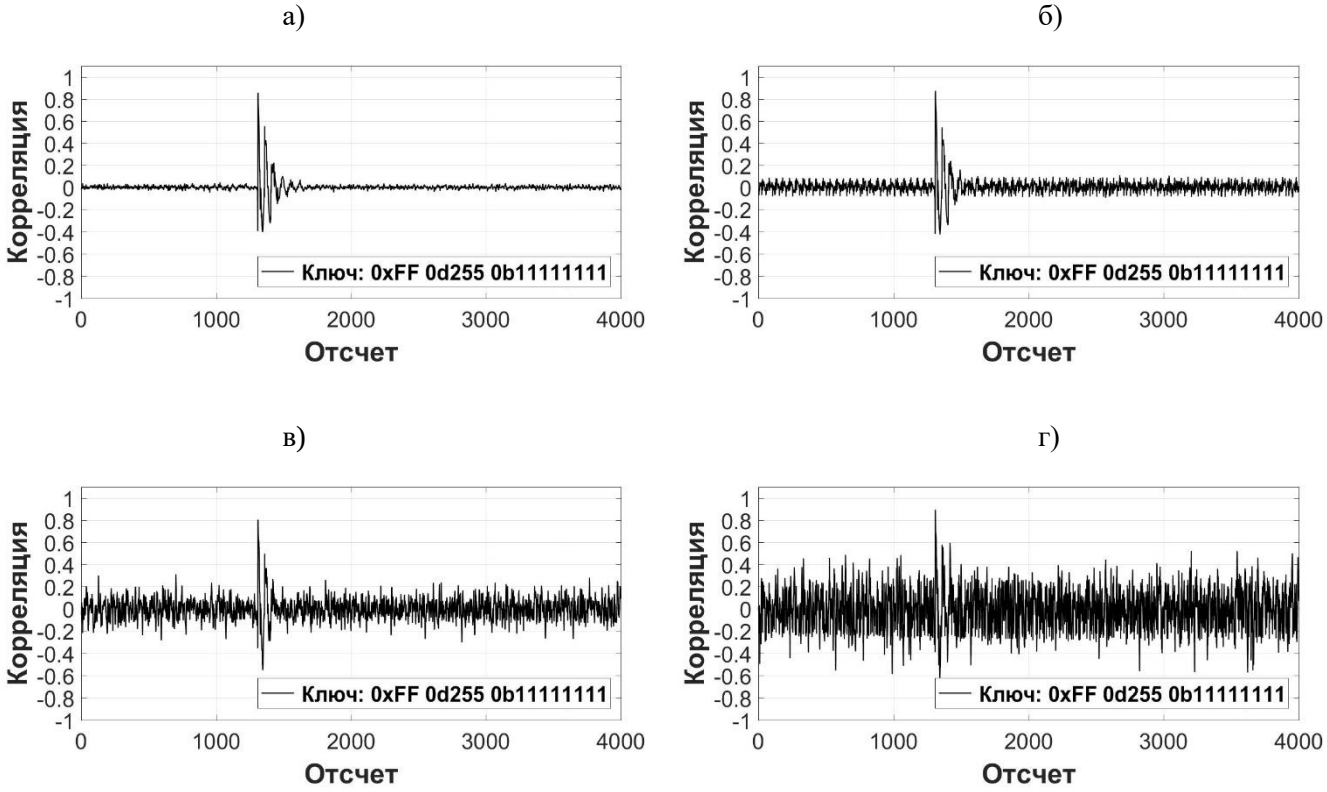


Рисунок 2.29 – Корреляционные вектора, полученные при использовании в качестве рассчитываемого промежуточного значения, комбинации $S_{7-8}^{OUT}\left(i,\ k'_{7-8}(d)\right)$, записываемой в чистую ячейку ОЗУ, на истинном варианте ключа, при обработке: а) 2 000 форм сигнала; б) 1 000 форм сигнала; в) 100 форм сигнала; г) 30 форм сигнала

Визуальный анализ позволяет сделать вывод о том, что уменьшение количества обрабатываемых форм сигнала влечёт за собой увеличение шумовой компоненты, что экспериментально подтверждает представленные ранее теоретические выкладки. Для рассматриваемой операции даже при использовании 30-ти форм сигнала возможно однозначное определение истинного ключа (всплеск на наихудшем ложном варианте ключа будет иметь амплитуду на 15% меньше, чем на истинном). Полученные экспериментальные результаты подтверждают чрезвычайную опасность рассматриваемого метода взлома чипов.

Существует очень удобный подход, описанный во множестве работ [10; 11; 41; 44; 57; 59], позволяющий определить, как изменяется амплитуда всплесков корреляционного вектора, в зависимости от количества обрабатываемых форм сигнала. Алгоритм заключается в следующем:

1. Выбирается шаг (количество форм сигнала), с которым будут производиться атаки DPA, в идеальном случае шаг должен быть равен одной форме сигнала.
2. Количество форм сигнала увеличивается на величину шага.
3. С использованием количества форм сигнала из п.2 рассчитываются корреляционные вектора $\vec{c}\left(k'_{7-8}(d)\right)$ для всех вариантов перебираемого ключа.
4. Для каждого из 256-ти полученных векторов $\vec{c}\left(k'_{7-8}(d)\right)$, определяется максимальный и минимальный элемент (то есть значения амплитуд максимального положительного и максимального отрицательного всплесков), эти значения наносятся на график.
5. Пункты 2-4 повторяются для количества форм сигнала, увеличенного на величину шага.
6. Максимумы, для корреляционных векторов, полученных при обработке разного количества форм сигнала, на каждом варианте перебираемого ключа, соединяются линиями, в итоге получается график, представленный на рисунках 2.30.

Далее такие графики будут называться сводными графиками корреляционных векторов.

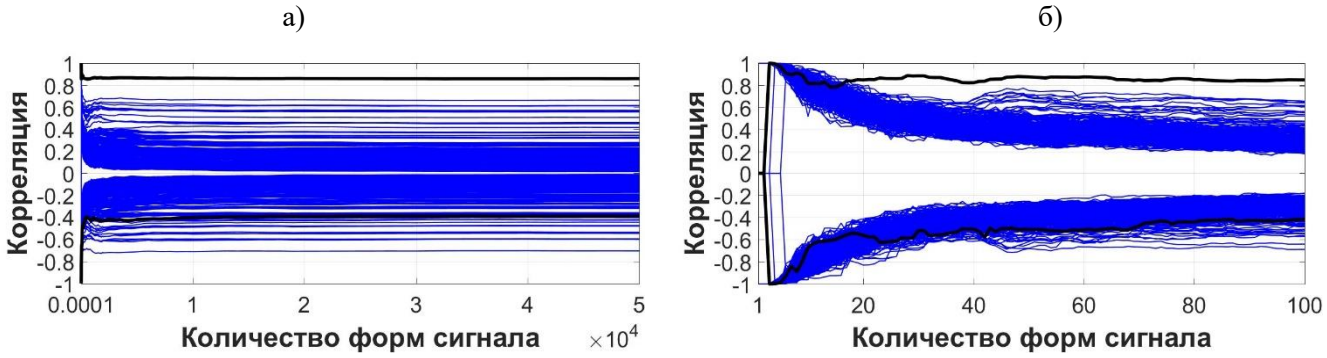


Рисунок 2.30 – Сводные графики корреляционных векторов для операции записи в чистую ячейку ОЗУ: а) диапазон от двух до 50 000 форм сигнала; б) диапазон от двух до 100 форм сигнала

Анализируя представленные графики, можно сделать вывод, что к примеру, максимальный всплеск корреляционного вектора $\vec{c}\left(k'_{7-8}(d)\right)$, рассчитанного на истинном ключе, при проведении атаки DPA с использованием 51 200 форм сигнала составит примерно 0.88 (чёрная жирная линия), максимальная корреляция на ложных ключах не превысит 0.68. Из рисунка 2.30б видно, что максимальные положительные всплески на истинном варианте перебираемого ключа будут иметь б**о**льшие амплитуды, чем на 255-ти ложных, начиная приблизительно с 20-ти обрабатываемых форм сигнала. То есть атака DPA с использованием промежуточного значения, сохраняемого в ячейку ОЗУ, окажется успешной при использовании весьма небольшого количества форм сигнала.

Из представленных выше сводных графиков корреляционных векторов видно, что однозначное определение истинного варианта ключа возможно только на основе сравнения положительных всплесков корреляционных векторов (то есть всплесков, обусловленных прямой зависимостью энергопотребления от обрабатываемых данных). Из рисунков 2.30 видно, что отрицательные всплески (обусловленные обратной зависимостью энергопотребления от обрабатываемых данных) на многих ложных вариантах ключа будут иметь б**о**льшее абсолютное значение амплитуды чем на истинном. Интересно рассмотреть эффективность определения истинного ключа на основе сравнения сумм абсолютных значений амплитуд максимального положительного и максимального отрицательного всплесков на каждом корреляционном векторе (рисунки 2.31). Далее такие графики будут называться сводными графиками корреляционных векторов по сумме максимумов и минимумов.

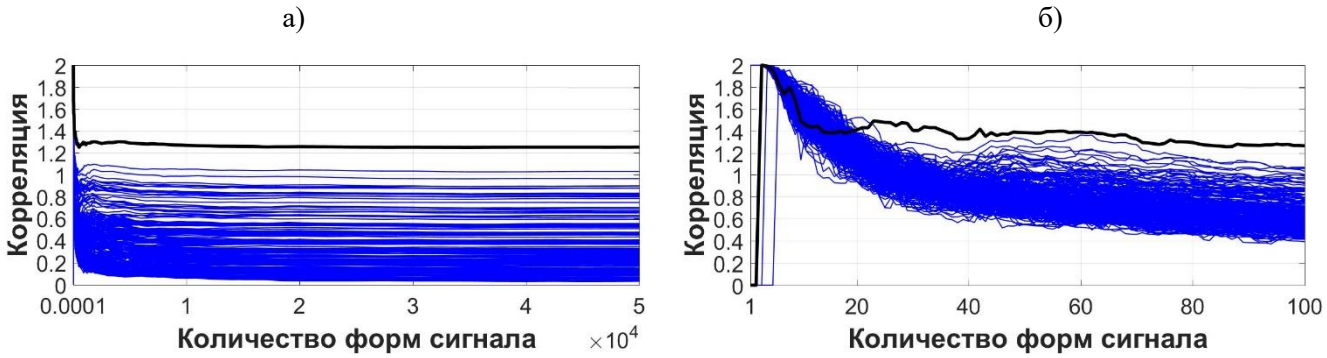
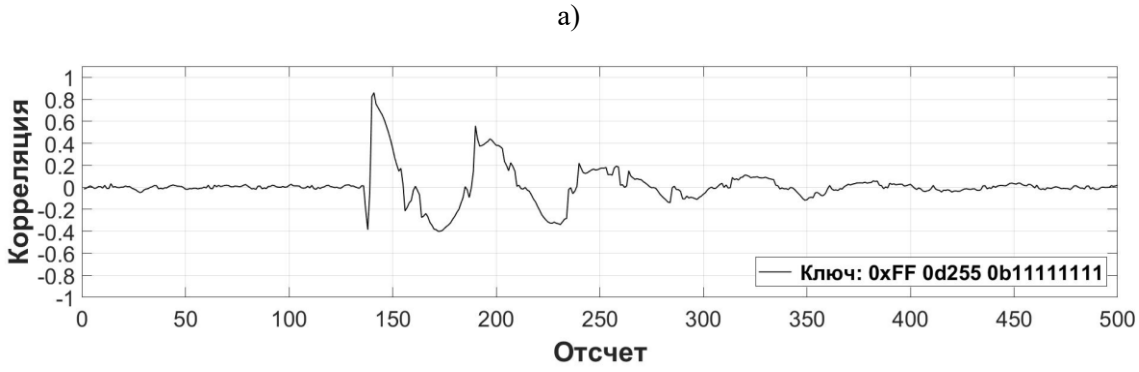


Рисунок 2.31 – Сводные графики корреляционных векторов по сумме максимумов и минимумов для операции записи в чистую ячейку ОЗУ: а) диапазон от двух до 50 000 форм сигнала; б) диапазон от двух до 100 форм сигнала

Анализ графиков позволяет сделать вывод, что метод, учитывающий, как положительные, так и отрицательные всплески на корреляционных векторах, не приводит к увеличению эффективности атаки DPA (по крайней мере для операции записи в ОЗУ).

Произведём более подробный анализ корреляционного вектора, полученного на истинном варианте перебираемого ключа (рисунки 2.32).



б) в)

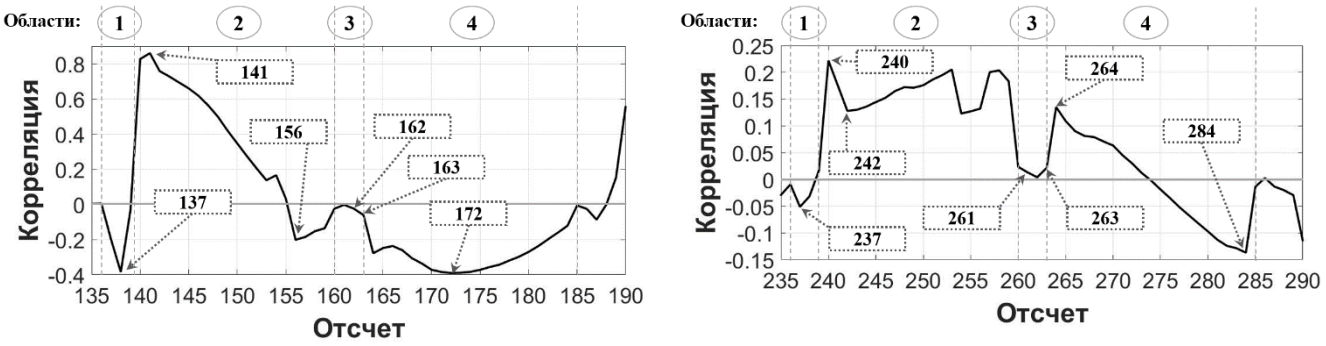


Рисунок 2.32 – а)-в) Увеличенные области корреляционного вектора, полученного при использовании в качестве рассчитываемого промежуточного значения, комбинации $S_{7-8}^{OUT}\left(i,\ k'_{7-8}\left(d\right)\right)$, записываемой в чистую ячейку ОЗУ, на истинном ключе, при обработке 50 000 форм сигнала

Проанализируем полученные всплески, с учётом данных о значениях $U_{\text{ДАН}}$, СКО и SNR полученных в подразделе 2.4.1, и представленных в таблице Е.1 приложения Е. На рисунке 2.32б изображены корреляционные всплески, соответствующие первому такту выполнения рассматриваемой операции (136-185 отсчёты снятых форм сигнала). В первой области такта максимальный всплеск имеет характер отрицательной зависимости, возникает на 138-м отсчёте, что соответствует максимальным значениям $U_{\text{ДАН}}$ и SNR. Во второй области такта максимальная амплитуда всплеска наблюдается на 141-м отсчёте, что также соответствует максимальным значениям $U_{\text{ДАН}}$ и SNR. В третьей области такта максимальное значение $U_{\text{ДАН}}$ наблюдается на 162-м отсчёте (составляя 0.462 В), однако за счёт значительной шумовой составляющей SNR в этом отсчёте составляет ничтожные 0.0007, в тоже время на 163-м отсчёте значение $U_{\text{ДАН}}$ несколько ниже (0.337 В), однако за счёт несколько меньшей шумовой компоненты, SNR в этом отсчёте увеличивается до 0.0038. Из рисунка 2.32б видно, что на 163-м отсчёте абсолютное значение корреляционного всплеска будет иметь б**о**льшую амплитуду чем на 162-м. Отсюда можно сделать вывод, что даже при усреднении значительного количества форм сигнала шумовая компонента оказывает заметное влияние на амплитуду корреляционного всплеска. Максимальная амплитуда всплеска в третьей области, аналогично первым двум – соответствует максимальным значениям $U_{\text{ДАН}}$ и SNR.

На рисунке 2.32в изображены корреляционные всплески, соответствующие третьему такту выполнения рассматриваемой операции (236-285 отсчёты снятых форм сигнала). В первой области третьего такта максимальное значение $U_{\text{ДАН}}$ соответствует максимальному значению SNR и наблюдается на 237-м отсчёте, именно в этом отсчёте всплеск будет иметь наибольшую

амплитуду. Во второй области такта максимальное значение $U_{ДАН}$ наблюдается на 242-м отсчёте составляя 0.271 В, при СКО шумовой компоненты равной 2.98, SNR в этом случае составляет 0.01648. На 240-м отсчёте значение $U_{ДАН}$ окажется несколько меньше (0.244 В), однако за счёт меньшего СКО шумовой компоненты: 1.74, SNR возрастёт более чем в три раза до 0.05361. Из рисунка 2.32в видно, что наибольший всплеск возникает именно на 240-м отсчёте, а всплеск на 242-м отсчёте имеет на 40% меньшую амплитуду. Аналогичная ситуация наблюдается в третьей области. В четвёртой области максимальное значение $U_{ДАН}$ для прямой зависимости наблюдается на 264-м отсчёте составляя 0.194 В, максимальное значение для обратной зависимости – на 284-м отсчёте, составляя 0.140 В, и на этом отсчёте наблюдается наибольший SNR – 0.01905 против 0.01802 на 264-м отсчёте. Это происходит за счёт того, что несмотря на меньшее значение $U_{ДАН}$, шумовая компонента также имеет небольшое значение составляя 1.53, против 1.71 на 264-м отсчёте. Отсюда в подтверждение выдвинутой гипотезы абсолютное значение амплитуды всплеска на 284-м отсчёте будет больше чем на 264-м (0.1372 против 0.1352).

На основе приведённого выше анализа полученных корреляционных векторов можно сделать следующие основные выводы:

- Полярность корреляционного всплеска определяется характером зависимости энергопотребления от обрабатываемых данных – при прямой зависимости всплеск положительный, а при обратной – отрицательный. Учитывая, что характер зависимости неоднократно меняется даже в рамках одного такта, и определяется, предположительно, конструктивными особенностями чипа, можно сделать вывод, что полезными для злоумышленника могут быть всплески обеих полярностей.
- Эффект остаточной корреляции весьма чётко проявляется на корреляционных векторах, можно заметить, что зависимость энергопотребления от обрабатываемых данных существует минимум на шести тактах работы микроконтроллера, хотя рассматриваемая операция выполняется всего за два такта.
- Амплитуда корреляционных всплесков определяется значением полезной составляющей сигнала: $U_{ДАН}$ и СКО шумовой компоненты, *даже после усреднения весьма большого количества форм сигнала*. Причины этого явления будут более подробно объяснены третьей главе.

Корреляционные вектора, для операции записи в предварительно заполненную ячейку ОЗУ, при использовании модели расстояния Хэмминга и веса Хэмминга представлены на рисунках 2.33. Анализ графиков подтверждает сделанные в подразделе 2.4.2 выводы, об

отсутствии зависимости между энергопотреблением микроконтроллера и количеством изменившихся бит в ячейке-назначения ОЗУ.

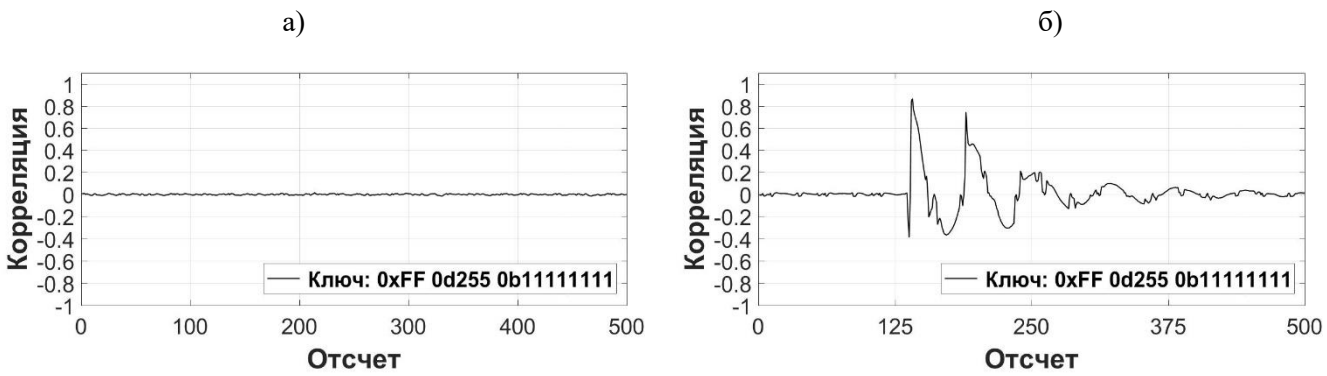


Рисунок 2.33 – Корреляционные вектора, полученные при использовании в качестве рассчитываемого промежуточного значения, комбинации $S_{7-8}^{OUT}\left(i,\ k'_{7-8}(d)\right)$ записываемой в предварительно заполненную ячейку ОЗУ, при обработке 50 000 форм сигнала, на истинном ключе, с применением модели: а) расстояния Хэмминга; б) веса Хэмминга

Корреляционные вектора, рассчитанные для операции записи в чистую ячейку ОЗУ и предварительно заполненную, при использовании модели веса Хэмминга очень похожи, сводные график корреляционных векторов представлены на рисунке 2.34.

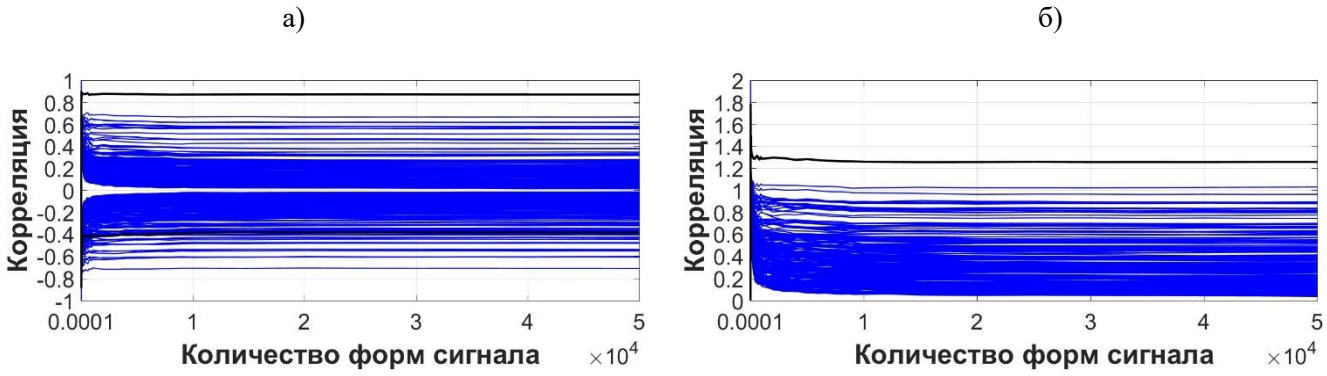


Рисунок 2.34 – а) Сводные графики корреляционных векторов; б) сводные графики корреляционных векторов по сумме максимумов и минимумов для операции записи в предварительно заполненную ячейку ОЗУ, при использовании модели веса Хэмминга

На рисунке 2.35 представлены сводные графики корреляционных векторов для диапазона количества форм сигнала от 2 до 256-ти.

а) б)

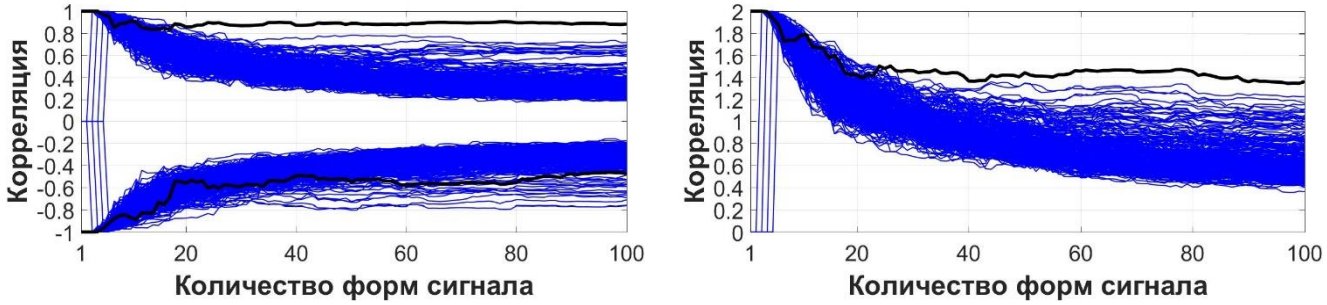


Рисунок 2.35 – Сводные графики корреляционных векторов; б) сводные графики корреляционных векторов по сумме максимумов и минимумов, для операции записи в предварительно заполненную ячейку ОЗУ, при использовании модели веса Хэмминга, диапазон от двух до 100 форм сигнала

Сравнение рисунка 2.35 с рисунками 2.30, 2.31 позволяет сделать вывод о том, что отсутствует существенная разница в эффективности реализации DPA, с использованием промежуточного значения, сохраняемого в чистую или предварительно заполненную ячейку ОЗУ, в случае использования модели веса Хэмминга. Поэтому независимо от того, осуществляется ли запись в чистую ячейку ОЗУ, или предварительно заполненную – в обоих случаях, для определения записываемой комбинации, по снятым формам сигнала, необходимо использовать модель веса Хэмминга.

2.5.3 Реализация DPA для иных операций

Описанный эксперимент был повторён для остальных операций рассмотренных в разделе 2.4, в результате чего были подтверждены выводы, сделанные в предыдущем подразделе – полярность всплесков на корреляционных векторах определяется характером зависимости обрабатываемых данных от энергопотребления, а амплитуда всплесков, определяется значениями SNR (когда в качестве полезного сигнала выбираются значения $U_{\text{ДАН}}$).

В таблице 2.3 представлены основные качественные характеристики корреляционных векторов, полученных при реализации атак DPA для каждой рассматриваемой операции. В частности, для каждой операции приведена амплитуда максимального всплеска корреляционного вектора, рассчитанного на истинном и наихудшем для злоумышленника ложном вариантах перебираемого ключа (при обработке 51 200 форм сигнала). Также на основе рассчитанных сводных графиков корреляционных векторов определено минимальное количество форм сигнала, требующееся для однозначного определения истинного ключа. В

отношении последнего необходимо отметить, что представленные цифры являются весьма приближёнными и на небольших выборках для различных итераций могут отличаться в 2-3 раза, ввиду сильного влияния оказываемого шумовой составляющей на корреляционные всплески. Однако очевидно, что именно эта характеристика является наиболее правильной оценкой эффективности реализации DPA в отношении конкретной операции и применяемой модели.

Из таблицы 2.3 видно, что для разных операций амплитуды максимальных всплесков на истинном и наихудшем для злоумышленника ложном вариантах ключа могут весьма сильно отличаться. Объяснение этому явлению весьма простое – шумовые компоненты (как электронный так и алгоритмический шум) на снимаемых, при выполнении рассматриваемых операций, формах сигнала имеют приблизительно одинаковое СКО, однако величина полезного сигнала $U_{\text{ДАН}}$ для разных операций весьма сильно отличается. Отсюда значения SNR, а, следовательно, и определяемая ими амплитуда корреляционных всплесков, будут значительно отличаться (математическая основа данного явления будет подробно рассмотрена в следующей главе). Отсюда положим, что при выполнении операции записи в ОЗУ, значение $U_{\text{ДАН}}$ таково, что при обработке 51 200 форм сигнала амплитуда максимального корреляционного всплеска на истинном ключе составляет 0.86, а на наихудшем ложном – 0.66. Тогда как при чтении байта данных из ОЗУ в чистую ячейку рабочего регистра, согласно таблице 2.3 соотношение будет следующим: 0.22 для истинного ключа и 0.18 для ложного. Учитывая, что влияние шумовой составляющей в обоих случаях одинаково убывает с увеличением количества обрабатываемых форм сигнала, становится очевидным, что для операции чтения данных из ОЗУ потребуется обработка значительно большего количества форм сигнала, чтобы было возможно чётко отличить всплески на истинном ключе от всплесков на ложных.

Таблица 2.3 – Основные характеристики корреляционных векторов, полученных при реализации DPA в отношении ряда операций

Выполняемая операция	Состояние ячейки назначения	Используемая модель энергопотребления	Вид анализа: по максимальным положительным всплескам или по сумме положительных и отрицательных	Максимальная амплитуда всплеска на истинном ключе	Максимальная амплитуда всплеска на наихудшем ложном ключе (абсолютная и в процентах от истинного)	Минимальное количество форм сигнала, требуемое для успешной реализации атаки
Копирование из ОЗУ	Чистая	Вес Хэмминга	Положительные	0.22	0.18 (80.6 %)	410
			Сумма	0.26	0.23 (88.2 %)	830
	Предварительно заполненная	Вес Хэмминга	Однозначное определение истинного ключа невозможно			
		Расстояние Хэмминга	Положительные	0.14	0.09 (64.9 %)	560
			Сумма	0.18	0.12 (66.0 %)	850
Копирование из ПЗУ	Чистая	Вес Хэмминга	Положительные	0.53	0.52 (98.3 %)	160
			Сумма	0.84	0.78 (92.8 %)	160
	Предварительно заполненная	Вес Хэмминга	Однозначное определение истинного ключа невозможно			
		Расстояние Хэмминга				
Копирование между рабочими регистрами	Чистая	Вес Хэмминга	Положительные	0.70	0.60 (85.7 %)	25
			Сумма	0.91	0.78 (85.5 %)	31
	Предварительно заполненная	Вес Хэмминга	Положительные	0.38	0.28 (73.5 %)	170
			Сумма	0.48	0.38 (79.8 %)	340
		Расстояние Хэмминга	Положительные	0.17	0.11 (67.0 %)	1580
			Сумма	0.21	0.15 (69.0 %)	1120
Сложение по mod2 между рабочими регистрами	Вес Хэмминга комбинации из первой ячейки		Положительные	0.72	0.59 (82.4 %)	20
			Сумма	0.95	0.79 (83.3 %)	32
	Вес Хэмминга суммы по mod2		Положительные	0.065	0.043 (66.1 %)	2500
			Сумма	0.087	0.059 (67.9 %)	8500

Из таблицы 2.3 видно, что, к примеру, если в качестве промежуточного значения выбирается комбинация, записываемая в чистую ячейку ОЗУ, то максимальный корреляционный всплеск на истинном варианте перебираемого ключа составит 0.22, а на наихудшем ложном – 0.18, или 80.6% от истинного (рисунки И.1 приложения И). При этом однозначное определение истинного варианта ключа окажется возможным при обработке 410 и более форм сигнала (рисунки И.1в,г приложения И). Также понятно, что использование суммы положительных и отрицательных всплесков окажется менее эффективным (разница между всплесками на истинном и ложном ключах будет меньше).

Повторение эксперимента для чтения в предварительно заполненную ячейку регистра, показывает, что атака с использованием модели веса Хэмминга окажется не эффективной (всплески корреляционных векторов, полученных на ложных вариантах ключах, будут иметь б**о**льшие амплитуды чем на истинном, рисунки И.2 приложения И). А при использовании модели расстояния Хэмминга эффективность атаки оказывается приблизительно такой же, как и при чтении в чистую ячейку регистра (рисунки И.3 приложения И). В целом можно заключить, что реализация DPA с использованием комбинации считываемой из ОЗУ в ячейку рабочего регистра оказывается заметно менее эффективной, чем для операции записи в ОЗУ.

Чрезвычайно важной для рассмотрения, является операция чтения из ПЗУ (Flash памяти) – как ранее говорилось, при помощи данной операции реализуются преобразования на S-box. В таких алгоритмах как DES, ГОСТ, AES выход S-box является зависимым от преобразуемого сообщения и раундового ключа, поэтому, обладая минимальными знаниями о реализации алгоритма на конкретном чипе, злоумышленник прежде всего, в качестве промежуточного значения для DPA, старается использовать выход S-box, преобразование на котором всегда реализуется при помощи подхода «чтения со смещением» из ПЗУ. На анализируемом микроконтроллере эта операция является самой «неблагоприятной» для злоумышленника по следующим причинам:

- Существует значительная «фоновая корреляция» – всплески корреляции от -0.1 до 0.1, на тех участках, на которых не выполняется никаких операций (рисунки И.4 приложения И). Этот «фон» возникает в том числе и на ложных вариантах перебираемого ключа, на них он имеет как меньшую, так и большую амплитуду.
- Крайне небольшая разница между амплитудами всплесков корреляционных векторов, полученных на истинном и ложном вариантах ключа.
- Отсутствие возможности однозначного определения истинного варианта ключа при чтении из ПЗУ в предварительно заполненную ячейку регистра.

Эффективность реализации DPA в отношении операции копирования данных из одного рабочего регистра в другой, предварительно очищенный регистр, оказывается весьма высокой (сравнима с операцией записи в чистую ячейку ОЗУ) – определение ключа будет возможно после обработки всего нескольких десятков форм сигнала (рисунки И.5 приложения И). Эффективность атаки значительно снижается в случае наличия в регистре назначения каких-либо данных (рисунки И.6-И.7 приложения И), в частности значительно увеличивается количество требуемых форм сигнала, но несмотря на это определение копируемой комбинации остаётся возможным.

Не менее эффективной является атака, в случае использования в качестве промежуточного значения, аргументов операции сложения по mod2. В данном случае, предполагается, что злоумышленнику известно содержимое первого регистра, а в качестве рассчитываемого промежуточного значения используется зависящее от ключа содержимое второго регистра или суммы первого и второго регистров. Такой сценарий атаки вполне возможен, например, при сложении выхода S-box с левой частью входа раунда алгоритмов «Магма» и DES. В алгоритм работы микроконтроллера (подраздел 2.5.1) необходимо внести исправления на третьем и девятом шагах – на третьем шаге ПК должен отсылать на микроконтроллер не один, а два случайных байта – второй записывается в 20-й регистр. На девятом шаге содержимое 19-го регистра (значение $S_{7-8}^{OUT}(i)$) складывается с 20-м, с сохранением результата в 19-м регистре. Очевидно, что для определения содержимого 19-го регистра необходимо применять модель веса Хэмминга подсчитывая количество единиц в комбинации $S_{7-8}^{OUT}(i)$, зависящей от неизвестного ключа. Для определения суммы 19-го и 20-го регистров на первый взгляд можно применять как модель веса Хэмминга (определяя количество единиц в полученной сумме по mod2), так и модель расстояния Хэмминга, определяя количество изменившихся бит в процессе перезаписи, рассчитанной суммой по mod2, содержимого регистра, в который производится запись данной суммы. Рассмотрим последний случай подробнее, обозначим содержимое 19-го регистра как $R_{19}(i) = S_{7-8}^{OUT}(i)$, содержимое 20-го регистра как $R_{20}(i)$, их сумму по mod2 как $R_{19}(i) \oplus R_{20}(i)$. Тогда, если предположить, что полученная сумма перезаписывает содержимое 19-го регистра, то расстояние Хэмминга между полученной суммой $R_{19}(i) \oplus R_{20}(i)$ и содержимым 19-го регистра будет равно $HW\left[\left(R_{19}(i) \oplus R_{20}(i)\right) \oplus R_{19}(i)\right] = HW\left(R_{20}(i)\right)$, то есть весу Хэмминга содержимого 20-го регистра. Но во-первых содержимое 20-го регистра не зависит от перебираемого ключа (а следовательно на рассчитанном корреляционном векторе будут отсутствовать всплески), а во-вторых $R_{20}(i)$ как-правило известно злоумышленнику, поэтому модель расстояния Хэмминга в данном случае не даст ожидаемого результата. Теперь предположим, что полученная сумма

перезаписывает содержимое не 19-го, а 20-го регистра, тогда расстояние Хэмминга между полученной суммой $R_{19}(i) \oplus R_{20}(i)$ и содержимым 20-го регистра будет равно $HW\left[\left(R_{19}(i) \oplus R_{20}(i)\right) \oplus R_{20}(i)\right] = HW\left(R_{19}(i)\right)$, то есть весу Хэмминга 19-го регистра, что очевидно эквивалентно случаю с расчётом веса Хэмминга комбинации $S_{7-8}^{OUT}(i)$. Поэтому для определения содержимого 19-го регистра и для определения результата сложения по mod2 содержимого 19-го и 20-го регистров следует применять модель веса Хэмминга.

Реализация атаки DPA, с расчётом веса Хэмминга содержимого 19-го регистра (то есть $S_{7-8}^{OUT}(i)$), окажется на столько же эффективной как и операции записи в ОЗУ и копирования данных между рабочими регистрами (рисунки И.8 приложения И). В случае использования результата сложения по mod2 эффективность атаки существенно снижается, однако несмотря на это, при использовании значительного количества форм сигнала, она вполне возможна (рисунки И.9 приложения И).

Из приведённых выше экспериментальных данных можно сделать следующие промежуточные выводы:

- Во всех рассмотренных операциях, на которых присутствовали корреляционные всплески – они выходили за рамки тех тактов, на которых выполнялась соответствующая команда. Так как защитные алгоритмы в чипах реализуются путём последовательного выполнения соответствующих программных команд, это приведёт к неминуемому наложению скачков энергопотребления, вызванных выполнением одних команд (операций), на те команды, которые следуют за ними. Это означает, что на операцию, аргументы которой, будут использоваться в виде промежуточного значения для атаки DPA, будет неминуемо накладываться дополнительный алгоритмический шум.
- Для многих операций успешная реализация атаки DPA возможна после обработки менее 100 форм сигнала, что делает данную атаку чрезвычайно опасной.
- Целесообразность применения модели расстояния Хэмминга, даже в том случае, когда происходит запись данных предварительно заполненную ячейку памяти для многих операций отсутствует. Известное из многих теоретических работ [10; 11; 42; 55; 63], строгое разграничение случаев, применения модели веса Хэмминга и модели расстояния Хэмминга не подтверждается экспериментально.

Также важным на практике является подтверждение следующего допущения: многоразрядные чипы (в частности, рассматриваемые, восьми разрядные) обрабатывают комбинации равные их разрядности не побитово, а одновременно. Следовательно, на формах сигнала возможно отличить лишь общий всплеск энергопотребления характеризующий вес

Хэмминга обрабатываемой комбинации в целом, а не восемь отдельных всплесков, характеризующих каждый бит в отдельности. По этой причине в качестве промежуточного значения защитного алгоритма, рассчитываемого на перебираемом ключе в процессе реализации DPA, строго рекомендуется выбирать комбинацию равную разрядности чипа – это позволит минимизировать алгоритмический шум. К примеру, если в приведённом выше примере для операции записи в ОЗУ, в качестве рассчитываемого промежуточного значения вместо восьми бит выхода 7-8 S-box выбирать один бит, то оставшиеся семь бит, обрабатываемые одновременно с ним, будут представлять уже не полезную составляющую, а дополнительный (весьма значительный) алгоритмический шум. Вследствие чего значительно снизится SNR, поэтому для успешного различения всплесков, полученных на истинном и ложных вариантах перебираемого ключа потребуется обработка большего количества форм сигнала. В принципе допущение, что обрабатываемые чипом восьми битовые комбинации приводят к общему всплеску, а не к последовательной серии из восьми отдельных, уже было подтверждено представленными в разделе 2.4 и в приложении Д усреднёнными формами сигнала. Однако его можно подтвердить ещё одним, более надёжным и наглядным способом – произвести расчёт корреляционных векторов с использованием каждого из восьми бит выхода S-box по отдельности. Очевидно, что в случае последовательной обработки бит, всплески на корреляционном векторе, должны смещаться по временной оси, в противном случае они должны оставаться на одних и тех же местах (рисунки 2.36).

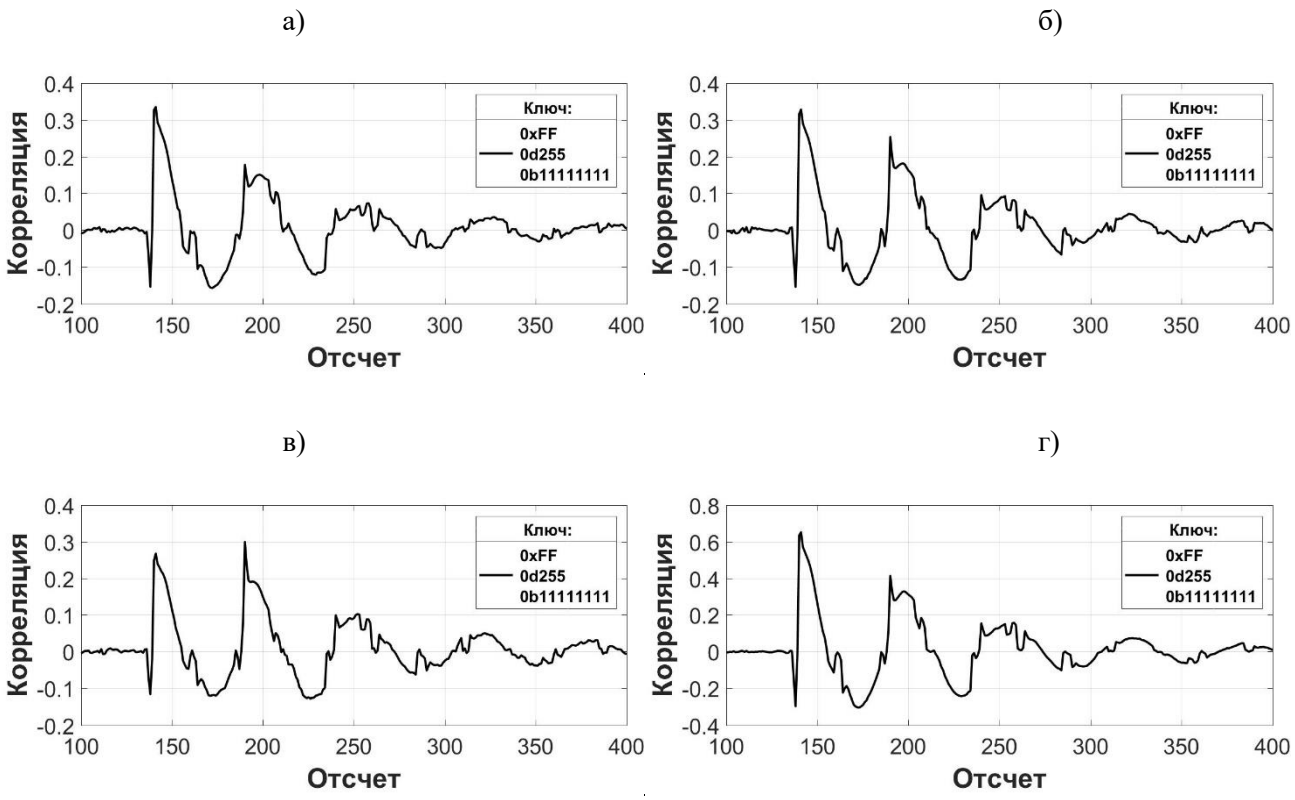


Рисунок 2.36 – Увеличенная область корреляционного вектора, полученного после обработки 50 000 форм сигнала, на истинном ключе при использовании в качестве рассчитываемого промежуточного значения, комбинации $S_{7-8}^{OUT}\left(i,\ k'_{7-8}(d)\right)$ копируемой в чистую ячейку ОЗУ, и расчёте веса Хэмминга: а) бита младшего разряда комбинации $S_{7-8}^{OUT}\left(i,\ k'_{7-8}(d)\right)$; б) бита второго разряда комбинации $S_{7-8}^{OUT}\left(i,\ k'_{7-8}(d)\right)$; в) бита восьмого разряда комбинации $S_{7-8}^{OUT}\left(i,\ k'_{7-8}(d)\right)$; г) младших четырёх бит комбинации $S_{7-8}^{OUT}\left(i,\ k'_{7-8}(d)\right)$

Анализ всплесков корреляционных векторов, представленных на рисунках 2.36 показывает, что в случае выбора в качестве рассчитываемого промежуточного значения разных разрядов одной обрабатываемой комбинации, всплески остаются на тех же местах во временной области. Этот эксперимент безусловно подтверждает, что на снимаемых формах сигнала возможно отличить лишь общие всплески, характеризующие энергопотребление, затрачиваемое на обработку восьми разрядов в целом, а не каждого разряда по отдельности. Также проведённый эксперимент подтверждает, что в случае использования меньшего количества бит, снижается соотношение сигнал/шум на формах сигнала, и как следствие уменьшаются амплитуды корреляционных всплесков. Поэтому для чёткого различения всплесков, полученных на истинном и ложных вариантах перебираемого ключа потребуется обработка большего количества форм сигнала.

2.6 Выводы по главе

По результатам проведённых экспериментов были сделаны следующие основные выводы:

- Энергопотребление чипа зависит от обрабатываемых данных, что полностью согласуется с описанными в литературе теоретическими положениями.
- Амплитуда корреляционных всплесков определяется двумя параметрами отсчётов форм сигнала: значением $U_{\text{ДАН}}$ и СКО шумовой компоненты. Которые определяют соотношение сигнал/шум в выбранной точке формы сигнала, оно, в свою очередь, характеризует объём информации «утекающей» по цепи электропитания. Значение $U_{\text{ДАН}}$ сильно отличается (на порядки) при выполнении чипом разных операций с одними и теми же данными. Чем меньшее значение имеет $U_{\text{ДАН}}$, тем большее количество форм сигнала потребуется обработать, для достаточного снижения шумовой составляющей, чтобы однозначно отделить истинный вариант перебираемого ключа от ложных.
- Анализ таблиц приложения Е позволяет сделать вывод, что для разборчивого различения весов Хэмминга обрабатываемых на разных операциях данных, по усреднённым формам сигнала, необходима измерительная установка с шагом квантования в 100-200 мкВ.
- Было показано, что зависимость энергопотребления от обрабатываемых данных, имеет весьма сложный характер. Её направление неоднократно меняется даже в рамках одного такта работы анализируемого микроконтроллера (пример приведён на рисунке 2.15, а также рисунках приложения Д и таблицах приложения Е), при этом значение $U_{\text{ДАН}}$ (а следовательно и утечка информации по цепи электропитания) в разных областях такта значительно отличается. Используемая измерительная установка, работающая с частотой дискретизации 100 Мвыб/с, позволяла описать каждый такт работы микроконтроллера 50-ю отсчётами, что можно считать вполне достаточным. Однако, необходимо учесть, что анализируемый микроконтроллер тактировался от сравнительно низкочастотного кварцевого генератора с частотой 2 МГц, а реальные невскрываемые чипы могут тактироваться от генераторов с частотой в 8 МГц и более. Учитывая это можно сделать вывод, о том, что снижение частоты дискретизации измерительной установки, например, до 20-40 Мвыб/с приведёт к тому, что энергопотребление на каждом такте работы микроконтроллера будет описываться на снимаемых формах сигнала всего несколькими отсчётами. Следовательно, достоверность определения зависимости энергопотребления от обрабатываемых данных будет весьма низкая. Поэтому для задач анализа энергопотребления чипов, с целью дальнейшей реализации побочных атак по цепям электропитания, можно посоветовать использовать измерительные установки с частотой

дискретизации в несколько сотен Мвыб/с (по крайней мере, если предполагается сколько-нибудь значительная универсальность установки).

- При сравнении операций чтения и записи в пустые и предварительно заполненные ячейки, установлено, что в некоторых случаях энергопотребление практически не зависит от исходного состояния ячейки (операции: записи данных в ячейку ОЗУ, чтения данных из ячейки ПЗУ), а в некоторых эти зависимости существуют (операции: чтения данных из ОЗУ, копирования данных из регистра в регистр, сложение по модулю 2).
- Несмотря на то, что все исследованные операции выполняются устройством за 1-2 такта, зависимость энергопотребления от обрабатываемых данных наблюдалась и на 5-7-ми последующих тактах. При этом сильная зависимость наблюдалась на 2-3-ёх тактах, далее она спадала, постепенно сходя на нет. Поскольку в реальных чипах все операции выполняются последовательно друг за другом, описанные «остаточные» всплески будут накладываться на последующие операции, являясь для них шумом, что окажет негативное влияние на возможность анализа чипа.
- Рассинхронизацию форм сигнала можно принципиально устранить путём их предобработки (выравнивания), данный процесс также несложно автоматизировать. Следовательно, для успешного анализа энергопотребления чипа не требуется покупка дорогих высокостабильных источников тактового сигнала.
- Шумовую составляющую, присутствующую на формах сигнала легко минимизировать путем увеличения количества обрабатываемых форм сигнала, что подтверждает неэффективность борьбы с DPA при помощи зашумления цепи электропитания, а также подчеркивает высокую потенциальную опасность данной атаки, поскольку даже сильно зашумленные формы сигнала, после статистической обработки, поддаются успешному анализу.

Основное содержание главы и полученных научных результатов изложено в работах автора [15; 23; 28; 30; 31; 32].

ГЛАВА 3. ИССЛЕДОВАНИЕ ТЕОРЕТИЧЕСКИХ И ПРАКТИЧЕСКИХ АСПЕКТОВ РЕАЛИЗАЦИИ DPA

В первой главе были кратко приведены основные сведения об атаке DPA, во второй главе даны результаты практического исследования «утечек» информации по цепи электропитания для реального чипа. В данной главе будет рассмотрен ряд, не описанных ранее в известных источниках, особенностей побочных атак по цепям электропитания, в частности:

- приведено аналитическое обоснование эффекта занижения всплесков на корреляционных векторах, в зависимости от соотношения сигнал/шум;
- показано, что эффективность реализации DPA с расчётом корреляционных и ковариационных векторов отличается;
- показаны отличия в эффективности реализации DPA в отношении российских стандартов шифрования «Магма» и «Кузнечик».

В конце главы будет описана атака DPA на аппаратную реализацию шифра «Магма» с использованием экспериментальных данных, а также дана оценка её сложности.

3.1 Описание используемой модели энергопотребления чипа

Ниже будут рассмотрены два подхода (предполагающие расчёт ковариационных и корреляционных векторов), которые позволяют определить зависимость между рассчитываемым, на варианте перебираемого ключа, промежуточным значением защитного преобразования и снятыми формами сигнала. При этом, как было показано в главе 2, энергопотребление реального чипа имеет весьма сложную зависимость от обрабатываемых им данных. Более того, можно допустить, что для разных моделей интегральных чипов эти зависимости будут иметь весьма сильно отличающиеся особенности. Поэтому является вполне логичным оценить эффективность и описать особенности реализации этих подходов вначале с применением форм сигнала, сгенерированных специальным образом, с использованием некоторой идеализированной модели энергопотребления. А затем уже использовать формы сигнала, полученные экспериментально, от реального чипа.

Генерирование форм сигнала будет производиться с заданными параметрами шумовой и полезной (сигнальной) компонент. При этом, полагается, что моделируемое устройство имеет восьми разрядную архитектуру. Поэтому комбинации будут обрабатываться восьми битовыми

блоками, а скачки энергопотребления на формах сигнала, будут пропорциональны весу Хэмминга этих блоков.

Каждая форма сигнала генерируется согласно следующему алгоритму:

1. Генерируется случайное 128-ми битовое сообщение $m(i)$, $i = 1, ..., N$, где N – общее количество генерируемых форм сигнала.
2. С использованием полученного сообщения и раундового ключа k (для примера был взят ключ: $k = 00\ 01\ 02\ 03\ 04\ 05\ 06\ 07\ 08\ 09\ 0A\ 0B\ 0C\ 0D\ 0E\ 0F$), производится расчёт выходов 16-ти S-box «Кузнечика» первого раунда: $a(i, s)$, где $s = 1, ..., 16$ номер анализируемого S-box.
3. Генерируется шумовой вектор из 350 случайных элементов (представляющих отсчёты формы сигнала) с гауссовским распределением и нулевой средней $\xi(i, j)$, $j = 1, ..., 350$, $\mu = 0$, $\sigma_{ШУМ} = \sigma(\xi(i, j)) = 0,\ 0.1,\ 0.2,\ 0.3$.
4. Значение энергопотребления выхода каждого S-box прибавляется к соответствующему элементу шумового вектора в соответствии с формулой (3.1). При этом полагается, что энергопотребление, затрачиваемое на выработку каждой единицы обрабатываемой комбинации, равно некоторой величине $U_{ДАН}$ (возьмём к примеру $U_{ДАН} = 0.2$ некоторых условных единиц, допустим мВ), а на выработку нуля энергия не затрачивается.

$$S(i, j) = HW(a(i, s)) \cdot U_{ДАН} + \xi(i, j) = HW(a(i, s)) \cdot 0.2 + \xi(i, j), \qquad (3.1)$$

5. Отсчёты распределялись по формам сигнала в зависимости от обрабатываемого S-box следующим образом: 27-й отсчёт характеризует энергопотребление, затрачиваемое на выработку выхода 16-го S-box. Между отсчётами, характеризующими энергопотребление соседних S-box на формах сигнала был установлен промежуток в 19 отсчётов. Таким образом 47-й отсчёт, характеризует выработку выхода 15-го S-box, 67-й отсчёт – 14-го и так далее вплоть до 327-го отсчёта, относящегося к первому S-box.

При демонстрации реализаций DPA по известным сообщениям, на перебираемом варианте l-го подблока ключа $k_l'(d)$, $d = 1, ..., 256$ будет производиться расчёт промежуточной комбинации шифра «Кузнечик» $a_l'(i, k_l'(d))$, представляющей оценку выхода 16-го S-box первого раунда, следовательно будет анализироваться всплеск на корреляционном/ковариационном векторе, в районе 27-го отсчёта, далее этот отсчёт будет обозначаться как v.

3.2 Особенности реализации DPA с расчётом ковариационных векторов

В работе П. Кёхера [48], впервые описывающей принципы реализации DPA, для анализа чипа использовался подход с расчётом ковариационного вектора, при этом предлагалось рассчитывать на перебираемом подблоке ключа лишь один бит выхода S-box. В настоящее время авторы работ по теме DPA используют подход с расчётом корреляционных векторов при этом производится расчёт не одного бита промежуточного значения защитного преобразования, а количества бит, соответствующего разрядности чипа (или максимально приближенное к ней). Выше было показано, что использование восьми бит (в случае если производится анализ чипа с восьми разрядной архитектурой) приводит к значительно лучшему соотношению сигнал/шум, а, следовательно, для успешной реализации DPA потребуется обработка меньшего количества форм сигнала. Однако в известных источниках не приводилось сравнения эффективности применения подходов с расчётом ковариационных и корреляционных векторов. Ниже будут рассмотрены особенности применения этих подходов в контексте реализации DPA, а также произведён анализ их эффективности.

Из формул (1.12)-(1.16) известно, что при обработке достаточно большого количества форм сигнала, шумовая составляющая коэффициента ковариации может быть сведена к пренебрежимо малому уровню. В этом случае ковариация характеризует зависимость между рассчитанным весом Хэмминга выхода S-box $h_l'\left(i,\ k_l'(d)\right)$ «Кузнечика» и полезной составляющей отсчёта на формах сигнала $HW\left(a(i,\ s)\right)\cdot U_{\text{ДАН}}$. Очевидно, что при выборе истинного варианта подблока ключа для всех используемых сообщений выход S-box будет рассчитан верно, следовательно его вес Хэмминга будет строго соответствовать значению отсчёта v. В данном случае под суммой в формуле (1.16) будет возможно появление девяти различных сочетаний значений $h_l'\left(i,\ k_l'(d)\right)$ и $HW\left(a(i,\ s)\right)\cdot U_{\text{ДАН}}$, появляющихся с определённой вероятностью, приведённой в таблице 3.1.

Таблица 3.1 – Возможные сочетания значений $h_l'\left(i,\ k_l'(d)\right)$ и $HW\left(a(i,\ s)\right)\cdot U_{\text{ДАН}}$ на истинном варианте перебираемого подблока ключа

Вероятность появления	$h_l'\left(i,\ k_l'(d)\right)$	$HW\left(a(i,\ s)\right)\cdot U_{\text{ДАН}}$	$h_l'\left(i,\ k_l'(d)\right)\cdot HW\left(a(i,\ s)\right)\cdot U_{\text{ДАН}}$
1/256	8	$8\cdot U_{\text{ДАН}}$	$64\cdot U_{\text{ДАН}}$
8/256	7	$7\cdot U_{\text{ДАН}}$	$49\cdot U_{\text{ДАН}}$
28/256	6	$6\cdot U_{\text{ДАН}}$	$36\cdot U_{\text{ДАН}}$
56/256	5	$5\cdot U_{\text{ДАН}}$	$25\cdot U_{\text{ДАН}}$
70/256	4	$4\cdot U_{\text{ДАН}}$	$16\cdot U_{\text{ДАН}}$
56/256	3	$3\cdot U_{\text{ДАН}}$	$9\cdot U_{\text{ДАН}}$
28/256	2	$2\cdot U_{\text{ДАН}}$	$4\cdot U_{\text{ДАН}}$
8/256	1	$1\cdot U_{\text{ДАН}}$	$1\cdot U_{\text{ДАН}}$
1/256	0	$0\cdot U_{\text{ДАН}}$	$0\cdot U_{\text{ДАН}}$

Отсюда, на истинном варианте подблока ключа формулу (1.16), можно преобразовать в (3.2)-(3.3). Из чего видно, что в отсутствие шума (или при обработке количества форм сигнала, позволяющего минимизировать шумовую компоненту до пренебрежимо малых значений) всплеск на ковариационном векторе в v-м отсчёте будет достигать значения $2\cdot U_{\text{ДАН}}$. Если вариант подблока ключа злоумышленником угадан неверно, то за счёт однозначности операций сложения по mod2 и табличного нелинейного преобразования выходы S-box «Кузнечика» для всех N используемых сообщений будут рассчитаны неверно. Справедливо отметить, что веса Хэмминга неверно рассчитанных выходов S-box, могут совпадать с весами Хэмминга выходов S-box, рассчитанных на истинном варианте перебираемого подблока ключа. Например, исходя из вероятностных характеристик, представленных в таблице 1.2 видно, что если вес Хэмминга рассчитанного на истинном варианте подблока ключа выхода S-box равен к примеру трём, то существует ещё 55 других восьми битовых комбинаций имеющих тот же вес Хэмминга. Отсюда теоретически существует вероятность того, что выходы S-box, рассчитанные на определённом ложном варианте подблока ключа для всех N сообщений будут иметь те же веса Хэмминга, что и на истинном. Тогда формы сигнала будут корректно рассортированы по девяти массивам в зависимости от веса Хэмминга выхода S-box (третий столбец таблицы 3.1), поэтому на соответствующем ковариационном векторе всплеск будет иметь такую же амплитуду, что и на истинном варианте перебираемого подблока ключа.

$$COV\left(h_l'\left(k_l'(d)\right),\, S(v)\right) \xrightarrow[N\to\infty]{} \left[\frac{1}{N}\sum_{i=1}^{N}\left(h_l'\left(i,\, k_l'(d)\right)\cdot S(i,\, v)\right)\right] - 16\cdot U_{\text{ДАН}} =$$

$$\frac{1}{N}\left[\begin{array}{l}\sum\limits_{\substack{i:\\ h_l'(i,\,k_l'(d))=0}}^{N} 0\cdot S(i,\, v) + \sum\limits_{\substack{i:\\ h_l'(i,\,k_l'(d))=1}}^{N} 1\cdot S(i,\, v) + \sum\limits_{\substack{i:\\ h_l'(i,\,k_l'(d))=2}}^{N} 2\cdot S(i,\, v) + \sum\limits_{\substack{i:\\ h_l'(i,\,k_l'(d))=3}}^{N} 3\cdot S(i,\, v) + \sum\limits_{\substack{i:\\ h_l'(i,\,k_l'(d))=4}}^{N} 4\cdot S(i,\, v) + \\ + \sum\limits_{\substack{i:\\ h_l'(i,\,k_l'(d))=5}}^{N} 5\cdot S(i,\, v) + \sum\limits_{\substack{i:\\ h_l'(i,\,k_l'(d))=6}}^{N} 6\cdot S(i,\, v) + \sum\limits_{\substack{i:\\ h_l'(i,\,k_l'(d))=7}}^{N} 7\cdot S(i,\, v) + \sum\limits_{\substack{i:\\ h_l'(i,\,k_l'(d))=8}}^{N} 8\cdot S(i,\, v)\end{array}\right] - 16\cdot U_{\text{ДАН}} =$$

(3.2)

$$= \frac{1}{N}\left[\begin{array}{l} 0\cdot \sum\limits_{i=1}^{1\cdot N/256}\left(0\cdot U_{\text{ДАН}} + \xi(i,\, l)\right) + 1\cdot \sum\limits_{i=1}^{8\cdot N/256}\left(1\cdot U_{\text{ДАН}} + \xi(i,\, v)\right) + 2\cdot \sum\limits_{i=1}^{28\cdot N/256}\left(2\cdot U_{\text{ДАН}} + \xi(i,\, v)\right) + 3\cdot \sum\limits_{i=1}^{56\cdot N/256}\left(3\cdot U_{\text{ДАН}} + \xi(i,\, v)\right) + \\ + 4\cdot \sum\limits_{i=1}^{70\cdot N/256}\left(4\cdot U_{\text{ДАН}} + \xi(i,\, v)\right) + 5\cdot \sum\limits_{i=1}^{56\cdot N/256}\left(5\cdot U_{\text{ДАН}} + \xi(i,\, l)\right) + 6\cdot \sum\limits_{i=1}^{28\cdot N/256}\left(6\cdot U_{\text{ДАН}} + \xi(i,\, v)\right) + 7\cdot \sum\limits_{i=1}^{8\cdot N/256}\left(7\cdot U_{\text{ДАН}} + \xi(i,\, v)\right) + \\ + 8\cdot \sum\limits_{i=1}^{1\cdot N/256}\left(8\cdot U_{\text{ДАН}} + \xi(i,\, v)\right) \end{array}\right] - 16\cdot U_{\text{ДАН}},$$

При $N\to\infty$, очевидно, что шумовая составляющая (имеющая гауссовское распределение с нулевым средним) под суммами в формуле (3.2) устремится к нулю, тогда:

$$COV\left(h_l'\left(k_l'(d)\right),\ S(v)\right)\xrightarrow[N\to\infty]{}$$

$$\frac{1}{\cancel{N}}\left[\begin{array}{l} \cancel{0\cdot\left(0\cdot U_{\text{ДАН}}\right)\cdot\frac{\left(1\cdot\cancel{N}\right)}{256}} +1\cdot\left(1\cdot U_{\text{ДАН}}\right)\cdot\frac{\left(8\cdot\cancel{N}\right)}{256}+2\cdot\left(2\cdot U_{\text{ДАН}}\right)\cdot\frac{\left(28\cdot\cancel{N}\right)}{256}+3\cdot\left(3\cdot U_{\text{ДАН}}\right)\cdot\frac{\left(56\cdot\cancel{N}\right)}{256}+4\cdot\left(4\cdot U_{\text{ДАН}}\right)\cdot\frac{\left(70\cdot\cancel{N}\right)}{256}+ \\ +5\cdot\left(5\cdot U_{\text{ДАН}}\right)\cdot\frac{\left(56\cdot\cancel{N}\right)}{256}+6\cdot\left(6\cdot U_{\text{ДАН}}\right)\cdot\frac{\left(28\cdot\cancel{N}\right)}{256}+7\cdot\left(7\cdot U_{\text{ДАН}}\right)\cdot\frac{\left(8\cdot\cancel{N}\right)}{256}+8\cdot\left(8\cdot U_{\text{ДАН}}\right)\cdot\frac{\left(1\cdot\cancel{N}\right)}{256} \end{array}\right]-16\cdot U_{\text{ДАН}}= \tag{3.3}$$

$$=\frac{1}{256}\left[8\cdot U_{\text{ДАН}}+112\cdot U_{\text{ДАН}}+504\cdot U_{\text{ДАН}}+1120\cdot U_{\text{ДАН}}+1400\cdot U_{\text{ДАН}}+1008\cdot U_{\text{ДАН}}+392\cdot U_{\text{ДАН}}+64\cdot U_{\text{ДАН}}\right]-16\cdot U_{\text{ДАН}}=$$

$$=\frac{1}{256}\left[4608\cdot U_{\text{ДАН}}\right]-16\cdot U_{\text{ДАН}}=18\cdot U_{\text{ДАН}}-16\cdot U_{\text{ДАН}}=2\cdot U_{\text{ДАН}}$$

,

Однако моделирование показывает, что для шифра «Кузнечик» такой сценарий невозможен – для любого ложного варианта, перебираемого подблока ключа (конечно при шифровании абсолютно случайных сообщений) вероятность совпадения веса Хэмминга выхода S-box рассчитанного на истинном и ложном вариантах перебираемого подблока ключа будет колебаться в диапазоне от 10.1563% до 28.9063%. Таким образом для большинства сообщений будут нарушены сочетания $h_l'\left(i,\, k_l'(d)\right)$ и $HW\left(a(i,\, s)\right)\cdot U_{\text{ДАН}}$ представленные во втором и третьем столбцах таблицы 3.1. В свою очередь моделирование показывает, что из всех 362 880 теоретически возможных сочетаний значений элементов выборок $h_l'\left(i,\, k_l'(d)\right)$ и $HW\left(a(i,\, s)\right)\cdot U_{\text{ДАН}}$, сумма произведений $h_l'\left(i,\, k_l'(d)\right)\cdot HW\left(a(i,\, s)\right)\cdot U_{\text{ДАН}}$, будет максимальной только на тех сочетаниях, которые представлены в таблице 3.1. Это доказывает, что выбор ложного варианта, перебираемого подблока ключа, неминуемо приведёт к уменьшению амплитуды всплеска в соответствующем отсчёте ковариационного вектора. Аналогичные результаты были получены в отношении шифров DES, AES и «Магма».

Для проверки представленных выше теоретических выводов был проведён расчёт ковариационных векторов. Использовались формы сигнала, сгенерированные по описанному в разделе 3.1 алгоритму, при этом вначале полагалось, что шумовая компонента на формах сигнала отсутствует: $\xi(i,\, j)=0$. На рисунках 3.1 представлены полученные ковариационные вектора на истинном (рисунки 3.1а,б), и на наихудшем для злоумышленника ложном (рисунки 3.1б,г) вариантах перебираемого подблока ключа.

Расчёт этих векторов производился следующим образом:

1. Выбирался вариант перебираемого подблока ключа $k_l'(d)=0,\ ...,\ 255$.
2. На выбранном варианте ключа, с использованием задействованных ранее (для генерации форм сигнала) сообщений $m(i)$, $i=1,\ ...,\ N$ рассчитывался выход 16-го S-box, определялся его вес Хэмминга. В результате получался вектор-столбец значений весов Хэмминга: $h_l'\left(i=1\div N,\, k_l'(d)\right)$, состоящий из N элементов.
3. Рассчитывались значения коэффициента ковариации, с использованием элементов вектор-столбца $h_l'\left(i=1\div N,\, k_l'(d)\right)$ и каждого из 350-ти вектор-столбцов $S\left(i=1\div N,\, j\right)$, $j=1,\ ...,\ 350$.
4. Из 350-ти значений $COV\left(h_l'\left(k_l'(d)\right),\, S(j)\right)$, $j=1,\ ...,\ 350$ отрисовывался график ковариационного вектора.

5. Шаги 1-4 повторялись для остальных 255-ти вариантов перебираемого подблока ключа $k_l'(d)$, в итоге было получено 256 ковариационных векторов – один для истинного варианта ключа, а остальные для ложных.

Из рисунков 3.1 видно, что всплеск на ковариационном векторе, рассчитанном на истинном ключе составит 0.4, а на наихудшем ложном 0.08125 (что составляет 20.3% от истинного).

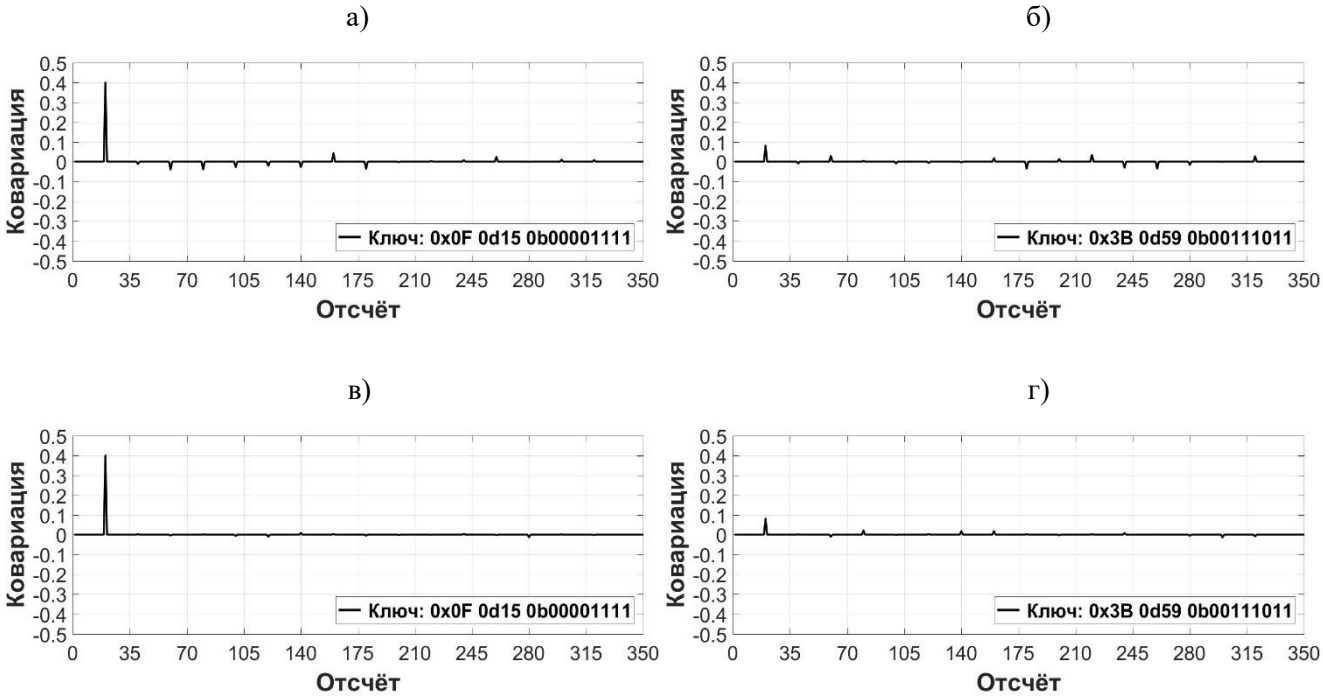
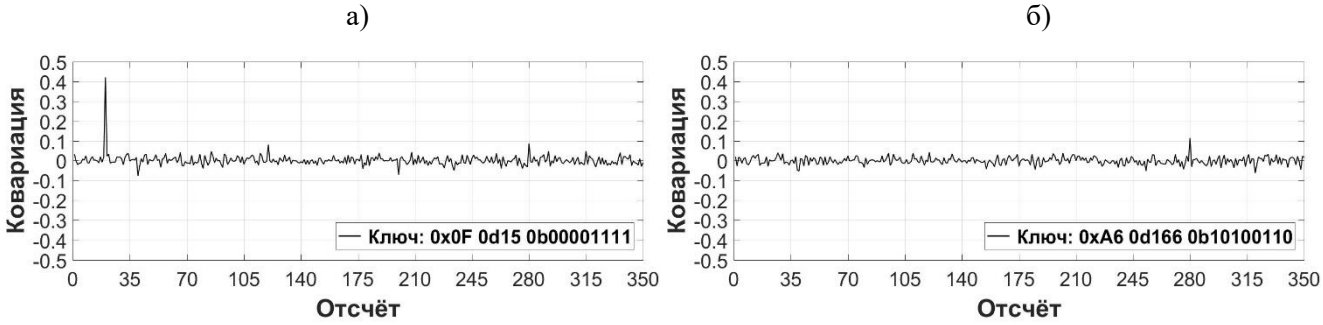


Рисунок 3.1 – Ковариационные вектора, рассчитанные с использованием не зашумлённых форм сигнала, на истинном и наихудшем для злоумышленника ложном вариантах перебираемого подблока ключа при обработке: а), б) 256-ти форм сигнала; в), г) 2 048-ми форм сигнала

Атака с использованием зашумлённых форм сигнала также приводит к однозначному определению истинного варианта подблока ключа (рисунки 3.2).



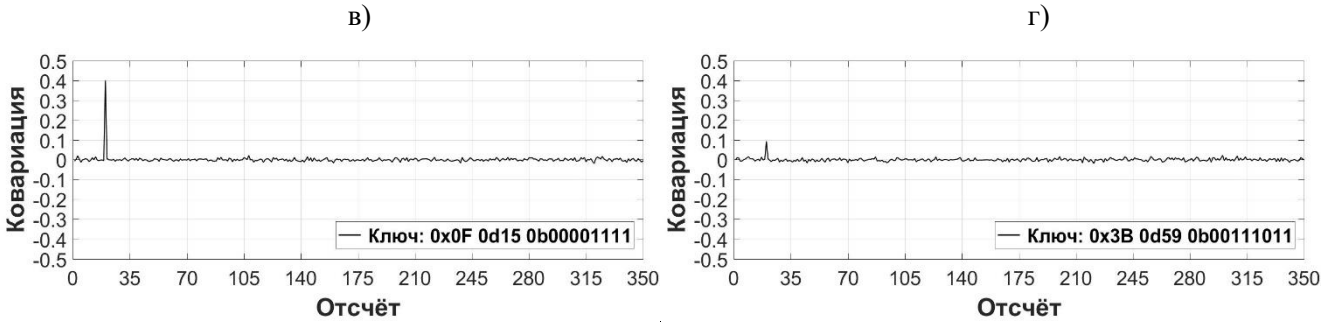


Рисунок 3.2 – Ковариационные вектора рассчитанные, с использованием зашумлённых форм сигнала (СКО шума: 0.2), на истинном и наихудшем для злоумышленника ложном вариантах перебираемого подблока ключа при обработке: а), б) 256-ти форм сигнала; в), г) 2 048-ми форм сигнала

Реализация эксперимента в отношении различного количества зашумлённых форм сигнала (с тремя уровнями зашумлённости: $\sigma_{ШУМ} = 0.1$, $\sigma_{ШУМ} = 0.2$ и $\sigma_{ШУМ} = 0.3$) подтверждает что даже при значительном СКО шумовой компоненты на уровне $\sigma_{ШУМ} = 0.3$ и использовании всего 256-ти форм сигнала подход с расчётом ковариационных векторов позволяет однозначно отличить истинный вариант подблока ключа от ложных (таблица 3.2). В приведённом примере ковариационный вектор, рассчитанный на истинном варианте перебираемого подблока ключа будет иметь более чем в три раза большую амплитуду всплеска, чем на наихудшем ложном.

Таблица 3.2 – Амплитуды максимальных всплесков на ковариационных векторах, рассчитанных на наихудшем для злоумышленника ложном варианте перебираемого подблока ключа (в процентах, по сравнению с амплитудами всплесков, на векторах, рассчитанных на истинном варианте), в зависимости от количества обрабатываемых форм сигнала и СКО шумовой компоненты

	Амплитуды максимальных всплесков, полученных на наихудших ложных вариантах перебираемого подблока ключа, %			
$N \setminus \sigma_{ШУМ}$	0	0.1	0.2	0.3
256	20.3	26.8	27.1	30.5
512	20.3	19.6	20.1	25.1
1 024	20.3	19.9	24.4	23.8
2 048	20.3	20.4	23.2	22.3
4 096	20.3	20.1	20.7	21.1
8 192	20.3	20.8	19.6	20.8
16 384	20.3	19.8	20.2	22.2
32 768	20.3	20.3	20.5	20.5
65 536	20.3	20.3	20.3	20.2

Из таблицы 3.2 можно определить общую тенденцию – независимо от мощности шума, с увеличением количества обрабатываемых форм сигнала, за счёт снижения шумовой компоненты,

разница между амплитудой ковариационного всплеска, полученной на истинном варианте подблока ключа и на наихудшем ложном будет увеличиваться (стремиться к разнице, которая была в отсутствии шума, то есть к 20.3%), таким образом увеличение количества обрабатываемых форм сигнала безусловно облегчит выделение истинного варианта ключа на фоне 255-ти ложных.

Необходимо разъяснить, что физически, зависимость между весом Хэмминга обрабатываемой комбинации и отсчётами на снятых формах сигнала характеризуется значением $U_{\text{ДАН}}$, то есть величиной энергопотребления, на которую увеличивается значение соответствующего отсчёта на снятой форме сигнала при выработке каждой единицы комбинации $a(i,\ s)$. Как показали практические исследования энергопотребления микроконтроллера, представленные во второй главе, каждый такт его работы, характеризуется весьма нетривиальной кривой энергопотребления, пример которой представлен на рисунке 2.15. При этом каждый отсчёт этой кривой характеризуется разным значением $U_{\text{ДАН}}$ и накладываемым на него значением шумовой компоненты. Из формулы (1.16) понятно, что величина всплесков на ковариационном векторе не зависит от постоянной составляющей отсчёта, а помимо самого факта наличия зависимости двух выборок определяется значением $U_{\text{ДАН}}$, в отношении последнего будет наблюдаться прямо пропорциональная зависимость:

$$\begin{aligned} &COV\left(h_l'\left(k_l'(d)\right),\ S(v)\right)\xrightarrow[N\to\infty]{}\frac{1}{N}\sum_{i=1}^{N}\left(h_l'\left(i,\ k_l'(d)\right)\cdot HW\left(a(i,\ s)\right)\cdot U_{\text{ДАН}}\right)-16\cdot U_{\text{ДАН}}= \\ &=U_{\text{ДАН}}\cdot\frac{1}{N}\sum_{i=1}^{N}\left(h_l'\left(i,\ k_l'(d)\right)\cdot HW\left(a(i,\ s)\right)\right)-16\cdot U_{\text{ДАН}}= \\ &=U_{\text{ДАН}}\cdot\left[\frac{1}{N}\sum_{i=1}^{N}\left(h_l'\left(i,\ k_l'(d)\right)\cdot HW\left(a(i,\ s)\right)\right)-16\right] \end{aligned}, \qquad (3.4)$$

На рисунке 3.3 представлен ковариационный вектор, построенный с использованием реальных форм сигнала, полученных во второй главе при записи данных в чистую ячейку ОЗУ. При его сопоставлении с содержимым таблицы Е.1 приложения Е, можно однозначно подтвердить прямо пропорциональную зависимость амплитуды всплесков на ковариационном векторе от значений $U_{\text{ДАН}}$. При этом, видно, что после обработки значительного количества форм сигнала, шумовая компонента практически не вносит никакого вклада в амплитуды ковариационных всплесков.

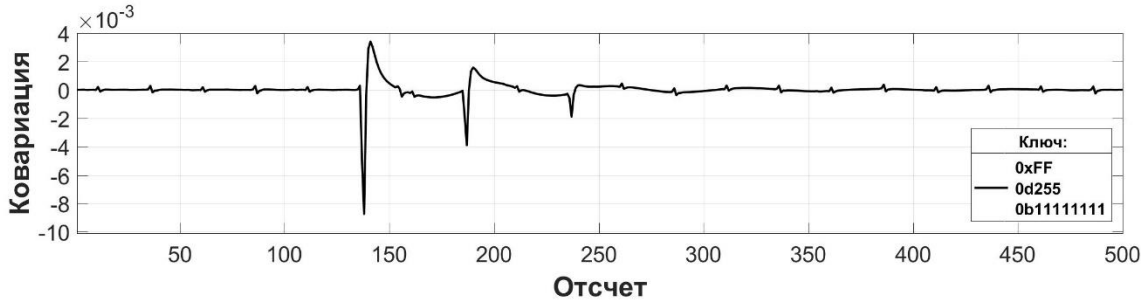


Рисунок 3.3 – Увеличенная область ковариационного вектора, полученного при использовании в качестве рассчитываемого промежуточного значения, комбинации, записываемой в чистую ячейку ОЗУ, при обработке 50 000 форм сигнала, на истинном варианте перебираемого подблока ключа

Рассмотрим два отсчёта на представленном ковариационном векторе: 138-й и 141-й – на первом возникает максимальный отрицательный всплеск ($-8.74\cdot10^{-3}$), на втором – максимальный положительный ($3.41\cdot10^{-3}$). Из таблицы Е.1 приложения Е видно, что значение $U_{\text{ДАН}}$ для 138-го отсчёта составляет 7.155 мВ, а для 141-го – 2.727 мВ. Не составляет труда проверить, что амплитуда всплеска на 138-м отсчёте во столько же раз превышает амплитуду всплеска на 141-м отсчёте во сколько раз различаются значения $U_{\text{ДАН}}$ для этих отсчётов (то есть в 2.6 раза, с незначительной погрешностью, обусловленной неполным устранением шумовой компоненты). Рассмотрим по отдельности всплески, возникающие на двух этих отсчётах. Экспериментально подтверждается, что для всех 256-ти вариантов перебираемого подблока ключа всплеск на 141-м отсчёте имеет наибольшую амплитуду на истинном ключе, этот факт полностью согласуется с приведёнными ранее теоретическими доводами. Аналогичная ситуация наблюдается для 138-го отсчёта – амплитуда всплеска на этом отсчёте равная $-8.74\cdot10^{-3}$ будет максимальной лишь на истинном варианте перебираемого ключа. Однако учитывая, что значение $U_{\text{ДАН}}$ на 138-м отсчёте в 2.6 раза больше чем на 141-м отсчёте, можно сделать вывод, о существовании вероятности того, что всплеск полученный на некоторых ложных вариантах перебираемого подблока ключа на 138-м отсчёте будет превышать амплитуду всплеска на 141-м отсчёте на истинном ключе. Например, на рисунке 3.4 представлен ковариационный вектор полученный на наихудшем для злоумышленника ложном варианте перебираемого ключа «0111 1101», из него видно, что амплитуда всплеска на 138-м отсчёте составляет $7.29\cdot10^{-3}$, что очевидно меньше всплеска на 138-м отсчёте на истинном ключе, но значительно превышает всплеск на 141-м отсчёте. Заметим, что в случае если бы значение $U_{\text{ДАН}}$ на 138-м отсчёте составляло бы ту же величину, что и на 141-м, то всплеск на рассматриваемом ложном ключе не превысил бы значения $2.1\cdot10^{-3}$ (что очевидно, было бы меньше всплеска полученного на истинном ключе на 141-м отсчёте).

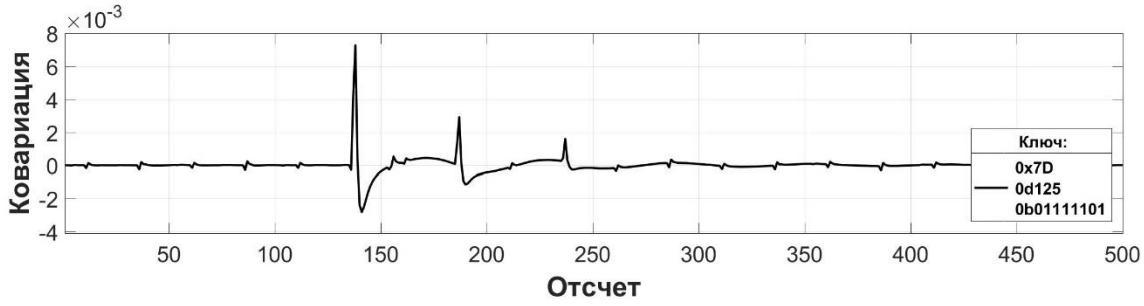


Рисунок 3.4 – Увеличенная область ковариационного вектора, полученного при использовании в качестве рассчитываемого промежуточного значения, комбинации, копируемой в чистую ячейку ОЗУ, при обработке 50 000 форм сигнала, на ложном варианте перебираемого подблока ключа

Данный пример весьма ясно, демонстрирует на практике, такую негативную особенность подхода с расчётом ковариационного вектора, при реализации DPA, как зависимость амплитуд всплесков от размерностей отсчётов форм сигнала.

Необходимо обратить внимание на то, что в приведённом примере наибольший всплеск всё же будет наблюдаться на истинном варианте перебираемого подблока ключа. Рассмотрим этот вопрос подробнее. Представим отсчёты в рассматриваемом окне со 136-го по 436-й отсчёты, как вектор $\vec{v} = \left(\nu\left(i,\, 136\right),\ \nu\left(i,\, 137\right),..,\ \nu\left(i,\, 436\right)\right)$, значение каждого отсчёта определяется энергопотреблением, затрачиваемым на копирование в ОЗУ комбинации $a\left(i,\, s\right)$ с определённым весом Хэмминга: $HW\left(a\left(i,\, s\right)\right)$. При этом каждый отсчёт имеет различное значение $U_{\text{ДАН}}$ – энергопотребления на которое будет увеличиваться (или уменьшаться) значение отсчёта с каждым увеличением на единицу веса Хэмминга обрабатываемой комбинации. В том числе имеется некоторый отсчёт $\nu\left(i,\, l\right)$, который имеет максимальное значение $U_{\text{ДАН}}$ (в приведённом выше примере это 138-й отсчёт).

Очевидно, что при выборе истинного перебираемого подблока ключа значение веса Хэмминга обрабатываемой комбинации, для всех используемых сообщений, будет рассчитано верно. Следовательно, на каждом отсчёте соответствующего ковариационного вектора будет наблюдаться наибольший возможный, для соответствующего значения $U_{\text{ДАН}}$, всплеск, а всплеск на отсчёте $\nu\left(i,\, l\right)$ будет наибольшим по отношению к остальным. Очевидно, что на ложных ключах зависимость между рассчитанными значениями выхода S-box и истинными будет меньше. Следовательно, всплески, возникающие на соответствующих отсчётах будут иметь гарантированно меньшую амплитуду, по сравнению со всплесками на этих же отсчётах на истинном ключе (по крайней мере после обработки количества форм сигнала, достаточного для минимизации шумовой компоненты), следовательно всплеск в отсчёте $\nu\left(i,\, l\right)$ будет иметь

меньшую амплитуду, чем на истинном ключе. Отсюда, несмотря на описанную особенность ковариационных векторов понятно, что однозначное определение истинного варианта перебираемого подблока ключа будет оставаться возможным, по крайней мере, при рассмотрении всплесков на ковариационном векторе в рамках тактов, на которых обрабатывается рассчитываемая комбинация.

Наибольшую опасность для злоумышленника данная особенность ковариационных векторов имеет в областях форм сигнала, соответствующих обработке комбинаций, относящихся к другим частям анализируемого алгоритма защитного преобразования, и при использовании небольшой выборки. При использовании выборки, в несколько десятков или сотен элементов, входы или выходы соседних S-box могут иметь сильную зависимость с выходом анализируемого S-box, рассчитанным на ложном варианте перебираемого подблока ключа. В этом случае всплески на ковариационном векторе в районе таких S-box могут быть весьма значительными. И если значение $U_{\text{ДАН}}$ в отсчётах, относящихся к этим S-box будет превышать величину $U_{\text{ДАН}}$ в отсчётах, относящихся к анализируемому S-box, то эти побочные всплески могут превышать всплески полученные на истинном ключе в районе анализируемого S-box (то есть будут происходить ложные всплески). Ниже более подробно будет описан пример подобного случая.

Анализ преобразований, выполняемых шифром «Кузнечик», позволяет сделать вывод о том, что принципиально возможно возникновение такого события, когда, к примеру, выходы одного или нескольких соседних S-box, будут частично или в точности повторять выходы анализируемого S-box, рассчитанные на ложном варианте перебираемого подблока ключа. Таким образом, если злоумышленник не знает точного момента обработки, рассчитываемой им на перебираемом ключе, комбинации, то у него могут возникнуть трудности в идентификации всплеска, соответствующего истинному варианту перебираемого подблока ключа. Для простоты иллюстрации данного случая возьмём два соседних S-box «Кузнечика»: 15-й и 16-й, и положим, что злоумышленник пытается восстановить восемь младших бит раундового ключа (рисунок 3.5).

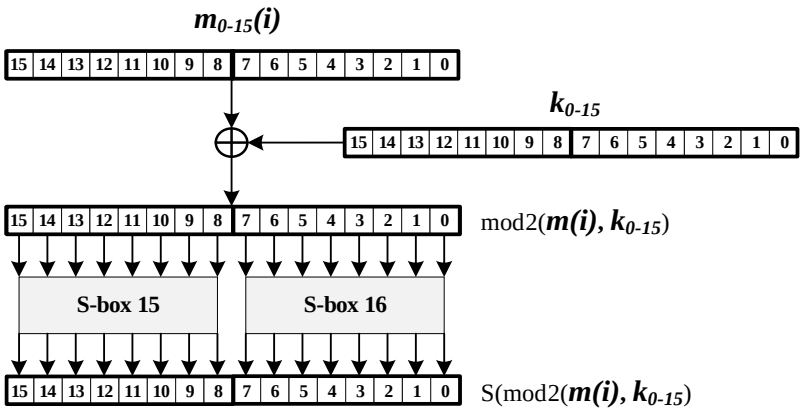


Рисунок 3.5 – Часть первого раунда шифра «Кузнечик»

На первом этапе, используя случайные 128-ми битовые сообщения, злоумышленник многократно инициирует операции шифрования, получая N форм сигнала, характеризующих энергопотребление анализируемого чипа. Является вполне очевидным, что в процессе генерации N случайных сообщений, существует вероятность того, что N раз выпадут такие 8-15 биты сообщения $m_{8-15}(i)$, которые, при сложении по mod2 с 8-15 битами раундового ключа k_{8-15}, дадут такой вход 15-го S-box: $m_{8-15}(i) \oplus k_{8-15}$, который все N раз будет совпадать с входом 16-го S-box, при его расчёте на ложных 0-7 битах перебираемого подблока ключа $k'_{0-7}(d)$:

$$m_{8-15}(i) \oplus k_{8-15} = m_{0-7}(i) \oplus k'_{0-7}(d), \tag{3.5}$$

для всех i, где $i = 1, ..., N$; $k'_{0-7}(d)$ – фиксированный перебираемый подблок ключа первого раунда, отличный от истинного k_{0-7}.

Отсюда вес Хэмминга выхода 16-го S-box, рассчитанный на ложном варианте перебираемого подблока ключа $k'_{0-7}(d)$, для всех N сообщений будет совпадать с весом Хэмминга выхода 15-го S-box, поэтому будет иметь сильную зависимость с отсчётами форм сигнала, соответствующими выработке бит выхода 15-го S-box. Следовательно, в процессе расчёта ковариационного вектора на ложном ключе, в этой области возникнет всплеск, равный всплеску получаемому на векторе, рассчитанном на истинном ключе k_{0-7} в районе 16-го S-box. Конечно данный случай весьма маловероятен, однако даже если формула (3.5) будет выполняться не для всех N сообщений, этот эффект всё равно может значительно затруднить восстановление ключа (тем более в случае обработки небольшого количества зашумлённых форм сигнала). Ситуация значительно усугубится в случае, если значения $U_{\text{ДАН}}$, для отсчётов форм сигнала, относящихся к 15-му S-box, будут больше, значений $U_{\text{ДАН}}$, для отсчётов относящихся к 16-му S-box. Максимально снизить вероятность наступления подобного события очевидно можно увеличением количества обрабатываемых форм сигнала (за счёт того, что 0-7 биты генерируемого сообщения не зависят от 8-15 бит). (Подобный пример можно с лёгкостью экстраполировать на другие преобразования анализируемого алгоритма, ложные всплески могут возникать в отсчётах, относящихся, например, к выполнению операций сложения по mod2, L-преобразования, циклического сдвига и тому подобные)

Приведённые в данном разделе теоретические выкладки и результаты моделирования позволяют систематизировать причины появления побочных всплесков на ковариационных векторах – их всего три:

- Первая причина – сильная зависимость элементов выборки $a'_l\left(i,\ k'_l(d)\right)$ с элементами выборки $S\left(i=1\div N,\ j\neq v\right)$. Эта причина появления побочных всплесков обусловлена как программным алгоритмом, который выполняет чип, так и его конструктивными особенностями. Наибольшие всплески обычно возникают в районе v-го и соседних отсчётов, характеризующих выработку бит выхода рассматриваемого S-box, так как они очевидно неустранимы увеличением количества обрабатываемых форм сигнала (в приведённом на рисунке 3.5 примере видно, что независимо от количества пробеганий сообщением $m_{0-7}(i)$ каждого из 256-ти значений, вероятность совпадения веса Хэмминга рассчитываемой комбинации $a'_l\left(i,\ k'_l(d)\right)$ с весом Хэмминга истинной комбинации $a(i,\ 16)$ останется одинаковой). Всплески в районе отсчётов, характеризующих выработку бит выходов других S-box можно минимизировать – так как биты сообщения, а, следовательно, и входов/выходов разных S-box не зависят друг от друга (в частности $m_{8-15}(i)$ не будет зависеть от $m_{0-7}(i)$, поэтому при использовании больших выборок зависимость веса Хэмминга комбинации $a'_l\left(i,\ k'_l(d)\right)$ с любыми другими комбинациями будет сходить на нет.
- Вторая причина – сильная зашумлённость форм сигнала. Наибольшим образом проявляется на небольших выборках, в этом случае шумовая компонента оказывает значительное влияние на отсчёты ковариационного вектора, однако с увеличением количества обрабатываемых форм сигнала такие всплески быстро сходят на нет.
- Третья причина – отличие значений $U_{\text{ДАН}}$ для разных отсчётов форм сигнала. Напрямую, увеличением количества обрабатываемых форм сигнала, их снизить невозможно. Однако отличие значений $U_{\text{ДАН}}$ для разных отсчётов проявляются в виде значительных всплесков лишь в случае существования зависимостей между рассчитываемой комбинацией и отсчётами форм сигнала. Из формулы (3.4) видно, что значение $U_{\text{ДАН}}$ оказывает влияние на значение ковариации лишь в случае наличия какой-либо зависимости между двумя выборками (большое значение $U_{\text{ДАН}}$ может лишь «подтянуть» существующий всплеск на ковариационном векторе), таким образом, например в два раза большее значение $U_{\text{ДАН}}$ – увеличивает амплитуду корреляционного всплеска в два раза. Однако если зависимость между выборками отсутствует, то даже весьма большие значения $U_{\text{ДАН}}$ не приведут к появлению сколько-нибудь значительных всплесков. Учитывая тот факт, что зависимости рассчитываемого промежуточного значения от отсчётов, отличных от рассматриваемых

сходят на нет с увеличением количества обрабатываемых форм сигнала, то данную причину возникновения побочных всплесков также можно считать устранимой.

По итогам описанного в текущем разделе можно сделать следующие основные выводы:

- Реализация DPA с расчётом ковариационных векторов принципиально возможна, не только теоретически, но и с использованием форм сигнала, полученных в эксперименте, от реального чипа.
- Негативной особенностью подхода с расчётом ковариационных векторов является зависимость значений последних не только от наличия самого факта зависимости между рассчитываемыми и обрабатываемыми данными, но и от величины «утечки» по цепи электропитания (от значений $U_{\text{ДАН}}$). Влияние этой особенности на успешность реализации DPA минимизируется увеличением количества обрабатываемых форм сигнала, однако на практике эта особенность может привести к необходимости обработки чрезвычайно большого количества форм сигнала. Более подробно этот вопрос будет рассмотрен, после приведённого далее описания особенностей подхода к реализации DPA с расчётом корреляционных векторов.

3.3 Особенности реализации DPA с расчётом корреляционных векторов

В подразделе 1.7.2 пояснялось, что расчёт корреляционного вектора представляет собой нормировку значений ковариационного, что позволяет привести все отсчёты ковариационного вектора к единой размерности. В итоге устраняется зависимость ковариации от значений $U_{\text{ДАН}}$.

Приведём пример расчёта коэффициентов ковариации и корреляции для отсчётов, имеющих разные значения $U_{\text{ДАН}}$. Положим существование на форме сигнала двух отсчётов:

- отсчёта v характеризующего энергопотребление, затрачиваемое чипом на выработку выхода 16-го S-box $a(i,\ 16)$. Его значение будет прямо пропорционально зависеть от веса Хэмминга этой комбинации. Значение $h'_l\left(i,\ k'_l(d)\right)$ в свою очередь рассчитывается злоумышленником на известном i-м сообщении и варианте перебираемого подблока ключа $k'_l(d)$.
- отсчёта $t \neq v$. характеризующего энергопотребление чипа в процессе выработки другой комбинации $e(i)$. При этом на некоторой выборке, значение её веса Хэмминга $h'\left(e(i)\right)$

будет иметь зависимость с $h'_l\left(k'_l(d)\right)$, рассчитанным на некотором ложном варианте перебираемого подблока ключа.

Если принять что на выработку каждой единицы комбинации выхода рассматриваемого 16-го S-box будет затрачиваться энергия равная $U_{\text{ДАН}} = 0.2$. То на истинном варианте перебираемого подблока ключа всплеск на ковариационном векторе в отсчёте v составит 0.4 (формула (3.3) и рисунок 3.1а).

Всплеск на корреляционном векторе для того же отсчёта ожидаемо составит единицу:

$$c\left(k'_l(d),\ t\right) = \frac{COV\left(h'_l\left(k'_l(d)\right),\ S(t)\right)}{\sigma\left(h'_l\left(k'_l(d)\right)\right)\cdot\sigma\left(S(t)\right)} \approx \frac{0.4}{\sqrt{2}\cdot\sqrt{0.08}} = 1.00, \tag{3.6}$$

Положим, что при тех же параметрах энергопотребления, наличие зависимости между $h'\left(e(i)\right)$ и $h'_l\left(k'_l(d)\right)$ (рассчитанном на некотором ложном варианте перебираемого подблока ключа) приводит к тому, что в отсчёте t на ковариационном векторе возникнет всплеск равный 0.1 (25% от истинного). В этом случае отличить истинный вариант подблока ключа от ложного с использованием ковариационного вектора не составит труда. Всплеск на корреляционном векторе, также ожидаемо будет иметь в четыре раза меньшую амплитуду:

$$c\left(k'_l(d),\ t\right) = \frac{COV\left(h'_l\left(k'_l(d)\right),\ S(t)\right)}{\sigma\left(h'_l\left(k'_l(d)\right)\right)\cdot\sigma\left(S(t)\right)} \approx \frac{0.1}{\sqrt{2}\cdot\sqrt{0.08}} = 0.25, \tag{3.7}$$

Однако если предположить, что на выработку бит комбинации $e(i)$ требуется больше энергии, чем на выработку бит комбинации $a(i,\ 16)$, допустим в пять раз больше, то размерность отсчёта t окажется больше размерности отсчёта v (его значения будут изменяться в больших пределах). Отсюда всплеск на ковариационном векторе в районе отсчёта t также линейно возрастёт в пять раз до 0.5, то есть он превысит всплеск на истинном ключе в районе отсчёта v. В этом случае отличить истинный подблок ключа от ложного окажется невозможно.

При расчёте всплеска на корреляционном векторе необходимо учесть, что значение ковариации в числителе возрастёт в пять раз, но вместе с этим в пять раз увеличится и СКО отсчёта v в знаменателе:

$$\sigma\left(S(v)\right) = \sqrt{2}, \tag{3.8}$$

Поэтому амплитуда корреляционного всплеска не изменится:

$$c\left(k_l'(d),\ t\right)=\frac{COV\left(h_l'\left(k_l'(d)\right),\ S(t)\right)}{\sigma\left(h_l'\left(k_l'(d)\right)\right)\cdot\sigma\left(S(t)\right)}\approx\frac{0.5}{\sqrt{2}\cdot\sqrt{2}}=0.25, \tag{3.9}$$

На рисунках 3.6 представлены корреляционные вектора, полученные путём моделирования на истинном и наихудшем для злоумышленника ложном вариантах перебираемого подблока ключа с использованием не зашумлённых форм сигнала. Из них видно, что в случае отсутствия шума графики будут в точности повторять ковариационные.

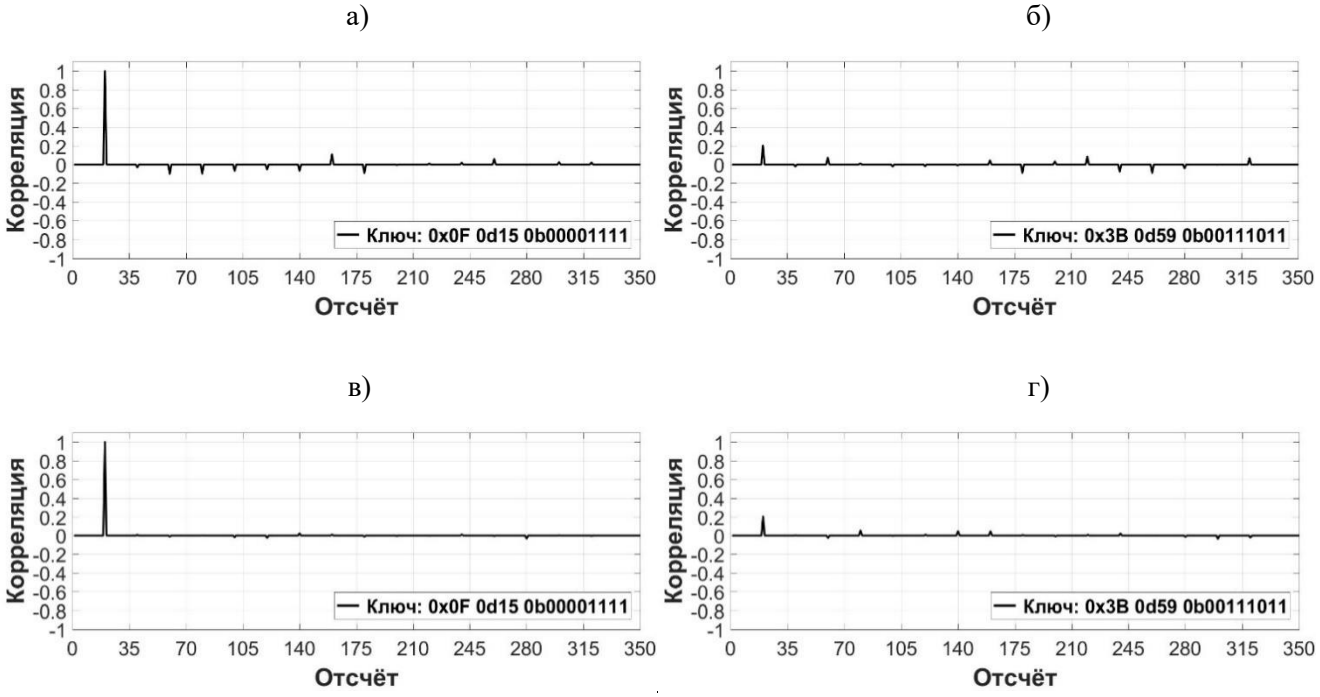
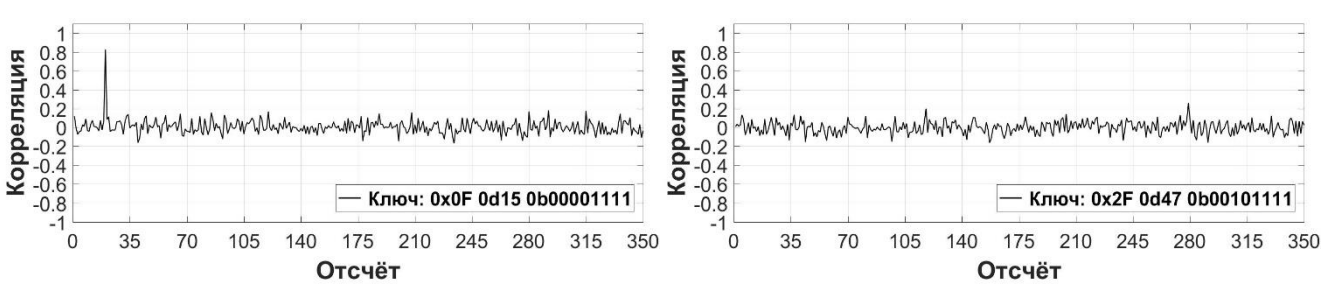


Рисунок 3.6 – Корреляционные вектора, рассчитанные с использованием не зашумлённых форм сигнала на истинном и наихудшем для злоумышленника ложном вариантах перебираемого подблока ключа при обработке: а), б) 256-ти форм сигнала; в), г) 2 048-ми форм сигнала

На рисунках 3.7 представлены корреляционные вектора, рассчитанные на истинном и наихудшем для злоумышленника ложном вариантах перебираемого подблока ключа, для $\sigma_{ШУМ}=0.2$, с использованием соответственно: 256-ти, 2 048-ми, и 65 536-ти форм сигнала.



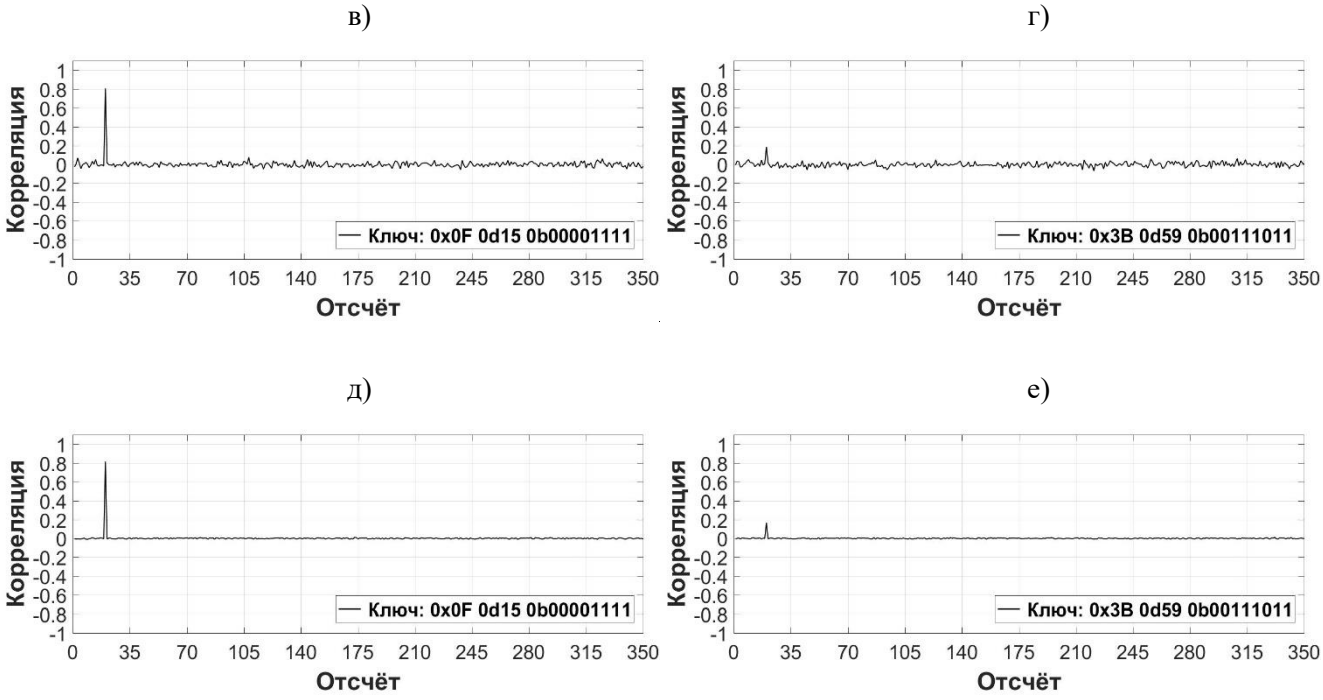


Рисунок 3.7 – Корреляционные вектора, рассчитанные на истинном и наихудшем для злоумышленника ложном вариантах перебираемого подблока ключа, с использованием зашумлённых форм сигнала (СКО шума: 0.2) при обработке: а), б) 256-ти форм сигнала; в), г) 2 048-ми форм сигнала; д), е) 65 536-ти форм сигнала

Из рисунков 3.7 видна, важная особенность при расчёте корреляционных векторов с использованием зашумлённых форм сигнала, не представленная в известной литературе – в случае наличия шума на снятых формах сигнала, амплитуды корреляционных всплесков будут занижаться. При этом на амплитуду всплеска шумовая компонента будет оказывать влияние независимо от количества обрабатываемых форм сигнала, ниже будут показаны причины этого явления.

Коэффициент корреляции для v-го отсчёта рассчитывается как частное от деления значения ковариации на произведение СКО соответствующих выборок $h'_l\left(k'_l(d)\right)$ и $S(j)$. Ранее было показано, что при использовании достаточно больших выборок, шумовая компонента на значение ковариации не будет оказывать практически никакого влияния, в то же время СКО выборки $h'_l\left(k'_l(d)\right)$ не зависит ни от шума ни от перебираемого варианта подблока ключа и при использовании случайных сообщений будет являться константой: $\sigma\left(h'_l\left(k'_l(d)\right)\right)=\sqrt{2}$. Однако СКО выборки $S(j)$ от шума очевидно зависит, его можно описать классической формулой, которая в дальнейшем, основываясь на правилах сложения и возведения в степень, может быть разложена на составляющие:

$$\sigma\left(S\left(v\right)\right)=\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(S\left(i,\ v\right)-\overline{S\left(v\right)}\right)^{2}}=\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left[\left(S\left(i,\ v\right)\right)^{2}-2\cdot S\left(i,\ v\right)\cdot\overline{S\left(v\right)}+\overline{S\left(v\right)}^{2}\right]}$$
$$=\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(S\left(i,\ v\right)\right)^{2}-\frac{1}{N}\sum_{i=1}^{N}2\cdot S\left(i,\ v\right)\cdot\overline{S\left(v\right)}+\frac{1}{N}\sum_{i=1}^{N}\overline{S\left(v\right)}^{2}}, \quad (3.10)$$

Очевидно, что значение $\overline{S\left(v\right)}$ при $N\to\infty$, будет являться константой так как шумовая компонента минимизируется:

$$\overline{S\left(v\right)}=\frac{1}{N}\sum_{i=1}^{N}S\left(i,\ v\right)=\frac{1}{N}\sum_{i=1}^{N}\left[h\left(i\right)\cdot U_{\text{ДАН}}+\xi\left(i,\ v\right)\right]=\frac{1}{N}\sum_{i=1}^{N}h\left(i\right)\cdot U_{\text{ДАН}}+\frac{1}{N}\sum_{i=1}^{N}\xi\left(i,\ v\right)\xrightarrow[N\to\infty]{}$$
$$\xrightarrow[N\to\infty]{}4\cdot U_{\text{ДАН}}+\cancel{\frac{1}{N}\sum_{i=1}^{N}\xi\left(i,\ v\right)}, \quad (3.11)$$

где $h\left(i\right)$ – вес Хэмминга комбинации выхода S-box; $U_{\text{ДАН}}$ – энергопотребление, затрачиваемое на выработку каждой единицы выхода S-box.

Отсюда, для больших N формулу (3.10) можно разложить в:

$$\sigma\left(S\left(v\right)\right)\approx\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(S\left(i,\ v\right)\right)^{2}-2\cdot4\cdot U_{\text{ДАН}}\cdot\frac{1}{N}\sum_{i=1}^{N}S\left(i,\ v\right)+\cancel{\frac{1}{N}}\cdot\cancel{N}\cdot\left(4\cdot U_{\text{ДАН}}\right)^{2}}=$$
$$=\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(S\left(i,\ v\right)\right)^{2}-2\cdot4\cdot U_{\text{ДАН}}\cdot4\cdot U_{\text{ДАН}}+16\cdot U_{\text{ДАН}}^{\ 2}}=\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(S\left(i,\ v\right)\right)^{2}-16\cdot U_{\text{ДАН}}^{\ 2}}, \quad (3.12)$$

В формуле (3.12) раскроем левую часть разности:

$$\frac{1}{N}\sum_{i=1}^{N}\left(S\left(i,\ v\right)\right)^{2}=\frac{1}{N}\sum_{i=1}^{N}\left[h\left(i\right)\cdot U_{\text{ДАН}}+\xi\left(i,\ v\right)\right]^{2}=$$
$$=\frac{1}{N}\sum_{i=1}^{N}\left(h\left(i\right)\cdot U_{\text{ДАН}}\right)^{2}+\frac{1}{N}\sum_{i=1}^{N}\left[2\cdot\left(h\left(i\right)\cdot U_{\text{ДАН}}\right)\cdot\xi\left(i,\ v\right)\right]+\frac{1}{N}\sum_{i=1}^{N}\left(\xi\left(i,\ v\right)\right)^{2}, \quad (3.13)$$

Крайний правый аргумент формулы (3.13) является дисперсией шумовой компоненты:

$\frac{1}{N}\sum_{i=1}^{N}\left(\xi\left(i,\ v\right)\right)^{2}$ – дисперсия шума, классическая формула расчёта дисперсии выглядит как $\frac{1}{N}\sum_{i=1}^{N}\left(\xi\left(i,\ v\right)-\overline{\xi\left(i,\ v\right)}\right)^{2}$, где $\overline{\xi\left(i,\ v\right)}$ – математическое ожидание, как известно по условию, для шумовой компоненты оно равно нулю.

Теперь рассмотрим средний аргумент суммы в формуле (3.13) – известно, что математическое ожидание произведения двух случайных величин равно произведению их

математических ожиданий плюс значение их ковариации. Поэтому будет справедливо следующее преобразование:

$$\begin{aligned}&\frac{1}{N}\sum_{i=1}^{N}\left[2\cdot\left(h(i)\cdot U_{\text{ДАН}}\right)\cdot\xi(i,\ v)\right]=\\&=2\cdot\frac{1}{N}\sum_{i=1}^{N}\left(h(i)\cdot U_{\text{ДАН}}\right)\cdot\frac{1}{N}\sum_{i=1}^{N}\xi(i,\ v)+COV\left(\left(h(i=1\div N)\cdot U_{\text{ДАН}}\right),\ \xi(i=1\div N,\ v)\right)\end{aligned}, \quad (3.14)$$

При больших N (3.14) преобразуется в:

$$\begin{aligned}&\frac{1}{N}\sum_{i=1}^{N}\left[2\cdot\left(h(i)\cdot U_{\text{ДАН}}\right)\cdot\xi(i,\ v)\right]=\\&=2\cdot 4\cdot U_{\text{ДАН}}\cdot\cancel{\frac{1}{N}\sum_{i=1}^{N}\xi(i,\ v)}+COV\left(\left(h(i=1\div N)\cdot U_{\text{ДАН}}\right),\ \xi(i=1\div N,\ v)\right)\end{aligned}, \quad (3.15)$$

По той причине, что зависимость между весом Хэмминга выхода S-box и шумовой компонентой, накладываемой на v-й отсчёт, будет практически отсутствовать: $COV\left(\left(h(i=1\div N)\cdot U_{\text{ДАН}}\right),\ \xi(i=1\div N,\ v)\right)\xrightarrow[N\to\infty]{}0$, то формулу (3.12) можно преобразовать к следующему виду:

$$\sigma\left(S(v)\right)\xrightarrow[N\to\infty]{}\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(S(i,\ v)\right)^2-16\cdot {U_{\text{ДАН}}}^2}\approx\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(h(i)\cdot U_{\text{ДАН}}\right)^2+\sigma\left(\xi(v)\right)^2-16\cdot {U_{\text{ДАН}}}^2}, \quad (3.16)$$

где $\sigma\left(\xi(v)\right)^2$ – дисперсия шумовой компоненты, накладывающейся на значения энергопотребления.

Отсюда, если к примеру $U_{\text{ДАН}}=0.2$, $\sigma\left(\xi(v)\right)=0.2\Rightarrow\sigma\left(\xi(v)\right)^2=0.04$, то

$$\sigma\left(S(v)\right)\approx\sqrt{0.72+0.04-0.64}\approx 0.3464, \quad (3.17)$$

В отсутствии шума получится:

$$\sigma\left(S(v)\right)\approx\sqrt{0.72+0-0.64}\approx 0.2828, \quad (3.18)$$

Ещё раз подчеркнём, что в числителе формулы (1.7) шумовая компонента сходит на нет с увеличением количества обрабатываемых форм сигнала, и в этом случае отличие коэффициента ковариации, рассчитанного с использованием зашумлённых и не зашумлённых форм сигнала, устремится к нулю. Левая часть знаменателя формулы (1.7) (СКО рассчитываемого веса Хэмминга выхода S-box) в принципе не зависит от шумовой компоненты. Однако в правой части

знаменателя, при расчёте значения $\sigma\left(S\left(v\right)\right)$, шумовая составляющая будет вносить свой вклад в увеличение СКО соответствующего отсчёта (возможно весьма существенный) даже после обработки неограниченно большого количества форм сигнала.

Из формулы (3.16) видно, что значение $\sigma\left(S\left(v\right)\right)$, определяется двумя параметрами:

- энергопотреблением чипа затрачиваемым на выработку каждой единицы рассчитываемого промежуточного значения: $U_{\text{ДАН}}$;
- дисперсией шумовой компоненты: $\sigma\left(\xi\left(v\right)\right)^2$.

И если первый параметр не зависит от шумовой компоненты, то второй зависит и принципиально не устраним увеличением количества обрабатываемых форм сигнала. Также очевидно, что влияние шумовой компоненты на результат расчёта значения $\sigma\left(S\left(v\right)\right)$ определяется соотношением сигнал/шум – чем больше это соотношение, тем меньший вклад вносит шумовая компонента в результат расчёта по (3.16), в пределе будет справедливо следующее:

$$\sigma\left(S\left(v\right)\right)\xrightarrow[N\to\infty]{}\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(h\left(i\right)\cdot U_{\text{ДАН}}\right)^2+\sigma\left(\xi\left(v\right)\right)^2-16\cdot {U_{\text{ДАН}}}^2}\xrightarrow[SNR\to\infty]{}$$
$$\xrightarrow[SNR\to\infty]{}\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(h\left(i\right)\cdot U_{\text{ДАН}}\right)^2-16\cdot {U_{\text{ДАН}}}^2}, \qquad (3.19)$$

Из (3.19) видно, что увеличением соотношения сигнал/шум можно добиться значений $\sigma\left(S\left(v\right)\right)$ приближенных к получаемым в отсутствии шумовой компоненты. В этом случае коэффициент корреляции, рассчитанный с использованием зашумлённых форм сигнала, будет практически повторять полученный, при использовании не зашумлённых форм сигнала. В противном случае шумовая компонента может оказывать существенное влияние на расчёт значения $\sigma\left(S\left(v\right)\right)$, что очевидно приведёт к уменьшению амплитуды корреляционного всплеска.

Отсюда понятно, что в случае наличия даже весьма небольшой зашумлённости на формах сигнала, коэффициент корреляции, рассчитанный на истинном перебираемом подблоке ключа, никогда не будет давать всплеск равный единице – он всегда будет ниже за счёт увеличенного значения СКО рассматриваемого отсчёта. Для представленных в формулах (3.17), (3.18) соотношений СКО шума и полезной составляющей сигнала, шумовая компонента будет оказывать весьма значительное влияние на результат расчёта значения $\sigma\left(S\left(v\right)\right)$ – всплеск уменьшится на 18% (с единицы, в отсутствие шума, до 0.82). При этом необходимо подчеркнуть, что СКО v-го отсчёта не зависит от варианта перебираемого подблока ключа, поэтому

коэффициент корреляции, рассчитанный с использованием ложных вариантов подблока ключа, также пропорционально уменьшится.

Однако, как было получено ранее на примере ковариационных векторов, снятые на реальном чипе формы сигнала не ограничиваются одним отсчётом – v-ый отсчёт будут окружать другие отсчёты, параметры (в частности СКО) которых могут значительно отличаться. При использовании форм сигнала, снятых на реальном устройстве, корреляционные/ковариационные всплески в районе этих отсчётов, порой являются весьма значительными. Это серьёзно усложняет выделение вектора, рассчитанного на истинном варианте перебираемого подблока ключа, поэтому важной задачей является максимальное снижение амплитуд этих паразитных всплесков. Ранее утверждалось, что деление значений ковариационного вектора на значения СКО соответствующих выборок, позволяет нормировать значения первой что может помочь увеличить вероятность правильного определения истинного варианта подблока ключа. В отсутствии шума данное утверждение справедливо, однако при наличии шума, как показано выше, ситуация не будет столь однозначной.

Исходя из формул (3.16), (3.19) обнаруживается, что эффект уменьшения корреляционных всплесков при наличии шума для каждого отсчёта корреляционного вектора определяется соотношением сигнал/шум и чем это соотношение меньше, тем больше проявляется этот эффект. Отсюда следует, что нормировка коэффициента ковариации, благодаря которой рассчитывается коэффициент корреляции, не исключает полностью зависимость последнего от значения $U_{\text{ДАН}}$.

Рассмотрим подробнее проявление данного эффекта и его влияние на успешность реализации DPA, но вначале заметим, что СКО любого j-го отсчёта в описанной модели определяется двумя составляющими: полезной и шумовой. Как показали приведённые во второй главе результаты анализа энергопотребления чипа, значения как шумовой, так и полезной составляющих сигнала значительно изменяются даже в рамках одного такта при рассмотрении одной операции, а при рассмотрении различных операций отличия могут составлять один и даже два порядка. Для упрощения приводимого примера положим, что шумовая составляющая для всех отсчётов будет приблизительно одинаковой, отличается лишь полезная составляющая для разных отсчётов.

Ранее был приведён пример (формулы (3.6)-(3.9)), в котором, в отсутствие шума, корреляционный всплеск в районе v-го отсчёта на истинном варианте перебираемого подблока ключа составлял 1, а в районе отсчёта t, на некотором ложном варианте ключа – 0.25 (то есть в четыре раза меньше). И такое соотношение будет наблюдаться не зависимо от того какое значение имеет $U_{\text{ДАН}}$ на отсчёте t. В этом и заключается главное (по крайней мере теоретически) достоинство корреляционной модели по сравнению с ковариационной в контексте реализации

атаки DPA. Однако, анализ формул (3.16)-(3.19) позволяет сделать вывод, что при обработке зашумлённых форм сигнала, степень уменьшения корреляционных всплесков будет зависеть от соотношения сигнал/шум рассматриваемого отсчёта. То есть если разные отсчёты формы сигнала имеют отличную размерность полезной составляющей (при фиксированной шумовой компоненте), то степень проявления эффекта для них будет различной. Выше было показано, что корреляционный всплеск на отсчёте v, при $U_{\text{ДАН}} = 0.2$, $\sigma\left(\xi(v)\right)^2 = 0.04$ снизится на 18% (даже при обработке неограниченно большого количества форм сигнала). При этом, если значение $U_{\text{ДАН}}$ на отсчётах t и v будет одинаковым (то есть на выработку каждой единицы комбинаций $a(i,\ 16)$ и $e(i)$ потребуется одинаковое количество энергии: $U_{\text{ДАН}} = 0.2$), тогда очевидно, что СКО выборок $h'_l\left(k'_l(d)\right)$ и $S(t)$ будут такими же как и для отсчёта v:

$$\sigma\left(h'_l\left(k'_l(d)\right)\right) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(h'_l\left(i,\ k'_l(d)\right) - \overline{h'_l\left(k'_l(d)\right)}\right)^2} \approx \sqrt{2}, \tag{3.20}$$

$$\begin{aligned}\sigma\left(S(v)\right) = \sigma\left(S(t)\right) &\approx \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(h(i)\cdot U_{\text{ДАН}}\right)^2 + \sigma\left(\xi(t)\right)^2 - 16\cdot {U_{\text{ДАН}}}^2} = \\ &= \sqrt{0.72 + 0.04 - 0.64} = \sqrt{0.08 + 0.04} \approx 0.3464\end{aligned}, \tag{3.21}$$

Отличия очевидно будут лишь в коэффициенте ковариации, который в первом случае составит 0.4, а во втором 0.1. Отсюда для отсчёта v корреляционный всплеск на истинном варианте перебираемого подблока ключа составит:

$$c\left(k'_l(d),\ v\right) = \frac{COV\left(h'_l\left(k'_l(d)\right),\ S(v)\right)}{\sigma\left(h'_l\left(k'_l(d)\right)\right)\cdot\sigma\left(S(v)\right)} \approx \frac{0.4}{1.4142\cdot 0.3464} \approx 0.8165, \tag{3.22}$$

А амплитуда корреляционного всплеска на отсчёте t, на некотором ложном варианте перебираемого подблока ключа, будет в четыре раза меньше, чем всплеск на истинном ключе в районе отсчёта v (также как и для не зашумлённых форм сигнала):

$$c\left(k'_l(d),\ t\right) = \frac{COV\left(h'_l\left(k'_l(d)\right),\ S(t)\right)}{\sigma\left(h'_l\left(k'_l(d)\right)\right)\cdot\sigma\left(S(t)\right)} \approx \frac{0.1}{1.4142\cdot 0.3464} \approx 0.2041, \tag{3.23}$$

То есть если значение $U_{\text{ДАН}}$ для рассматриваемых отсчётов форм сигнала будет одинаковым, то использование зашумлённых форм сигнала не повлияет на успешность реализации атаки (все отсчёты корреляционного вектора уменьшат свои амплитуды в

одинаковой пропорции). В противном случае, если к примеру значение $U_{\text{ДАН}}$ для отсчёта t окажется больше чем для отсчёта v (например в пять раз), при том, что накладывающаяся шумовая компонента останется прежней, то соотношение сигнал/шум в отсчёте t изменится (станет больше). Тогда очевидно, что дисперсия шума станет оказывать значительно меньшее влияние на результат расчёта значения $\sigma\left(S\left(t\right)\right)$ по формуле (3.19):

$$\sigma\left(S\left(t\right)\right) \approx \sqrt{18+0.04-16} = \sqrt{2+0.04} \approx 1.4283, \tag{3.24}$$

Отсюда влияние шумовой компоненты на корреляционный всплеск будет практически отсутствовать (формула (3.24) в сравнении с (3.8), (3.21)), всплеск окажется не на 18%, а лишь на 0.8% ниже всплеска, получаемого при использовании не зашумлённых форм сигнала:

$$c\left(k_l'\left(d\right),\ t\right) = \frac{COV\left(h_l'\left(k_l'\left(d\right)\right),\ S\left(t\right)\right)}{\sigma\left(h_l'\left(k_l'\left(d\right)\right)\right)\cdot\sigma\left(S\left(t\right)\right)} = \frac{0.5}{1.4142\cdot 1.4283} = 0.248, \tag{3.25}$$

Не сложно проверить, что при дальнейшем увеличении соотношения сигнал/шум шумовая компонента будет оказывать всё меньшее влияние на СКО отсчёта t и корреляционный всплеск оказывается приближенным к всплеску на формах сигнала с отсутствующей шумовой компонентой (то есть к 0.25):

Из формулы (3.19) понятно, что выше этого значения, повышением соотношения сигнал/шум корреляционный всплеск увеличить нельзя. Этот вывод можно безусловно считать положительным для злоумышленника. Однако из этой же формулы следует, что уменьшением соотношения сигнал/шум корреляционный всплеск может быть уменьшен во сколько угодно раз. Поэтому вполне реальным может оказаться случай, когда на отсчёте v, соотношение сигнал/шум окажется очень мало (таким образом всплеск на истинном варианте ключа уменьшится к примеру в пять раз: с единицы до 0.2). А на некотором отсчёте t, соотношение сигнал/шум будет весьма значительным, таким образом, что, эффект уменьшения корреляционного всплеска на нём практически не проявится, и всплеск на ложном варианте ключа составит, как в примере выше: 0.24. В данном случае очевидно произойдёт ошибка восстановления перебираемого ключа.

Приведённые выше примеры подтверждают, что при одинаковой степени зашумлённости всех отсчётов всех собранных форм сигнала, получается, что чем большее значение имеет $U_{\text{ДАН}}$ на рассматриваемом отсчёте, тем в меньшей степени произойдёт снижение соответствующего всплеска на корреляционном векторе. Справедливо и обратное – чем меньшее значение имеет $U_{\text{ДАН}}$, тем сильнее будет занижен соответствующий всплеск на корреляционном векторе. Этот эффект будет оказывать влияние, аналогичное, описанному ранее, эффекту зависимости

коэффициента ковариации от значения $U_{ДАН}$, хоть и в значительно меньшей степени. Из чего можно сделать вывод, что (при использовании зашумлённых форм сигнала), нормировкой коэффициента ковариации не получится полностью исключить зависимость его значения от размерностей отсчётов форм сигнала. Вместе с этим необходимо учесть, что на всплески корреляционных векторов оказывает влияние не только значение $U_{ДАН}$, но и шумовой компоненты. Поэтому вводимые данным эффектом искажения, на практике будут носить несколько более сложный характер, по сравнению со случаем расчёта ковариационных векторов.

Отсюда можно построить следующую логическую цепочку:

- Корреляция от ковариации отличается наличием нормировки, позволяющей перенести амплитуды ковариационных всплесков в диапазон от -1 до 1, исключив (теоретически) зависимость от размерностей анализируемых выборок.
- Однако при использовании зашумлённых форм сигнала (то есть выборок, представляющих собой отсчёты, состоящие из полезной и шумовой компонент), будет наблюдаться эффект уменьшения корреляционных всплесков, таким образом даже при наличии абсолютной зависимости выборок, коэффициент корреляции никогда не сможет быть равен единице (в отличие от случая использования не зашумлённых форм сигнала). Причина появления данного эффекта заложена в самой процедуре нормировки и принципиально неустранима (в том числе увеличением количества обрабатываемых форм сигнала).
- Каждая снятая в процессе атаки форма сигнала состоит из множества отсчётов, лишь небольшая часть которых (в идеальной модели – один) будет характеризовать энергопотребление чипа при выполнении интересующей злоумышленника операции. Б**о**льшая часть отсчётов, будет относиться к выполнению других операций. Логично предположить, что соотношение сигнал/шум на этих отсчётах может отличаться от значений SNR на отсчётах, интересующих злоумышленника (данное предположение было подтверждено на практике, и описано во второй главе – отличия могут быть весьма значительными). Следовательно, эффект уменьшения корреляционных всплесков будет оказывать разное влияние на отсчёты с разным соотношением сигнал/шум. Чем меньше соотношение сигнал/шум тем существеннее будут занижены всплески на корреляционном векторе в соответствующих отсчётах. Получается, что этот эффект несколько нивелирует преимущество использования корреляционных векторов над ковариационными (в некоторой степени компенсирует нормировку, выполняемую при расчёте коэффициента корреляции через ковариацию). Отсюда получается, что при расчёте коэффициента корреляции зависимость его значения от объёма утекающей по цепи электропитания информации сокращается, но не исключается полностью, что может стать причиной

появления ложных всплесков на корреляционных векторах, по аналогии с рассмотренными ранее случаями для ковариационных векторов.

Очевидно, что не имеет смысла производить сравнение эффективности реализации DPA с расчётом ковариационных и корреляционных векторов, аналитически или с использованием описанной в разделе 3.1 упрощённой модели. Получаемые на практике формы сигнала характеризуют весьма сложную зависимость энергопотребления от обрабатываемых данных (в том числе шумовую компоненту, значения которой сильно варьируются даже в рамках одного такта). Поэтому для сравнения реальной эффективности обоих подходов необходимо произвести их применение к полученным на практике формам сигнала, что будет сделано в следующем разделе.

3.4 Сравнительная оценка эффективности реализации DPA с использованием ковариационных и корреляционных векторов

Повторно рассмотрим ковариационные вектора, полученные после обработки экспериментальных данных об энергопотреблении, на истинном и наихудшем для злоумышленника ложном вариантах перебираемого подблока ключа (рисунки 3.3-3.4). Амплитуды всплесков на них, для каждого отсчёта, очевидно определяются тремя причинами:

- Наличием зависимости значений соответствующих отсчётов на формах сигнала от промежуточной комбинации, рассчитываемой на конкретном варианте перебираемого подблока ключа.
- Значением $U_{\text{ДАН}}$ (амплитуда всплеска прямо пропорционально зависит от значения этой величины).
- Направлением зависимости (прямая или обратная) – эта характеристика определяет полярность всплеска.

Рассмотрим такты на которых существует зависимость энергопотребления чипа от обрабатываемых им данных. Значение $U_{\text{ДАН}}$ для любого отдельно взятого отсчёта на ковариационном векторе не зависит от варианта перебираемого подблока ключа. Поэтому отношения амплитуд всплесков, полученных на отдельном отсчёте на истинном и наихудшем ложном вариантах ключа, будут одинаковыми для всех отсчётов, хотя в абсолютных значениях

они могут отличаться. Приведём пример, амплитуда всплеска на истинном ключе на 141-м отсчёте составляет $3.41 \cdot 10^{-3}$, а на наихудшем ложном $-2.82 \cdot 10^{-3}$, что составляет 82.7% от истинного. На 138-м отсчёте амплитуда на истинном ключе составляет $-8.74 \cdot 10^{-3}$, а на наихудшем ложном $7.29 \cdot 10^{-3}$, что, учитывая погрешность округления, составляет те же 82.7%. По той причине, что $U_{\text{ДАН}}$ на 138-м отсчёте имеет максимальное значение, следовательно всплеск на ковариационном векторе на истинном и наихудшем для злоумышленника ложном вариантах ключа будет иметь максимальную амплитуду именно в этом отсчёте. Поэтому решение об истинности варианта, перебираемого подблока ключа будет приниматься по этому всплеску, хотя остальные имеют такое же отношение амплитуд на истинном и ложных вариантах ключа.

Как говорилось ранее, расчёт корреляционного вектора заключается в делении каждого отсчёта ковариационного вектора на произведение $\sqrt{2} \cdot \sigma\left(S\left(j\right)\right)$, значение которого определяют две компоненты:

- СКО весов Хэмминга рассчитываемого промежуточного значения (это фиксированная величина не зависящая от варианта перебираемого подблока ключа и параметров рассматриваемого отсчёта, она составляет $\sqrt{2}$).
- Значение СКО самого отсчёта, эта величина определяется значением $U_{\text{ДАН}}$ и значением накладываемой шумовой компоненты, они индивидуальны для каждого отсчёта, но от перебираемого варианта подблока ключа не зависят.

Поэтому расчёт массива корреляционных векторов будет представлять собой деление каждого отсчёта ковариационного вектора (для всех вариантов ключа) на некоторый коэффициент. Этот коэффициент будет разным для каждого отсчёта, но от перебираемого ключа зависеть не будет.

Соотношение сигнал/шум является наибольшим на 141-м отсчёте, в этом отсчёте значение ковариации будет разделено на 0.00397 (другими словами умножено на 252.14) – всплеск в этом отсчёте на корреляционном векторе окажется максимальным, на истинном ключе он будет составлять $\frac{3.41 \cdot 10^{-3}}{0.00397} = 3.41 \cdot 10^{-3} \cdot 252.14 = 0.8598$, на наихудшем ложном: $\frac{-2.82 \cdot 10^{-3}}{0.00397} = -2.82 \cdot 10^{-3} \cdot 252.14 = -0.70939$ (те же 82.7% от истинного).

Соотношение сигнал/шум на 138-м отсчёте окажется значительно ниже – на этом отсчёте значение ковариации будет разделено на 0.02269 (другими словами умножено на 44.08) – всплеск в этом отсчёте на корреляционном векторе будет составлять всего

$\frac{-8.74\cdot10^{-3}}{0.02269} = -8.74\cdot10^{-3}\cdot44.08 = -0.3852$, а на наихудшем ложном: $\frac{7.29\cdot10^{-3}}{0.02269} = 7.29\cdot10^{-3}\cdot44.08 = 0.3213$ (те же 82.7% от истинного).

Из приведённого примера понятно, что относительные амплитуды всплесков, рассчитанных на истинном и ложных вариантах перебираемого подблока ключа в отсчётах, относящихся к обработке рассчитываемого промежуточного значения для корреляционных и ковариационных векторов будут одинаковыми. Может измениться лишь отсчёт, в котором происходит максимальный всплеск на истинном и наихудших ложных вариантах ключа. Поэтому, по крайней мере после обработки достаточного, для устранения шумовой компоненты, количества форм сигнала, разница между эффективностью реализации DPA с использованием ковариационных и корреляционных векторов будет отсутствовать. (Рисунок 2.32а в сравнении с рисунком 3.3 и рисунок 3.8 в сравнении с рисунком 3.4.)

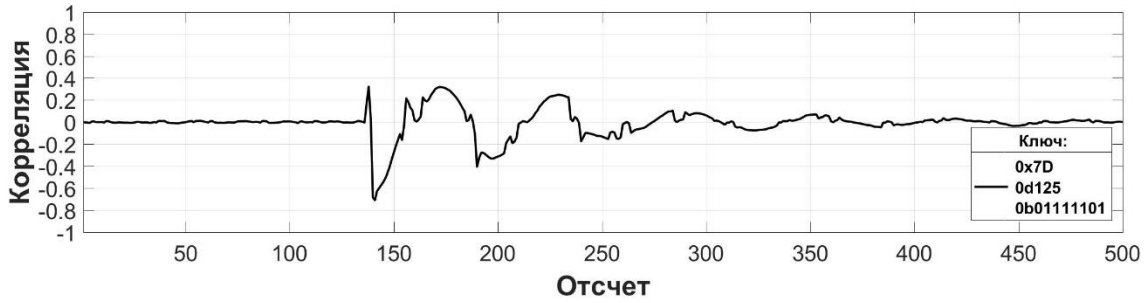
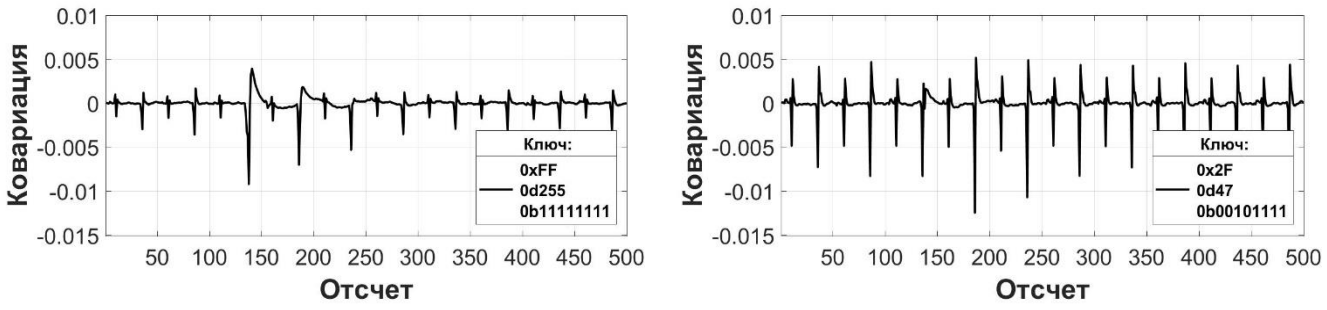


Рисунок 3.8 – Увеличенная область корреляционного вектора, полученного при использовании в качестве рассчитываемого промежуточного значения, комбинации, копируемой в чистую ячейку ОЗУ, при обработке 50 000 форм, на ложном варианте перебираемого подблока ключа

Отличия в эффективности DPA с использованием ковариационных и корреляционных векторов будут существовать лишь в количестве форм сигнала, требуемом для минимизации побочных всплесков. Рассмотрим полученные на практике ковариационные вектора, рассчитанные на истинном и наихудшем для злоумышленника ложном вариантах перебираемого подблока ключа, с использованием небольшого количества форм сигнала (рисунки 3.9а,б).

а) б)



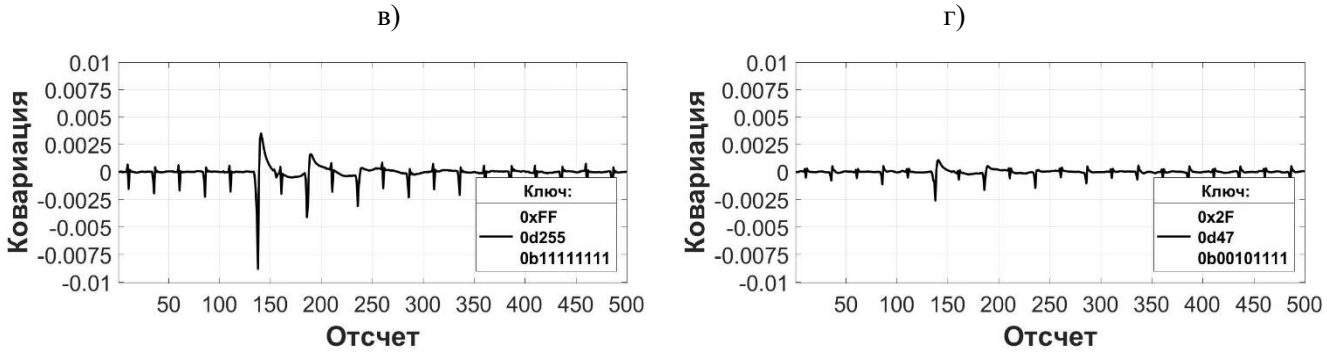


Рисунок 3.9 – Ковариационные вектора, рассчитанные на ключах: «1111 1111» (истинный) и «0010 1111» (наихудший ложный) при обработке: а), б) 256-и форм сигнала; в), г) 1 024-х форм сигнала

Из рисунков 3.9а,б видно, что при использовании 256-ти форм сигнала определение истинного варианта перебираемого подблока ключа по всплескам ковариационного вектора окажется невозможным. В рассматриваемом примере истинный ключ будет лишь восьмым претендентом. То есть ковариационные вектора, рассчитанные на семи из 255-ти ложных вариантов перебираемого подблока ключа будут иметь б**о**льшие всплески, чем вектор, рассчитанный на истинном ключе. При этом всплески на ложных ключах будут возникать в первой и третьей областях тактов (рисунок 2.15). Как говорилось ранее, в этих областях наблюдаются резкие спады и нарастания сигнала. Из-за ограниченности полосы пропускания измерительной установки, значения соответствующих отсчётов снятых форм сигнала, от измерения к измерению будут крайне нестабильными (рисунок 2.9). Таким образом, значения всплесков на представленном ковариационном векторе будут представлять собой значения коэффициентов ковариации между рассчитываемыми на перебираемом подблоке ключа комбинациями и изменяющимися в весьма широких пределах значениями соответствующего отсчёта. Как говорилось ранее особенностью коэффициента ковариации является то, что его значение определяется не только силой зависимости случайных величин, но и значением $U_{\text{ДАН}}$. Поэтому, даже если в этих областях будет наблюдаться небольшая зависимость значений сигнала от рассчитываемой на ложном варианте перебираемого подблока ключа комбинации, то за счёт значительной величины $U_{\text{ДАН}}$, значения ковариации могут быть весьма большими. При увеличении количества обрабатываемых форм сигнала эти зависимости сходят на нет (при этом относительные амплитуды всплесков на истинном и ложных вариантах ключа для ковариационного и корреляционного векторов оказываются приблизительно одинаковыми), поэтому даже несмотря на значительные величины $U_{\text{ДАН}}$, амплитуды всплесков сильно снижаются (рисунки 3.9в,г). В приведённом примере, увеличением количества обрабатываемых форм сигнала можно свести максимальные амплитуды побочных всплесков от величины $12 \cdot 10^{-3}$

(при обработке 256-и форм сигнала) до $1 \cdot 10^{-3}$ (при обработке 50 000 форм сигнала), что без труда позволяет выделить всплеск на истинном варианте ключа (амплитуда которого составляет порядка $8 \cdot 10^{-3}$).

При расчёте корреляционного вектора производится деление каждого отсчёта ковариационного вектора на произведение $\sqrt{2} \cdot \sigma\left(S\left(j\right)\right)$. На рисунке 3.10 приведён график СКО значений отсчётов, из него видно, что всплески значений СКО возникают в отсчётах, соответствующих первой и третьей областям тактов. Отсюда понятно, что побочные всплески на ковариационных векторах «гасятся» при делении на значения $\sqrt{2} \cdot \sigma\left(S\left(j\right)\right)$. Поэтому их амплитуда на корреляционных векторах в первой и третьей областях тактов, значительно снижается (рисунки 2.29) по сравнению со случаем расчёта ковариационных векторов. И даже на весьма небольших выборках становится возможным определение вектора, рассчитанного на истинном варианте перебираемого подблока ключа. Этот эффект вполне логичен, так как смысл расчёта корреляции и заключается в минимизации влияния размерностей случайных величин на значения ковариации. Как было показано в предыдущем разделе, при расчёте корреляции, по сравнению с ковариацией, не устраняется полностью зависимость значений от размерностей случайных величин, но она значительно снижается.

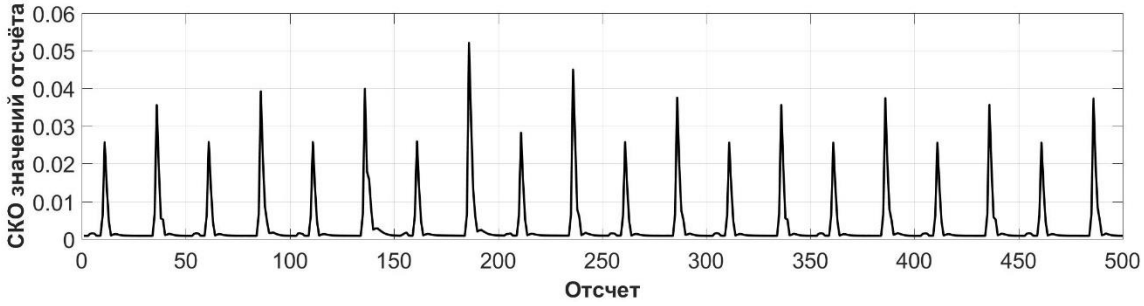


Рисунок 3.10 – График СКО значений отсчётов, статистика, получена с использованием 50 000 форм сигнала

Сводные графики ковариационных векторов (рисунки 3.11 в сравнении с рисунками 2.30-2.31) подтверждают, что на практике (с использованием форм сигнала, снятых с реального чипа), применение корреляционных векторов позволит определить истинный вариант ключа с использование меньшего количества форм сигнала (потребуется всего порядка 20 форм сигнала, по сравнению с 550 формами сигнала, требующимися в случае использования ковариационных векторов).

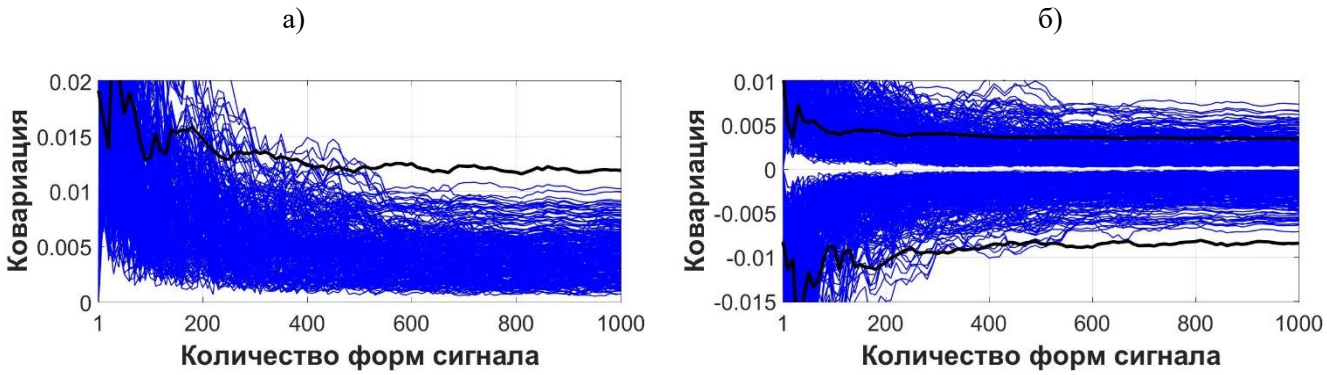


Рисунок 3.11 – а) Сводные графики ковариационных векторов; б) сводные графики ковариационных векторов по сумме максимумов и минимумов, полученные при использовании в качестве рассчитываемого промежуточного значения, комбинации, копируемой в чистую ячейку ОЗУ

В таблице 3.3, по аналогии со случаем рассмотрения корреляционных векторов (таблица 2.3), приведены основные качественные характеристики ковариационных векторов. Необходимо подчеркнуть, что при расчёте приводимых значений использовались те же формы сигнала, что и в случае рассмотрения корреляционных векторов.

Анализ данных показывает, что в случае использования ковариационных векторов, для однозначного определения истинного варианта перебираемого подблока ключа, потребуется обработка как минимум на порядок большего количества форм сигнала, чем в случае использования корреляционных. При этом, к примеру, для операции копирования из ОЗУ в чистую ячейку рабочего регистра недостаточно обработки 50 000 форм сигнала, хотя в случае использования корреляционных векторов – для однозначного определения ключа было достаточно всего 410 форм сигнала. Данный эффект весьма просто объясняется описанными выше особенностями ковариационных векторов – рассмотрим этот случай подробнее. Из таблицы Е.3 приложения Е известно, что операция чтения из ОЗУ в чистую ячейку регистра приводит к крайне малой утечке по цепи электропитания – максимальное значение $U_{\text{ДАН}}$ достигает 0.3 мВ, против 7 мВ для операции записи в ОЗУ (то есть более чем 20 раз меньше). Из приведённых выше разъяснений известно, что амплитуда всплесков на ковариационных векторах прямо пропорционально зависит от размерности $U_{\text{ДАН}}$. Отсюда амплитуда всплеска на ковариационном векторе, полученном на истинном варианте перебираемого подблока ключа, должна быть более чем в 20 раз меньше всплеска полученного для операции записи данных в ОЗУ (то есть вместо $8\cdot10^{-3}$ амплитуда должна составлять порядка $4\cdot10^{-4}$). Как говорилось ранее обработка 50 000 форм сигнала позволяет снизить амплитуды побочных всплесков лишь до значений порядка $1\cdot10^{-3}=10\cdot10^{-4}$, отсюда выделить ковариационный вектор рассчитанный на истинном варианте перебираемого подблока ключа на фоне ложных будет невозможно –

потребуется обработка большего количества форм сигнала для ещё большей минимизации побочных всплесков.

На основе полученных данных можно сделать следующие основные выводы:

- Прямо пропорциональная зависимость значений ковариационных векторов от величины утечки по цепи электропитания на практике приводит к тому, что для успешной реализации DPA требуется обработка весьма большого количества форм сигнала.
- Несмотря на то, что подход с расчётом корреляционных векторов не приводит к полному устранению зависимости амплитуд всплесков от значений $U_{\text{ДАН}}$, данный подход на практике является значительно более эффективным, чем подход с расчётом ковариационных векторов.

Таблица 3.3 – Основные характеристики ковариационных векторов, полученных при реализации DPA в отношении ряда операций

Выполняемая операция	Состояние ячейки назначения	Используемая модель энергопотребления	Вид анализа: по максимальным отрицательным всплескам или по сумме положительных и отрицательных	Максимальная амплитуда всплеска на наихудшем ложном ключе в процентах по отношению ко всплеску на истинном	Минимальное количество форм сигнала, требующееся для успешной реализации атаки
Копирование в ОЗУ	Чистая	Вес Хэмминга	Отрицательные	77.7%	550
			Сумма	83.1%	550
	Предварительно заполненная	Вес Хэмминга	Отрицательные	76.8%	450
			Сумма	81.6%	800
		Расстояние Хэмминга	Однозначное определение истинного ключа невозможно		
Копирование из ОЗУ	Чистая	Вес Хэмминга	Требуется использование более 50 000 форм сигнала		
	Предварительно заполненная	Вес Хэмминга	Однозначное определение истинного ключа невозможно		
		Расстояние Хэмминга	Отрицательные	84.93%	32 500
			Сумма	81.36%	32 500
Копирование из ПЗУ	Чистая	Вес Хэмминга	Отрицательные	После обработки 1200 форм сигнала и вплоть до 50 000 истинный ключ является вторым претендентом	
			Сумма	92.4%	11 000
	Предварительно заполненная	Вес Хэмминга	Однозначное определение истинного ключа невозможно		
		Расстояние Хэмминга			

Продолжение таблицы 3.3

Выполняемая операция	Состояние ячейки назначения	Используемая модель энергопотребления	Вид анализа: по максимальным отрицательным всплескам или по сумме положительных и отрицательных	Максимальная амплитуда всплеска на наихудшем ложном ключе в процентах по отношению ко всплеску на истинном	Минимальное количество форм сигнала, требующееся для успешной реализации атаки
Копирование между рабочими регистрами	Чистая	Вес Хэмминга	Отрицательные	86.7%	3 500
			Сумма	88.9%	3 500
	Предварительно заполненная	Вес Хэмминга	Отрицательные	75.1%	25 000
			Сумма	82.7%	3 500
		Расстояние Хэмминга	Отрицательные	72.9%	12 000
			Сумма	76.5%	32 500
Сложение по mod2 между рабочими регистрами	Вес Хэмминга комбинации из первой ячейки		Отрицательные	87.7%	1 100
			Сумма	86.1%	1 100
	Вес Хэмминга суммы по mod2		Отрицательные	Требуется использование более 50 000 форм сигнала	

3.5 Анализ отличий в эффективности применения DPA к шифрам «Магма» и «Кузнечик»

Анализ устойчивости защитного алгоритма к атаке DPA начинается с поиска некоторого промежуточного значения $a_l(i) = f\left(m_l(i),\ k_l\right)$, зависящего от известных сообщений и *небольшой* части секретного ключа, которое можно было бы рассчитать, при помощи несложных преобразований $f(\cdot)$, являющихся частью защитного алгоритма. Как правило, в качестве промежуточного значения выбирается выход S-box. В данном случае между «Магмой» и «Кузнечиком» существуют три отличия:

- Блоки сообщения/ключа в шифре «Кузнечик» длиннее чем в «Магме».
- В шифре «Магма» сообщение складывается с раундовым ключом по $\mathrm{mod}2^{32}$, а в шифре «Кузнечик» – по mod2.
- Шифры «Магма» и «Кузнечик» имеют S-box разной разрядности – в первом случае они четырёх разрядные, во втором случае – восьми разрядные.

Произведём аналитическое сравнение достоинств и недостатков операций сложения по mod2 и по $\mathrm{mod}2^{32}$, в контексте реализации DPA. На первый взгляд может показаться, что использование в «Кузнечике» операции сложения по mod2, вместо $\mathrm{mod}2^{32}$, скорее упрощает реализацию атаки, так как в данном случае отсутствует перенос бита между разрядами, и как следствие входы и выходы соседних S-box не будут зависеть друг от друга. Поэтому атаку не обязательно начинать с восстановления младших бит раундового ключа. Его можно восстанавливать поблочно (по восемь бит, в соответствии с анализируемым S-box), начиная с абсолютно любого из 16-ти S-box. Но справедливо отметить, что независимо от того будет ли злоумышленник восстанавливать подблоки раундового ключа передвигаясь от старших к младшим, или от младших к старшим, или в любой другой последовательности, ему всё равно будет необходимо восстановить 16 восьми битовых подблоков. Поэтому использование в «Кузнечике» операции сложения по mod2, вместо $\mathrm{mod}2^{32}$ не приведёт ни к увеличению вероятности корректного восстановления подблока ключа, ни к увеличению скорости реализации атаки. Однако это приведёт к тому, что ошибки в восстановлении младших бит ключа не будут влиять на истинность расчёта входов и выходов старших S-box, что может оказаться негативным фактором, так как использование операции сложения по $\mathrm{mod}2^{32}$ в «Магме» давало некоторую «помехозащищённость» при восстановлении старших частей раундового ключа. Если при восстановлении младших бит происходила ошибка, то старшие биты суммы $m(i)+k(r)$, где $+$ – операция сложения по $\mathrm{mod}2^{32}$, для ряда сообщений рассчитывались неверно, а следовательно

выходы соответствующего старшего S-box, на истинном варианте перебираемого подблока ключа, также рассчитывались не верно, что приводило к появлению меньших всплесков на корреляционном векторе. Это, в свою очередь, сигнализировало нападающему о необходимости поиска ошибки в младшей части восстанавливаемого ключа. Поэтому использование операции сложения по mod2 в «Кузнечике», в контексте реализации DPA можно считать недостатком.

Как было показано в подразделе 1.7.2, при использовании в качестве рассчитываемого промежуточного значения, результата сложения по mod2 подблоков сообщения и ключа, на ложных вариантах перебираемого подблока ключа, будут возникать всплески с весьма большой амплитудой (на восьми из 255 ложных ключей амплитуда всплеска будет достигать 75% от истинного, ещё на 28-ми вариантах – 50% от истинного, на остальных 219-ти вариантах амплитуда всплеска будет меньше 50% от всплеска, получаемого на истинном варианте ключа).

Не сложно проверить моделированием, что при использовании в качестве рассчитываемого промежуточного значения, результата сложения по $mod2^{32}$ подблоков сообщения и ключа, всего на одном ложном ключе, будет возникать всплеск с амплитудой в 75% от всплеска, полученного на истинном ключе. Ещё на двух вариантах – 62.5%, и на 12-ти вариантах, амплитуда всплесков будет лежать в диапазоне от 50.2% до 56.3% от амплитуды, всплеска, получаемого на истинном ключе. На остальных 240-а ложных вариантах ключа всплески будут иметь амплитуды менее 50% от амплитуд всплесков, получаемых на истинном ключе.

В обоих случаях всплески на ложных вариантах перебираемого подблока ключа можно считать чрезмерно большими для использования данного промежуточного значения при практической реализации атаки DPA. Несмотря на это, «Кузнечик» всё же можно считать более устойчивым к DPA, по той причине, что при сложении по mod2 будет восемь ложных вариантов ключа приводящих к возникновению всплесков с амплитудой в 75% от амплитуды, полученной на истинном варианте ключа. Тогда как для операции сложения по $mod2^{32}$ такие всплески будут наблюдаться лишь на одном варианте ключа, а всплески в 62.5% от истинного (тем более 56.3%) можно считать менее опасными для злоумышленника, чем 75%-е всплески.

Также шифры отличаются используемыми S-box. Так в «Магме» используются S-box, преобразующие четыре входных бита в четыре выходных, а в «Кузнечике» S-box, преобразующие восемь входных в восемь выходных. Отсюда можно сделать поспешное заключение, что атака на S-box «Магмы» окажется проще. Дело в том, что минимальный восстанавливаемый подблок раундового ключа «Кузнечика» состоит из восьми бит – очевидно, что перебором меньших частей ключа невозможно рассчитать вход S-box, а, следовательно, и достоверно определить его выход (не важно один это бит или все восемь). Поэтому злоумышленнику потребуется перебрать 256 возможных комбинаций подблока ключа и на каждом из которых (для массива шифруемых сообщений) нужно рассчитать ковариационный

вектор. В это же время, в «Магме» используются четырёх разрядные S-box, и поэтому минимальный подблок восстанавливаемого ключа будет состоять из четырёх бит. Следовательно, для восстановления восьми бит раундового ключа будет необходимо произвести перебор сначала четырёх младших бит (16 возможных комбинаций), получив 16 ковариационных векторов, из которых определить тот вектор, на котором всплески имеют б**о**льшую амплитуду. Затем аналогично найти четыре последующих бита ключа. Таким образом, в сумме, производится перебор 32-ух вариантов подблока ключа, что очевидно занимает меньше времени, и кроме этого, на практике, два раза найти один вектор с наибольшими всплесками из 15-ти векторов с меньшими всплесками может оказаться проще, чем один раз найти один вектор из 255-ти.

Однако необходимо понимать важную особенность современных микрочипов: на текущий момент сложно найти чип, имеющий четырёх разрядную архитектуру; б**о**льшая часть чипов имеет восьми разрядную архитектуру (сейчас уже всё б**о**льшую популярность получают также 16-ти и 32-ух разрядные чипы). Конечно, существуют специализированные криптографические чипы (называемые криптографическим сопроцессором или криптопроцессором [15]), которые могут изготавливаться как на отдельном кристалле, так и быть частью кристалла основного чипа. Они проектируются под конкретный алгоритм защитного преобразования (или несколько алгоритмов) и имеют наиболее эффективную (для реализации выполняемого алгоритма) архитектуру. Для выполнения шифра DES такой криптопроцессор может иметь шести разрядную архитектуру, для «Магмы» четырёх разрядную, что позволит обрабатывать каждый S-box отдельно. Впрочем, принципиально возможна реализация одновременного (за один такт работы) преобразования на всех восьми S-box «Магмы» (или 16-ти S-box «Кузнечика»), для этого, очевидно просто необходимо изготовить соответствующую электронную схему. Возможность изготовить электронную схему под конкретный шифр позволяет реализовать «Магму» и «Кузнечик» чрезвычайно большим количеством способов. Однако сравнение эффективности реализации всех таких вариантов выходит за рамки текущей работы, и по большому счёту является бессмысленным. Поэтому в данной работе производится сравнение эффективности реализации шифров «Магма» и «Кузнечик» при их программной реализации на наиболее распространённых, на сегодняшний день, восьми разрядных чипах (в частности микроконтроллерах) с применением соответствующих им типовых ассемблеров.

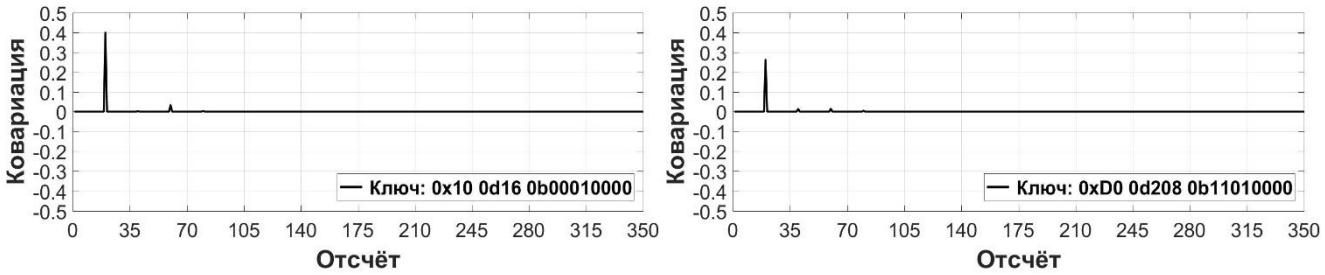
В приложении Ж описывались принципы практической реализации преобразований в S-box на чипах с восьми разрядной архитектурой. В частности, приводился пример, показывающий неэффективность четырёх битовых преобразований шифра «Магма» на восьми битовой архитектуре чипа. Поэтому в реальных чипах и в случае «Магмы», и в случае «Кузнечика» за один подход будет производиться преобразование над восемью битами. Табличное нелинейное

преобразование «Магмы» будет реализовано в виде преобразований на сдвоенных S-box, то есть за один подход будет осуществляться преобразование сразу на паре S-box. Следовательно, в «Магме» один отсчёт на форме сигнала будет характеризовать энергопотребление чипа, затрачиваемое на одновременную обработку выходов двух S-box.

Для демонстрации эффективности реализации DPA в отношении «Магмы» (с использованием выходов сдвоенных S-box) был сгенерирован массив форм сигнала с разным уровнем зашумлённости, по аналогии с формами сигнала для шифра «Кузнечик» (раздел 3.1). Отличие в данном случае заключалось в том, что на каждой форме сигнала вместо 16-ти, находилось всего четыре отсчёта, характеризующих энергопотребление чипа при обработке выходов S-box. (Это обусловлено тем, что в «Магме» на каждом раунде нелинейное преобразование осуществляется всего над 32-я битами, для чего требуется четыре сдвоенных S-box, в то время как в «Кузнечике» нелинейное преобразование осуществляется над 128-ю битами, для чего требуется 16 стандартных восьми битовых S-box.) Также, для обеспечения преемственности с результатами моделирования DPA в отношении шифра «Кузнечик», было принято решение не менять длину форм сигнала (350 отсчётов) и расстояние между значащими отсчётами (19 отсчётов между отсчётами, характеризующими энергопотребление соседних пар S-box). Алгоритм DPA применяемый к сгенерированным формам сигнала в точности повторял алгоритм для шифра «Кузнечик», в качестве промежуточного значения был выбран выход последнего сдвоенного 7-8 S-box, поэтому производился перебор восьми младших бит раундового ключа первого раунда, на каждом варианте которого, для массива преобразуемых сообщений рассчитывались выходы 7-8-го S-box «Магмы», а затем ковариационные вектора.

На рисунках 3.12 представлены ковариационные вектора, рассчитанные с использованием не зашумлённых форм сигнала. При их сравнении с рисунками 3.1 становится очевидной значительно меньшая разница между всплеском, полученным на истинном и наихудшем ложном вариантах перебираемого подблока ключа. Так всплеск на истинном ключе составил 0.4 (также, как и в «Кузнечике»), а на наихудшем ложном 0.2625 (что составляет 65.6% от истинного), тогда как в «Кузнечике» было 20.3% от истинного.

а) б)



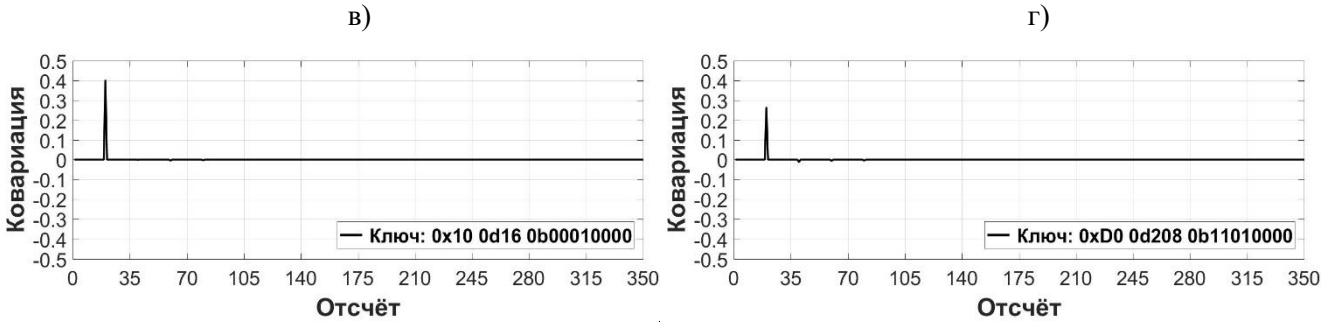


Рисунок 3.12 – Ковариационные вектора, рассчитанные с использованием не зашумлённых форм сигнала на истинном и наихудшем для злоумышленника ложном вариантах перебираемого подблока ключа при обработке: а), б) 256-ти форм сигнала; в), г) 2 048-ми форм сигнала

Реализация эксперимента была повторена в отношении различного количества зашумлённых форм сигнала (таблица 3.4), что подтвердило наличие значительных всплесков на ложных вариантах перебираемого подблока ключа.

На первый взгляд может показаться, что столь значительная амплитуда всплесков обусловлена структурой таблиц замен «Магмы», однако, в контексте реализации DPA степень нелинейности преобразований на S-box «Магмы» не имеет значительного влияния на результат атаки. Выявленная же проблема обусловлена особенностью работы сдвоенных S-box. Рассмотрим этот вопрос более подробно, сначала рассмотрим S-box «Кузнечика» (рисунок 3.5) где на вход S-box поступает сумма по mod2 восьми бит шифруемого сообщения и раундового ключа. При этом если произойдёт ошибка хотя бы в одном бите подблока ключа: $k'_{0-7}(d) \neq k_{0-7}$, то сумма: $m_{0-7}(i) \oplus k'_{0-7}(d)$, для любых вариантов сообщения $m_{0-7}(i)$ будет рассчитана неверно в том же разряде в котором произошла ошибка. (Операция сложения по mod2 не предусматривает переноса бита между разрядами, поэтому ошибка размножаться не будет.) Однако по причине нелинейности преобразования в S-box (то есть зависимости выхода S-box от абсолютно всех бит его входа) ошибка даже в одном бите на входе S-box, может привести к ошибкам вычисления от одного до восьми бит выхода S-box. Поэтому вес Хэмминга выхода S-box на ложных вариантах перебираемого подблока ключа, для конкретных сообщений, конечно, может совпадать с весом Хэмминга выхода S-box при его расчёте на истинном ключе, однако, после перебора всех возможных 256-ти вариантов $m_{0-7}(i)$ зависимость между ними будет практически отсутствовать.

Таблица 3.4 – Амплитуды максимальных всплесков на ковариационных векторах, рассчитанных на наихудшем для злоумышленника ложном варианте перебираемого подблока ключа (в процентах, по сравнению с амплитудами всплесков, на векторах, рассчитанных на истинном варианте), в зависимости от количества обрабатываемых форм сигнала и СКО шумовой компоненты

	Амплитуды максимальных всплесков, полученных на наихудших ложных вариантах подблока ключа, %			
$N \setminus \sigma_{ШУМ}$	0	0.1	0.2	0.3
256	65.6	66.0	74.1	65.0
512	65.6	66.0	65.0	67.1
1 024	65.6	66.0	65.1	65.2
2 048	65.6	64.9	66.0	64.5
4 096	65.6	66.2	65.8	66.0
8 192	65.6	65.9	66.5	66.1
16 384	65.6	65.6	65.3	66.8
32 768	65.6	65.5	65.8	65.8
65 536	65.6	65.8	65.7	65.8

В описанной же модели реализации «Магмы», в роли одного составного S-box, выступают два независимых S-box – 7-й и 8-й. В данном случае, вначале производится сложение по $\mathrm{mod} 2^{32}$ восьми младших бит сообщения с восемью битами подблока ключа, и при этом может происходить перенос бита в соседний разряд. Рассмотрим эту операцию подробнее (рисунок 3.13). Положим, что перебираемый вариант подблока ключа был угадан неверно и при этом он отличается от истинного в старших четырёх разрядах (не важно в одном бите или во всех четырёх, здесь и далее половина байта будет называться «ниббл»), а младший ниббл (на рисунке 3.13 это биты от 0 до 3) совпадает с истинным. В этом случае очевидно, что для всех преобразуемых сообщений, вход (а, следовательно, и выход) восьмого S-box будет рассчитан верно, а седьмого – неверно. Аналогично, если ошибка произойдёт в младшем ниббле, то вход и выход восьмого S-box для всех сообщений будет рассчитан неверно, а седьмого – верно. Однако, это произойдёт не для всех сообщений, потому что из-за ложности младшего ниббла подблока ключа могут возникать ошибки в переносе бита между двумя ниббламы, что неминуемо повлечёт за собой ошибки в старшем ниббле суммы (поступающем на седьмой S-box). Однако всё равно, для большей части сообщений вход и выход седьмого S-box будет рассчитан верно.

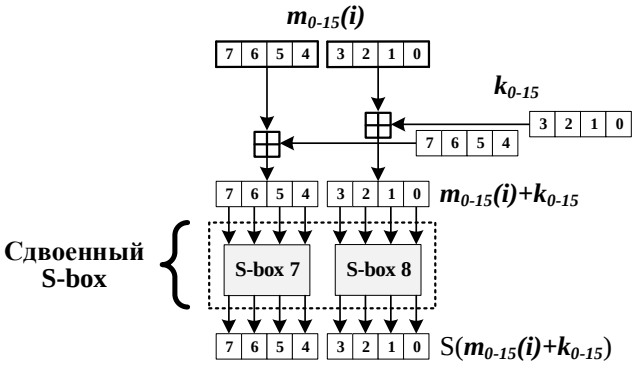


Рисунок 3.13 – Блок-схема части раунда шифра «Магма»

Отсюда очевидно следует что на 15-ти ложных вариантах перебираемого подблока ключа (отличающихся от истинного старшим нибблом) четыре из восьми бит выхода, сдвоенного S-box для всех преобразуемых сообщений, будут рассчитаны верно и поэтому на соответствующих ковариационных векторах возникнут весьма значительные всплески. Ещё на 15-ти ложных вариантах (отличающихся от истинного младшим нибблом) эти всплески будут также весьма значительными, но с несколько меньшими амплитудами. Таким образом, при реализации DPA, 30 ложных перебираемых вариантов подблока ключа гарантированно дадут высокие ковариационные всплески. Получим, что эффективность реализации DPA в отношении «Магмы» будет значительно ниже по сравнению с «Кузнечиком».

Однако, в данном случае, необходимо понимать, что в контексте реализации DPA, подход с обработкой чипом за один раз пары S-box выражается для нападающего лишь в том, что рассматриваемый отсчёт на формах сигнала, будет характеризовать энергопотребление, затрачиваемое на обработку не четырёх бит выхода S-box, а восьми бит пары S-box. Подход к реализации «Магмы» с использованием сдвоенных S-box ни в коем случае не изменяет алгоритм защитного преобразования, и поэтому злоумышленник может производить восстановление раундового ключа не по восемь бит, а по четыре (без учёта особенностей практической реализации алгоритма «Магмы»). Понятно, что за счёт существования явления переноса бит из старших разрядов в младшие, восстановление раундового ключа необходимо осуществлять, двигаясь от младших четырёх битовых подблоков к старшим. Очевидным достоинством данного подхода можно считать меньшее количество вариантов перебираемого подблока ключа – вместо 256-ти вариантов, в данном случае, будет необходимо перебрать всего 32 варианта. Вначале (рисунок 3.13) необходимо перебрать 16 вариантов четырёх младших бит ключа, при этом, на каждом варианте, для массива преобразованных сообщений, рассчитывается выход восьмого S-box, определяется его вес Хэмминга, затем рассчитываются корреляционные вектора. После восстановления 0-3 бит раундового ключа, можно начинать перебор 4-7 бит (если 0-3 биты ключа будут восстановлены верно, для всех зашифрованных сообщений можно будет корректно

определить перенос бита между двумя нибблами, при выполнении операции сложения по $mod 2^{32}$.

Данный метод обладает очевидным недостатком – за счёт того, что модель не будет учитывать энергопотребление, затрачиваемое на выработку четырёх бит выхода соседнего S-box – соответствующая часть всплеска на формах сигнала перейдёт из «полезной» составляющей в шумовую, то есть будет представлять собой дополнительный алгоритмический шум (рисунок 3.14). Следовательно, ухудшится соотношение сигнал/шум (это приведёт к тому, что потребуется обработка большего количества форм сигнала).

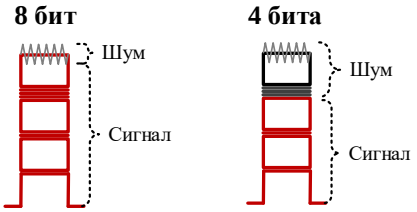
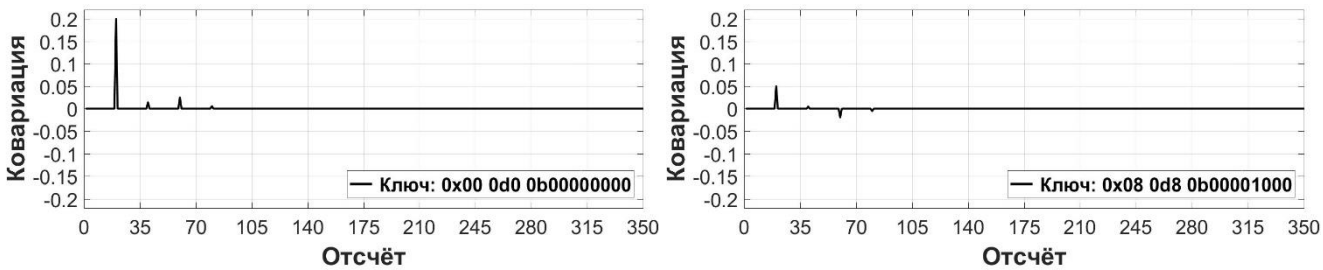


Рисунок 3.14 – Упрощенная модель энергопотребления, для рассматриваемого отсчёта

Однако очевидно, что старшие четыре бита преобразуемого сообщения не зависят от младших бит, и даже с учётом возникающего переноса из младшего ниббла в старший, выход седьмого S-box не будет иметь сколько-нибудь значимой зависимости от выхода восьмого S-box, рассчитанного на истинном или любом ложном варианте перебираемого подблока ключа (по крайней мере при накоплении значительной статистики). Это означает, что накладываемый алгоритмический шум будет иметь нулевое среднее, и его возможно будет минимизировать, увеличением количества обрабатываемых форм сигнала, также, как и электронный шум. На данном эффекте и основывается предположение о возможно большей эффективности реализации «четырёх битового» DPA в отношении сдвоенных S-box, по сравнению с «восьми битовым». На рисунках 3.15 представлены ковариационные вектора, полученные при восстановлении четырёх младших бит раундового ключа «Магмы»; на них видно значительное большее отличие всплеска, полученного на истинном подблоке ключа, от всплеска, полученного на наихудшем ложном (который оказывается ровно в 4 раза ниже истинного).

а) б)



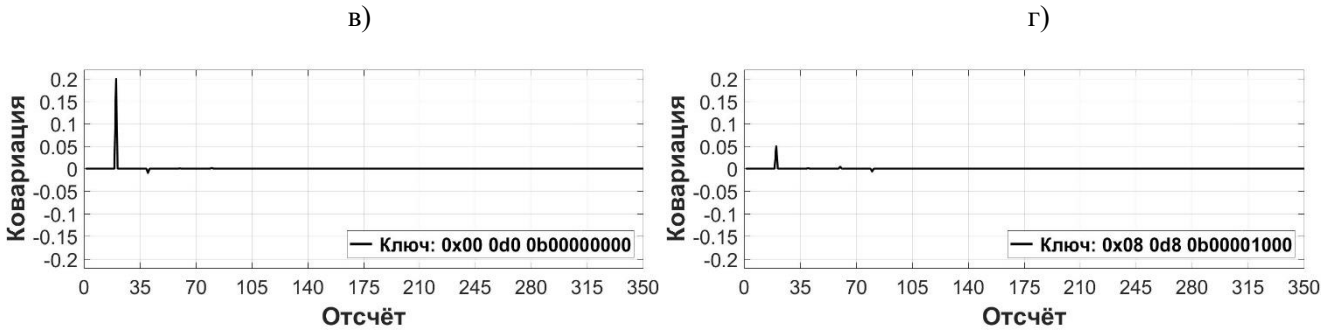
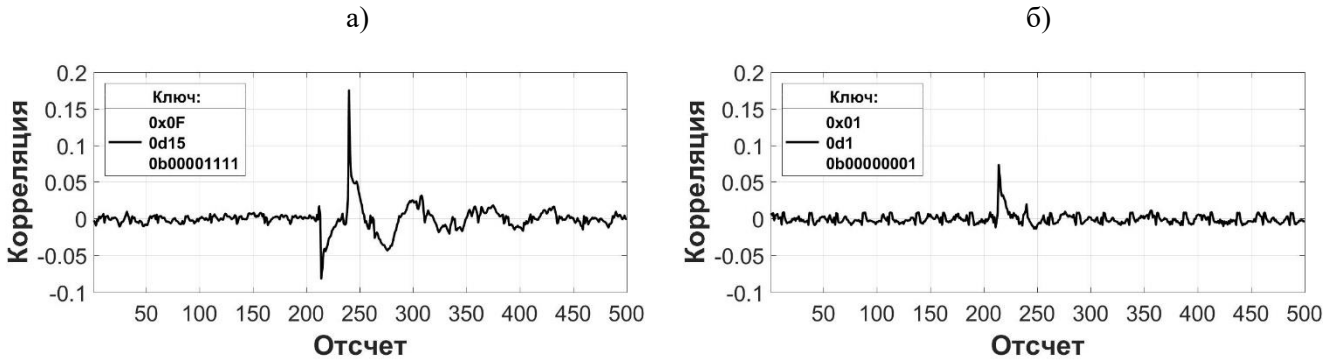


Рисунок 3.15 – Ковариационные вектора, рассчитанные с использованием не зашумлённых форм сигнала на истинном и наихудшем для злоумышленника ложном вариантах подблока ключа при обработке: а), б) 256-ти форм сигнала; в), г) 2 048-ми форм сигнала

Повторение эксперимента в отношении зашумлённых форм сигнала подтверждает полученные результаты (таблица 3.5а). Реализация четырёх битового DPA в отношении полученных на практике форм сигнала (рисунки 3.16а,б в сравнении с рисунком И.1 приложения И), подтверждает увеличение разницы между всплесками, полученными на истинном и наихудшем ложном вариантах перебираемого подблока ключа (по крайней мере в случае обработки значительного количества форм сигнала). Однако, сводные графики корреляционных векторов показывают (рисунки 3.16в,г), что данный подход не позволит снизить требующееся для успешной реализации DPA, количество обрабатываемых форм сигнала. Как говорилось выше использование четырёх, вместо восьми бит выхода S-box значительно снижает соотношение сигнал/шум в рассматриваемых отсчётах, поэтому на небольших выборках данный эффект будет оказывать значительное негативное влияние на успешность реализации DPA.



в) г)

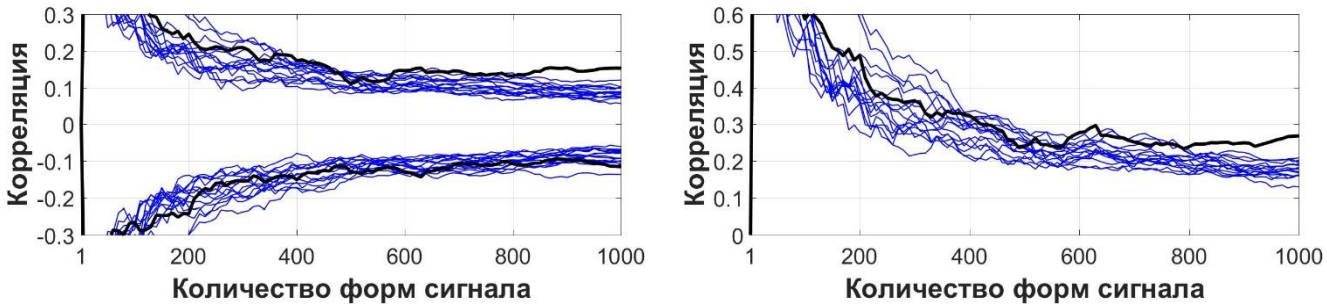


Рисунок 3.16 – а), б) Корреляционные вектора, полученные при использовании в качестве рассчитываемого промежуточного значения, комбинации считываемой из ячейки ОЗУ в чистую ячейку рабочего регистра, при обработке 50 000 форм сигнала, на соответственно истинном и наихудшем для злоумышленника ложном вариантах ключа; в) сводные графики корреляционных векторов; г) сводные графики корреляционных векторов по сумме максимумов и минимумов

Таблица 3.5 – Амплитуды максимальных всплесков на ковариационных векторах, рассчитанных на наихудшем для злоумышленника ложном варианте перебираемого подблока ключа (в процентах, по сравнению с амплитудами всплесков, на векторах, рассчитанных на истинном варианте ключа), в зависимости от количества обрабатываемых форм сигнала и СКО шумовой компоненты, при использовании в качестве рассчитываемого промежуточного значения выхода: а) 8-го S-box; б) 7-го S-box «Магмы»

а)

	Амплитуды максимальных всплесков, полученных на наихудших ложных вариантах подблока ключа, %			
$N \setminus \sigma_{ШУМ}$	0	0.1	0.2	0.3
256	25.0	26.4	30.4	40.4
512	25.0	25.7	25.4	21.1
1'024	25.0	30.1	24.8	19.9
2'048	25.0	24.8	24.0	26.9
4'096	25.0	23.7	25.8	25.3
8'192	25.0	24.4	24.8	23.7
16'384	25.0	25.2	25.1	26.3
32'768	25.0	25.1	25.3	25.3
65'536	25.0	24.6	24.9	24.9

б)

	Амплитуды максимальных всплесков, полученных на наихудших ложных вариантах подблока ключа, %			
$N \setminus \sigma_{ШУМ}$	0	0.1	0.2	0.3
256	31.3	30.9	42.0	37.4
512	31.2	31.0	30.8	33.0
1 024	31.3	31.7	31.3	31.6
2 048	31.3	30.4	31.8	32.2
4 096	31.3	32.4	30.0	32.3
8 192	31.2	31.7	32.4	31.3
16 384	31.2	31.2	31.0	33.5
32 768	31.3	31.1	31.6	31.3
65 536	31.3	31.4	31.1	31.4

Целесообразность использования четырёх битового DPA в отношении «Магмы», на практике можно считать весьма сомнительной по той причине, что данный подход не позволит получить преимущество в виде снижения требуемого количества обрабатываемых форм сигнала. В то время как при накоплении значительной статистики (и как следствие минимизации шумовой составляющей до ничтожных значений) отсутствует практическая разница для злоумышленника между случаями, когда всплеск на наихудшем ложном варианте ключа составляет 65% или 25% от истинного. Отсюда следует, что шифр «Магма» окажется несколько более устойчивым к DPA нежели чем «Кузнечик».

Использование в «Кузнечике» ППС вместо схемы Фейстеля, а также L-преобразования, вместо сдвига на 11 разрядов влево и сложения по mod2 с левой частью входа раунда, в данном случае не будет оказывать влияния на эффективность реализации DPA. Это очевидно происходит потому, что эти операции не используются в процессе расчёта промежуточного значения. После восстановления раундового ключа первого раунда шифра «Кузнечик», не составит труда для всех сообщений произвести L-преобразование над выходом 16-ти S-box, получив таким образом вход второго раунда и аналогичным образом восстановить второй раундовый ключ, это же относится и к «Магме».

Необходимо отметить, что в изученной литературе, в качестве промежуточного значения для атаки DPA исследователи чаще выбирали именно выходы S-box. Однако в ряде чипов, операция обработки комбинации, образующейся в результате преобразований на S-box может не приводить к должным скачкам энергопотребления, за счёт того, что операция чтения из ПЗУ, по средствам которой реализуются преобразования на S-box, не всегда приводит к должному уровню утечки информации по цепи электропитания [21]. Это в свою очередь (как показывали проведённые ранее лабораторные эксперименты) влечёт за собой возникновение крайне

небольших корреляционных всплесков на истинном варианте перебираемого подблока ключа, которые оказывается затруднительно выделить на фоне всплесков, получаемых на ложных. Также преобразование на S-box как правило реализуется таким образом, что считанное из ПЗУ значение выхода S-box, записывается в ячейку, уже содержащую какие-либо данные (по причине небольшого количества свободных регистров чипа). Как следствие такого подхода, возникает необходимость использовать модель расстояния Хэмминга, то есть рассчитывать не вес Хэмминга выхода S-box, а расстояние Хэмминга между комбинацией выхода S-box и комбинацией, содержащейся в той ячейке памяти, которую выход S-box перезаписывает. Это значение и будет соответствовать количеству переключившихся логических вентилей. Для этого необходимо знать какие данные располагались в ячейке до перезаписи. Конечно комбинацию можно предположить (вероятнее всего в ней будет находиться вход S-box, но в ней могут располагаться и какие-либо иные промежуточные значения алгоритма). Учитывая тот факт, что операция чтения из ПЗУ в эту ячейку и без того может приводить к крайне малой «утечке» информации, эмпирическая проверка предполагаемого заполнения ячейки может оказаться весьма затруднительной. По результатам проведённых во второй главе исследований, можно отметить, что для многих операций, начальное заполнение ячейки назначения часто не оказывает сильного влияния на энергопотребление чипа, и как следствие, его можно не учитывать, и работать исключительно с весом Хэмминга выбранного промежуточного значения. Однако данный эффект может и не проявляться в чипах, отличных от исследованных, поэтому полностью полагаться на него нельзя.

По этим причинам, для некоторых чипов может быть более эффективной реализация атаки с использованием не выходов S-box, а выхода раунда. Это вполне, возможно так как в шифре «Магма», на конечной операции каждого раунда, производится сложение по mod2 результата циклического сдвига (часть которого можно рассчитать с использованием небольшой части раундового ключа) с известной левой частью входа раунда. Исследования, описанные во второй главе, показали, что операция сложения по mod2 приводит к значительной утечке секретной информации. Поэтому использование её результата может быть более эффективным при реализации DPA, нежели использование выходов S-box. Кроме того, в большинстве известных ассемблеров, результат сложения содержимого двух регистров по mod2 неминуемо сохраняется в одном из этих регистров. Заполнение обоих регистров злоумышленнику очевидно известны и поэтому использование модели расстояния Хэмминга не составит труда.

Анализ структуры L-преобразования приводит к очевидному выводу, что окажется невозможным рассчитать даже малую часть (положим четыре или восемь бит) выхода первого раунда шифра «Кузнечик», не предполагая всех 128-ми бит его раундового ключа. Перебор 128-ми бит целиком в процессе реализации DPA, очевидно окажется невозможным и

противоречащим основной идее DPA – поэтапному перебору небольших частей раундового ключа. Однако, в данном случае, в качестве промежуточного значения можно использовать те же выходы S-box, сдвигаемые в ЛРР. Так как сдвиг происходит ровно на байт, то они также будут зависеть от восьми бит перебираемого подблока ключа. Кроме этого, операция копирования байта из регистра в регистр (при помощи многократного выполнения которой очевидно будет реализован сдвиг) может приводить к б**о**льшим утечкам информации по цепи электропитания чем операция чтения из ПЗУ. Помимо того, в данном случае, вполне определённо предполагаются начальные заполнения регистров назначения, и это значения младших байт комбинации, полученной с выхода S-box, на место которых и сдвигаются старшие байты, что делает несложным использование модели расстояния Хэмминга. Наконец, очевидно, что в качестве промежуточного значения возможно использование аргументов функции $l(a_{15},\ ...,\ a_1,\ a_0)$.

Необходимо также обратить внимание на тот факт, что даже в случае использования в качестве промежуточного значения выходов S-box, на ложных вариантах перебираемого подблока ключа возможно появление значительных корреляционных всплесков, в области форм сигнала, характеризующей энергопотребление при реализации L-преобразования, так возможна некоторая зависимость определённых промежуточных значений L-преобразования и выходов S-box, полученных на истинном варианте перебираемого подблока ключа. Хотя существование сильной зависимости маловероятно, однако, в любом случае, учитывая сложность этого преобразования, данную возможность исключать нельзя.

В целом эффективность реализации атаки DPA с использованием в качестве промежуточного значения комбинаций, следующих за преобразованиями на S-box в большей степени зависит от конкретной реализации обоих шифров и требует более детальных исследований путём моделирования или лабораторного эксперимента с реальным чипом.

Необходимо отметить, что шифры «Магма» и «Кузнечик» отличаются размерами блоков сообщения и ключа раунда – в «Магме» на каждом раунде расходуется 32 бита секретного ключа, для восстановления 256-ти бит потребуется последовательно совершить атаку на восемь раундов, восстанавливая на каждом раунде по 32 бита. Ключ «Кузнечика» расходуется на двух раундах – 128 бит на первом и столько же на втором. Как говорилось выше, в случае восстановления раундового ключа одного раунда, не составляет труда для всех преобразуемых сообщений, рассчитать вход следующего раунда, и затем реализовать атаку в отношении него. В данном случае можно сделать вывод, что отличия в длине блоков сообщения и ключа двух рассматриваемых шифров, а также схемы распределения секретного ключа по раундам не окажут сколько-нибудь значимого влияния на сложность реализации DPA.

На основе приведённых выше доводов можно сделать вывод, о том, что эффективность реализации DPA в отношении шифров «Магма» и «Кузнечик» будет отличаться, шифр «Магма» можно считать несколько более устойчивым к DPA, чем «Кузнечик», по большей части по причине способа практической реализации S-box «Магмы». Однако важно оговориться, что, используемая при реализации DPA измерительная установка, особенности энергопотребления анализируемого чипа, а также детали конкретной реализации на нём защитного алгоритма, могут оказывать значительно большее влияние на эффективность реализации DPA, чем отличия алгоритмов этих шифров.

3.6 Экспериментальное исследование реализации DPA применительно к шифру «Магма»

В этом разделе будут описаны принципы реализации раунда шифра «Магма» на анализируемом микроконтроллере, описана реализация в отношении него атаки DPA (с использованием экспериментальных данных), а также дана оценка сложности выполнения атаки.

3.6.1 Особенности реализации шифра «Магма» на анализируемом микроконтроллере

Для рассматриваемого шифра, каждый из 32-х его раундов можно описать в виде четырёх групп преобразований:

1. Арифметическое сложение 32-ух битовой правой части входа раунда $R(i,\ r)$ с раундовым ключом $k(r)$: $R(i,\ r)+k(r)$, где r – номер раунда, $i=1,\ ...,\ N$ – преобразуемое сообщение. Разбивка полученной суммы на 4-х битовые подблоки $S_{\mathrm{IN},\,l}(i,\ r)$, где $l=1,\ ...,\ 8$.
2. Преобразование полученных подблоков на восьми S-box: $S_{\mathrm{OUT},\,l}(i,\ r)=S_l\left(S_{\mathrm{IN},\,l}(i,\ r)\right)$, где $S_l(\cdot)$ – функция преобразования на l-м S-box, $S_{\mathrm{OUT},\,l}(i,\ r)$ – выход l-го S-box.
3. Циклический сдвиг последовательности, образованной выходами восьми S-box, на 11 разрядов влево.
4. Сложение полученной комбинации с левой частью входа раунда $L(i,\ r)$.

Ниже будет приведено краткое описание реализации каждого преобразования на примере первого раунда «Магмы». На предварительном этапе выполнения алгоритма, ПК, через COM-порт, последовательно отсылает на приёмник UART микроконтроллера восемь байт сообщения. Каждый принятый микроконтроллером байт сохраняется в ОЗУ, после приёма восьмого байта, микроконтроллер завершает сеанс приёма и приступает к выполнению алгоритма шифра.

Сначала каждый из восьми байт сообщения выгружается в восемь младших рабочих регистров: R1-R8 (в первом регистре R1 находятся восемь старших байт сообщения, а в восьмом R8 – восемь младших). Следует напомнить, что АЛУ микроконтроллера может совершать арифметические и логические операции лишь с теми данными, которые находятся в рабочих регистрах, всего в микроконтроллере ATmega16 32 рабочих регистра, каждый объёмом в один байт. Дополнительно, содержимое 5-8 регистров (правая часть входа раунда) копировалось в 22-25 регистры – данный шаг был необходим для того, чтобы в конце раунда заменить этими данными содержимое 1-4 регистров, обеспечив таким образом предусмотренную схемой Фейстеля перестановку правой и левой частей входа раунда.

Затем в 9-13 регистры из Flash памяти последовательно считываются 4 байта раундового ключа $k_l(r)$, где $l = 1, \ldots, 4$ – номер подблока ключа ($k_1(r)$ – младший подблок, $k_4(r)$ – старший), $r = 1, \ldots, 32$ – номер раунда. При этом в девятом регистре находятся восемь старших байт раундового ключа (подблок $k_4(1)$), а в 13-м – восемь младших (подблок $k_1(1)$).

После этого осуществляется побайтное сложение (начиная с младшего байта) восьми битовых подблоков правой части входа раунда $R_l(i, 1)$ с восьми битовыми подблоками раундового ключа $k_l(1)$, результат (в виде четырёх восьми битовых подблоков) перезаписывает содержимое 5-8 регистров. То есть вначале содержимое восьмого регистра $R_1(i, 1)$, складывается с содержимым 13-го $k_1(1)$, результат перезаписывает содержимое восьмого регистра, если при сложении восьмых разрядов возникал перенос байта в девятый, то соответствующий бит регистра статуса (бит переноса) устанавливался в единицу. При сложении вторых подблоков сообщения $R_2(i, 1)$ и ключа $k_2(1)$, используемая команда ассемблера учитывает содержимое бита переноса регистра статуса, поэтому, возникший в первом подблоке перенос, перейдёт во второй. Результат сохраняется в седьмой регистр перезаписывая комбинацию $R_2(i, 1)$. После сложения третьего и четвёртого подблоков сообщения с соответствующими подблоками раундового ключа содержимое 5-8-го регистров будет представлять собой 32-х битовую комбинацию $S_{\text{IN}}(i, 1) = R(i, 1) + k(1)$.

Далее, согласно алгоритму шифрования, комбинация $S_{\text{IN}}(i,\ 1)$ должна быть разбита на четырёх битовые подблоки, над каждым из которых совершается табличное нелинейное преобразование, описываемое таблицей соответствующего S-box. Ранее уже объяснялась (приложение Ж) неэффективность реализации четырёх битовых преобразований на чипах с восьми и более разрядной архитектурой. Поэтому согласно представленному в приложении Ж алгоритму производилось преобразование четырёх полученных восьми битовых подблоков $S_{\text{IN},\,l}(i,\ 1)$ на сдвоенных S-box.

Преобразование на S-box осуществлялось за три шага:

- В 31-32 регистры (так называемую регистровую пару ZL, ZH или упрощённо: регистр Z) загружается адрес ячейки Flash памяти, в которой записан первый байт таблицы преобразования, соответствующего сдвоенного S-box (алгоритм записи таблиц преобразования сдвоенных S-box подробно представлен в приложении Ж).
- Осуществляется арифметическое сложение содержимого 31-32-го регистров со значением $S_{\text{IN},\,l}(i,\ 1)$ из соответствующего регистра, результат сохраняется в регистре Z. Таким образом в регистре Z располагается адрес первой строки таблицы преобразования S-box со смещением на величину $S_{\text{IN},\,l}(i,\ 1)$ (от 0 до 255-ти).
- Соответствующей командой ассемблера производится чтение из Flash памяти по адресу, содержащемуся в регистре Z с сохранением результата в регистре, содержащим значение $S_{\text{IN},\,l}(i,\ 1)$.

Приведём пример, для байта $S_{\text{IN},\,1}(i,\ 1)$ (находящегося в восьмом регистре – R8), положим, что он равен $S_{\text{IN},\,1}(i,\ 1)=1111\,1111=255$. Положим, что таблица преобразования 7-8 S-box записана в 120-375 ячейки Flash памяти таким образом, что в 120 ячейке находится выход 7-8 S-box при входе «0000 0000», в 121-й ячейке – выход S-box при входе «0000 0001», и так далее, вплоть до 375-й ячейки содержимое которой соответствует выходу S-box при входе «1111 1111»

На первом этапе в регистры ZH и ZL загружается адрес первой ячейки S-box, то есть 120, таким образом содержимое регистра ZH будет равно «0000 0000», содержимое регистра ZL – «0111 1000» (=120).

На втором этапе производится сложение регистров ZL и R8 с сохранением результата в регистре ZL – в итоге содержимое регистра ZL перезаписывается комбинацией: $0111\,1000+1111\,1111=0111\,0111(=119)$ при этом происходит перенос бита в соседний (девятый) разряд (то есть флаг переноса в регистре статуса устанавливается в единицу). Этот перенос

можно учесть путём сложения содержимого регистра ZH с регистром, содержащим нулевой байт (в качестве такого, выбирался регистр R16, который предварительно очищался): ZH+R16=0000 0000[1]+0000 0000=0000 0001(=1), в итоге в сдвоенном регистре Z оказывается записанной комбинация: 0000 0001 0111 0111 (=375).

Затем, соответствующей командой ассемблера, осуществляется чтение по адресу Flash памяти, содержащемуся в регистре Z, то есть из 375-й ячейки, считывается комбинация «1111 0001», которая является выходом 7-8 S-box при входе «1111 1111», результат сохраняется в регистр R8. Проверка по таблице преобразования позволит удостовериться, что результат верный и описанный алгоритм работает корректно.

Таким образом осуществляется преобразование всех четырёх байт суммы $R(i,\ 1) \oplus k(1)$, в четыре байта $S_{\text{OUT},\,l}(i,\ 1) = S_l\left(S_{\text{IN},\,l}(i,\ 1)\right) = S_l\left(R(i,\ 1) + k(1)\right)$, для $l = 1,\ ...,\ 4$.

Затем над комбинацией $S_{\text{OUT}}(i,\ r)$ производится операция циклического сдвига на 11 разрядов влево. Команды, доступные на ассемблере анализируемого микроконтроллера позволяют за один подход производить сдвиг содержимого лишь одного регистра, на один разряд. Поэтому потребовалось совершить 11 проходок сдвига каждого из четырёх регистров: R5–R8.

Затем полученная комбинация (содержащаяся в регистрах R5–R8), побайтно, складывается по mod2 с левой частью входа раунда (содержащейся в регистрах R1–R4). Результат сохраняется в регистры R5–R8 – полученная комбинация является правой частью входа второго раунда. Содержимое 22-25 регистров копируется в 1-4 регистры – таким образом, в соответствии со схемой Фейстеля, осуществлялась перестановка правой и левой частей входа раунда (правая часть входа первого раунда переходит на место левой части входа второго раунда).

3.6.2 Оценка эффективности реализации DPA в отношении шифра «Магма», с использованием экспериментальных данных

В этом подразделе будет описана атака DPA, в отношении представленной выше, реализации алгоритма шифра «Магма». По своей сути она не отличается от атаки, применяемой в отношении отдельных операций, описанной в разделе 2.5.

На первом этапе производится получение данных об энергопотреблении микроконтроллера. Программное обеспечение ПК через COM-порт отсылает на микроконтроллер случайные 64-ёх битовые двоичные комбинации, с заданным интервалом. Над

каждой принятой микроконтроллером комбинацией совершаются преобразования, описанные в предыдущем подразделе. При этом, перед преобразованием на восьми S-box микроконтроллер генерирует триггерный сигнал, запускающий АЦП. АЦП, приняв триггерный импульс записывает данные об энергопотреблении и копирует их в заданную папку на ПК, длина каждой формы сигнала составляет 6 000 отсчётов, что является достаточным для того, чтобы записать энергопотребление в процессе совершения преобразований на каждом из четырёх сдвоенных S-box. ПО, отсылающее сообщение на COM-порт добавляет в название файла (формы сигнала), полученного от АЦП, сообщение при шифровании которого была получена соответствующая форма сигнала. На сбор пяти тысяч форм сигнала, при помощи описанной измерительной установки затрачивается порядка получаса.

На втором этапе, после накопления заданного количества форм сигнала, происходит их выравнивание. Реализация данного этапа для пяти тысяч форм сигнала также требует порядка получаса.

На третьем этапе реализуется алгоритм DPA, описанный в подразделах 1.7.1 и 2.5.2, при этом, в качестве промежуточного значения, используются выходы S-box, которые рассчитываются по известному для каждой формы сигнала сообщению и перебираемому восьми битовому подблоку раундового ключа. Таким образом для каждого S-box рассчитывается 256 корреляционных векторов – по одному на каждый вариант перебираемого подблока ключа от 0 до 255 (на данные расчёты по каждому сдвоенному S-box требовалось порядка 10 минут, в сумме на восстановление ключей по выходам четырёх S-box затрачивается порядка сорока минут). Каждый вектор характеризует зависимость между весом Хэмминга выхода S-box рассчитанного на известном сообщении и варианте перебираемого подблока ключа и каждым отсчётом соответствующих форм сигнала. По большому счёту в данном случае определяется успешность реализации DPA к операции чтения из Flash памяти, и в принципе результаты атаки должны повторять результаты, полученные в подразделе 2.5.3, где операция чтения выхода S-box из Flash памяти рассматривалась отдельно. Отметим, что выход S-box, считываемый из Flash памяти в регистр, в данном случае не перезаписывал содержимое соответствующего регистра (регистр предварительно очищался), потому что, как показано в подразделе 2.5.3, успешная реализация DPA с использованием промежуточного значения, считываемого из ПЗУ, возможна лишь в том случае, если считывание происходит в чистую ячейку регистра.

На рисунке 3.17а представлен корреляционный вектор, полученный, после обработки пяти тысяч форм сигнала на истинном варианте перебираемого подблока ключа, при реализации атаки, с использованием в качестве рассчитываемого промежуточного значения, выхода 7-8 сдвоенного S-box. Всплеск имеет наибольшую амплитуду, всплески, полученные на остальных 255-ти вариантах перебираемого подблока ключа имеют не менее чем на 10-15%

меньшую амплитуду, например, корреляционный вектор, рассчитанный на наихудшем для злоумышленника подблоке ключа представлен на рисунке 3.17б.

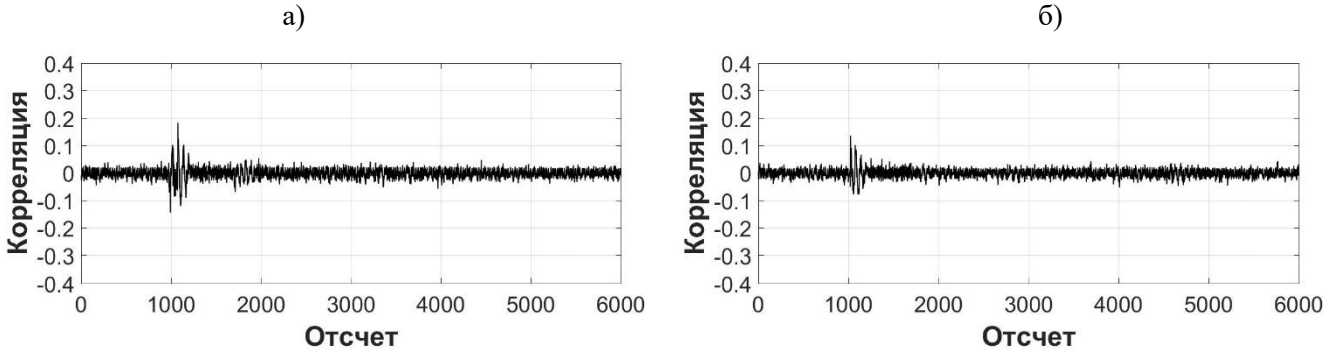


Рисунок 3.17 – Корреляционные вектора для случая, когда в качестве рассчитываемого промежуточного значения, выбирается выход 7-8 S-box, на: а) истинном варианте перебираемого подблока ключа; б) наихудшем для злоумышленника ложном варианте перебираемого подблока ключа

На рисунках 3.18 изображены корреляционные векторы полученные при реализации атаки с использованием, в качестве рассчитываемого промежуточного значения, выхода 5-6 сдвоенного S-box, из результатов видно, что всплеск сместился на 800 отсчётов вправо, что вполне логично, так как преобразования на S-box осуществляются последовательно друг за другом.

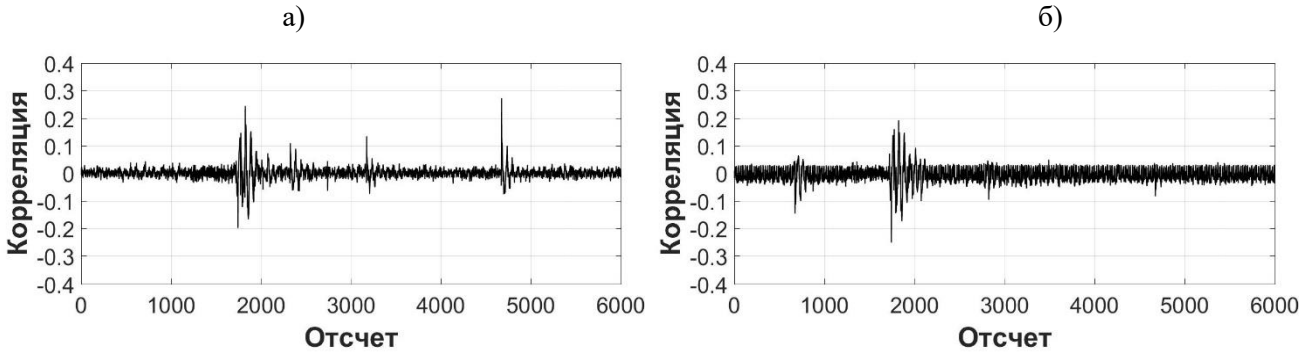


Рисунок 3.18 – Корреляционные вектора для случая, когда в качестве рассчитываемого промежуточного значения, выбирается выход 5-6 S-box, на: а) истинном варианте перебираемого подблока ключа; б) наихудшем для злоумышленника ложном варианте перебираемого подблока ключа

Из представленных графиков видно, что всплески на корреляционных векторах оказываются заметно ниже, чем при отдельном рассмотрении операции чтения из Flash памяти. Это связано с выявленным ранее эффектом остаточной корреляции – так, в данной реализации все команды микроконтроллера следуют строго друг за другом, без временных пропусков, следовательно, на колебания напряжения, обусловленные выполнением команды чтения из Flash памяти, оказывают влияние, колебания, вызванные выполнением одной или нескольких

предыдущих и последующих команд, вследствие чего значительно снижается соотношение сигнал/шум в отсчётах, характеризующих энергопотребление чипа при выполнении операции чтения из Flash памяти. Эти доводы найдут чёткое подтверждение в представленном ниже рассмотрении операции записи в ОЗУ.

Результат преобразования на каждом S-box после считывания в соответствующий регистр может пересылается затем в ячейки ОЗУ. Эта операция может потребоваться на моделях чипов с ограниченным количеством регистров, многие модели имеют 16 или восемь регистров, поэтому промежуточные данные просто необходимо пересылать на временное хранение в ОЗУ. Для моделирования подобной ситуации на четвёртом шаге была добавлена операция пересылки каждого, считанного из Flash памяти, выхода S-box в чистую ячейку ОЗУ.

На рисунке 3.19а изображён корреляционный вектор полученный на истинном варианте перебираемого подблока ключа (для 7-8 сдвоенного S-box). Корреляционный всплеск ожидаемо имеет значительную амплитуду. На ложных вариантах ключа амплитуды всплесков оказываются значительно меньше (рисунок 3.19б).

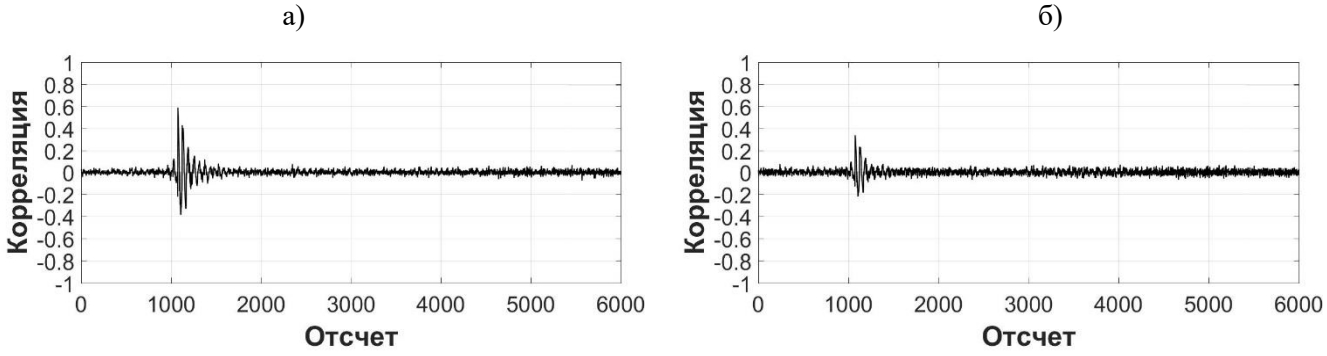


Рисунок 3.19 – Корреляционные вектора для случая, когда в качестве рассчитываемого промежуточного значения, выбирается выход 7-8 S-box, который дополнительно записывался в чистую ячейку ОЗУ, на: а) истинном варианте перебираемого подблока ключа; б) наихудшем для злоумышленника ложном варианте перебираемого подблока ключа

Однако, на истинном варианте перебираемого подблока ключа корреляционные всплески всё равно оказываются заметно ниже, чем при отдельном рассмотрении операции записи в ОЗУ. Если расположить до операции записи в ОЗУ (то есть после команды чтения из Flash памяти в регистр, но перед командой записи содержимого регистра в ОЗУ) и после неё по три команды «nop» (то есть пропуск операции, пустая команда) то амплитуда всплеска на корреляционном векторе, рассчитанном на истинном варианте перебираемого подблока ключа, будет практически идентична, той, что получена при отдельном рассмотрении операции записи в ОЗУ. Данный опыт подтверждает выдвинутую ранее гипотезу, что эффект остаточной корреляции негативно влияет на успешность реализации DPA.

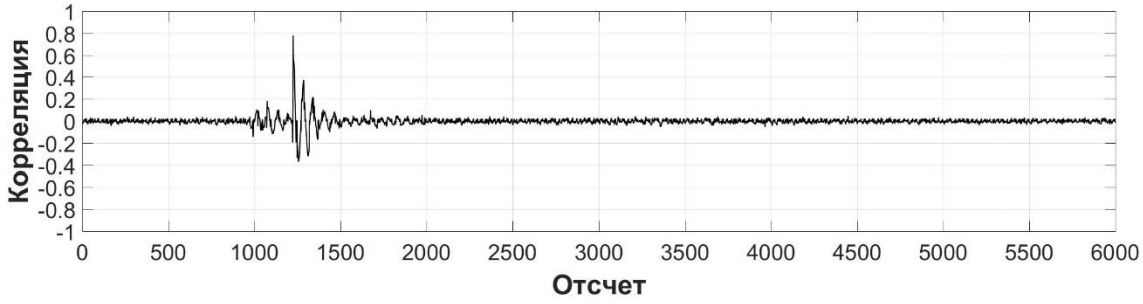


Рисунок 3.20 – Корреляционный вектор, полученный на истинном подблоке ключа для случая, когда в качестве рассчитываемого промежуточного значения, выбирается выход 7-8 S-box, который дополнительно копировался в чистую ячейку ОЗУ, перед и после операции копирования выполнялись по три команды «nop»

Также были получены экспериментальные данные с использованием в качестве рассчитываемого промежуточного значения других (рассмотренных ранее) часто применяемых в защитных алгоритмах команд (перемещение данных между регистрами, сложение по mod2, арифметическое сложение). В результате чего подтверждена тенденция, что рассмотрение команды не отдельно, а в контексте реализации полноценного защитного алгоритма (когда до и после рассматриваемой команды выполняются другие команды с другими данными) снижает амплитуды всплесков на полученных корреляционных векторах. Однако уменьшение амплитуд всплесков происходит как для истинного, так и для ложных перебираемых вариантов подблока ключа, поэтому вероятность верного определения ключа остаётся практически неизменной (лишь требуется увеличение количества обрабатываемых форм сигнала для более чёткого различения всплесков на фоне шума).

3.7 Выводы по главе

Учитывая результаты, приведённых выше теоретических выкладок, моделирования, а также практической реализации DPA можно сделать следующие основные выводы:

- Подходы с расчётом ковариационных и корреляционных векторов являются эффективными на практике. Прямо пропорциональная зависимость всплесков ковариационного вектора от размерностей отсчётов форм сигнала, весьма сильно затрудняет успешную реализацию DPA, в результате чего требуется обработка значительно большего количества форм сигнала по сравнению с корреляционными векторами. При расчёте корреляционных векторов подобный эффект также проявляется, но оказывает существенно меньшее влияние,

поэтому потребуется обработка значительно меньшего количества форм сигнала. В итоге эффективность реализации DPA с расчётом корреляционных векторов оказывается на порядок выше.

- Наиболее значительные отличия эффективности реализации DPA в отношении шифров «Магма» и «Кузнечик», обусловлены различиями в практической реализации преобразований на их S-box. Поэтому, если учитывать особенности реализации алгоритма шифра «Магма» на реальном чипе, то реализация DPA в отношении этого шифра окажется несколько сложнее, нежели чем в отношении «Кузнечика» – на практике шифр «Магма» окажется более устойчивым к DPA.
- DPA принципиально реализуема в отношении полноценных защитных алгоритмов. В процессе реализации атаки не требуется детального изучения получаемых данных об энергопотреблении – алгоритм сам «находит» области на формах сигнала, коррелированные с обрабатываемыми данными. Поэтому атака весьма просто автоматизируется и успешно применяется с минимальными знаниями об анализируемом чипе (конечно в том случае если чип не имеет специальной защиты, в противном случае атака также может быть автоматизирована, но после детального изучения чипа и нахождения путей преодоления методов защиты, что неоднократно подтверждалось на практике в отношении существующих коммерческих чипов [41; 44; 57; 59]).
- Явление остаточной корреляции несколько снижает вероятность успешной реализации DPA, однако катастрофического влияния на результат не оказывает.
- Реализация DPA возможна с использованием оборудования общей стоимостью в $2-3 тыс., при этом для шифра «Магма» восстановление секретного ключа (что сводится к восстановлению восьми раундовых ключей) занимает порядка 6-7 часов. Стоит подчеркнуть, что однажды приобретённое оборудование может быть использовано для анализа неограниченного количества различных чипов (в том числе различного форм-фактора), реализующих множество защитных алгоритмов. Поэтому данный тип атак имеет крайне низкую стоимость в пересчёте на каждое взломанное устройство.

Основное содержание раздела и полученных научных результатов изложено в работах автора [15; 23; 28; 30; 31; 32].

ГЛАВА 4. РАЗРАБОТКА УНИВЕРСАЛЬНОГО МЕТОДА ЗАЩИТЫ ОТ DPA

4.1 Обоснование необходимости разработки нового метода защиты от DPA

В разделе 1.8 уже была приведена краткая характеристика основных подходов к защите выполняемых на чипах преобразований от DPA. В частности, было показано, что обеспечить доказанный теоретически уровень защиты от побочных атак по цепям электропитания на практике возможно при помощи одного аппаратного метода защиты: аппаратной балансировки, и двух программных: программная балансировка и маскировка. Остальные методы защиты по физическим причинам не позволяют полностью устранить зависимости между обрабатываемыми чипом данными и его энергопотреблением. Они способны лишь затруднить применение атаки. Более того, б**о**льшая часть из них требует серьёзных доработок аппаратной части чипа – в конечном итоге их реализация является весьма затратной.

В свою очередь, основным недостатком метода аппаратной балансировки можно считать высокую стоимость его реализации, за счёт необходимости использования чипов, на кристалле которых размещены сразу два физических процессора. Поэтому одновременно дешёвыми в реализации и надёжными методами защиты являются лишь методы программной балансировки и маскировки. При этом их основным недостатком можно считать плохую унифицированность относительно защищаемых алгоритмов. Эти методы изначально разрабатывались для защиты американского стандарта шифрования AES и, к примеру, прямая адаптация маскировки к российскому шифру ГОСТ 28147-89 (по аналогии и к шифру «Магма»), потребует чрезвычайно большого объёма дополнительного ПЗУ для хранения таблиц защищённых S-box [25] (аналогичная ситуация будет наблюдаться и с шифром «Кузнечик»). В разделах 4.2, 4.3 будет показано, что метод программной балансировки, в свою очередь, также нельзя напрямую адаптировать к шифрам «Магма» и «Кузнечик». Отсюда очевидна актуальность вопроса разработки дешёвого и надёжного метода защиты шифров ГОСТ.

В данной главе будут более подробно изложены принципы построения таких программных методов защиты как маскировка и программная балансировка. Затем будут показаны сложности адаптации балансировки к шифрам ГОСТ. После чего, предложен новый подход к защите преобразований на интегральных чипах от DPA под названием универсальный метод защиты (УМЗ). И наконец подробно рассмотрено построение УМЗ.

Подход, на котором базируется предлагаемый метод защиты от DPA обладает чрезвычайно широкой универсальностью, как относительно защищаемых чипов, так и используемых в них алгоритмов (он эффективен по отношению к любым блочным шифрам, в том числе к шифрам ГОСТ). Причём предъявляются низкие требования к дополнительным аппаратным ресурсам чипа, обеспечивается простота и невысокая стоимость реализации. Предложенная реализация УМЗ ограничивается лишь изменением ассемблерного кода, описывающего алгоритм защищаемого преобразования, поэтому может быть реализована не только на специализированных чипах, но даже на типовых микроконтроллерах.

4.2 Краткая характеристика программных методов защиты от DPA

4.2.1 Описание метода маскировки

Целью любого метода защиты от DPA является устранение зависимости энергопотребления чипа от обрабатываемых им данных. Классическим программным методом устранения этой зависимости можно считать маскировку [10, с. 223-244; 25]. Маскировка предполагает наложение на промежуточные значения выполняемого алгоритма некоторой случайной последовательности (вырабатываемой генератором случайных чисел). Как правило под наложением подразумевается операция сложения по mod2. В качестве примера приведём простейшую операцию – защищённое маскировкой сложение восьми бит сообщения с раундовым ключом. Положим, на первом раунде какого-либо шифра существуют два восьми битовых подблока – сообщение $m_l(i)$, являющееся l-й частью блока сообщения $m(i)$, подающегося на вход шифра, и соответствующая часть раундового ключа первого раунда $k_{l,1}$. Эти две комбинации, согласно алгоритму шифра, должны быть сложены между собой по mod2. Злоумышленник может подавать на чип любые сообщения, то есть подблок $m_l(i)$ ему известен (он полностью контролируется злоумышленником), а подблок ключа ему неизвестен, целью злоумышленника является восстановление ключа.

При незащищённой реализации преобразований подблок сообщения просто складывается с подблоком ключа соответствующей командой ассемблера (рисунок 4.1)

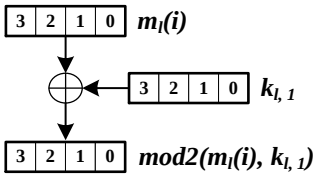


Рисунок 4.1 – Незащищённая реализация операции сложения по mod2

Защищённая маскировкой реализация оказывается несколько сложнее (рисунки 4.2):

1. Подблок сообщения $m_l(i)$ складывается по mod2 с подблоком случайной маски x_1 (рисунок 4.2а): $m_l(i) \to m_l(i) \oplus x_1$
2. Подблок ключа $k_{l,1}$ складывается по mod2 с суммой по mod2 двух масок: $x_1 \oplus x_2$, где x_1 – та же маска, которая использовалась для маскировки сообщения, а x_2 – вторая маска, не зависящая от x_1 (рисунок 4.2б): $k_{l,1} \to k_{l,1} \oplus x_1 \oplus x_2$
3. Замаскированные сообщение и ключ складываются по mod2 друг с другом (рисунок 4.2в): $\left(m_l(i) \oplus \cancel{x_1}\right) \oplus \left(k_{l,1} \oplus \cancel{x_1} \oplus x_2\right) = m_l(i) \oplus k_{l,1} \oplus x_2$

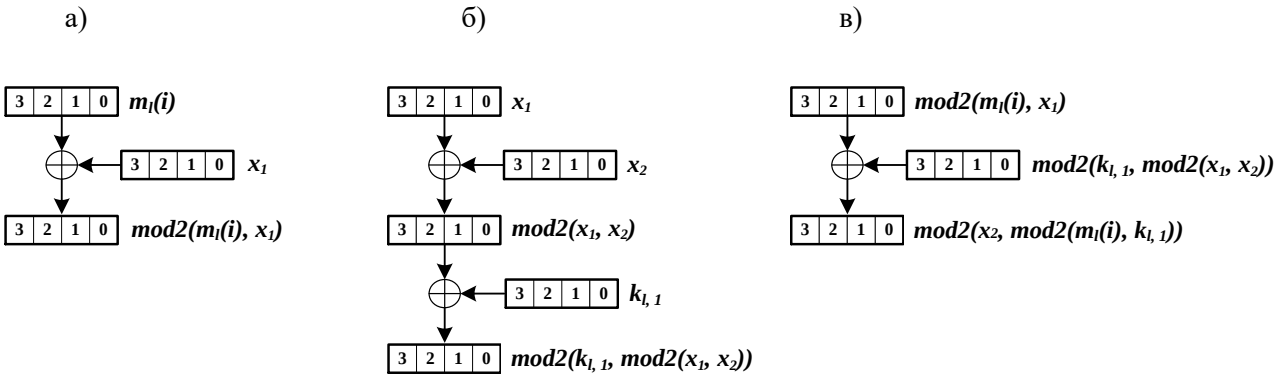


Рисунок 4.2 – Защищённая маскировкой реализация операции сложения по mod2: а) маскировка подблока сообщения; б) маскировка подблока ключа; в) защищённая маскировкой операция сложения подблоков сообщения и ключа

Из рисунка 4.2 видно, что сумма подблоков сообщения и ключа всегда оказывается сложенной по mod2 со случайной комбинацией x_2. В свою очередь комбинация x_2 вырабатывается генератором случайных чисел и не зависит ни от подблока сообщения ни от подблока ключа. Положим, что злоумышленник совершает атаку DPA, а в качестве рассчитываемого, на перебираемом варианте подблока ключа, промежуточного значения шифра выбирает выход суммы сообщения и ключа. В этом случае, если вариант подблока ключа будет угадан верно, то сумма $m_l(i) \oplus k_{l,1}$ для всех вариантов $m_l(i)$ будет рассчитана верно, но она

очевидно не будет иметь зависимости с соответствующими отсчётами на формах сигнала, так как они будут характеризовать энергопотребление, затрачиваемое на обработку комбинации $m_l(i) \oplus k_{l,1} \oplus x_2$.

Из приведённого примера метод маскировки кажется достаточно надёжным и простым в реализации. Однако, к примеру, реализация защищённого таким образом табличного преобразования на S-box в большинстве случаев оказывается весьма трудоёмкой, по той причине, что в таблицах S-box необходимо учитывать все возможные варианты маски x_2, более того, для защиты от атак DPA высокого порядка (High-Order DPA [10, с. 245-272; 25; 35]), выход S-box необходимо маскировать другой маской x_3 (не зависящей от маски x_2). Это в свою очередь, приводит к существенному увеличению объёма таблицы преобразования, защищённого маскировкой S-box (следовательно, и требуемого объёма ПЗУ чипа для её хранения), поэтому потребуется серьёзная модификации аппаратной части чипа. Вопросы контроля накладываемых масок, и их своевременной замены (на разных операциях алгоритма шифра) оказывается серьёзной проблемой для разных блочных шифров.

Справедливо отметить, что в шифре AES, существует возможность реализации преобразования на S-box как вычисление обратного элемента в поле $GF\left(2^8\right)$. Было предложено несколько подходов, позволяющих использовать данную особенность S-box AES для значительного упрощения реализации маскировки. Например, в работе [52] предлагается метод, который в принципе не требует выделения дополнительного объёма памяти чипа под хранение таблиц преобразований защищённых S-box. При этом не уменьшается скорость шифрования по сравнению с незащищённой версией. Вместе с этим, ранее проведённые исследования автора относительно шифра ГОСТ 28147-89, показали, что прямая адаптация реализации маскировки потребует значительных усложнений реализации шифра [25]. За счёт того, что S-box «Кузнечика» аналогично «Магме» задаются таблицей, адаптация маскировки применительно к нему также потребует значительного объёма памяти под хранения таблиц преобразований защищённых S-box. При этом, на основе анализа изученной литературы, можно сделать вывод что на сегодняшний день пока отсутствуют более «изящные» и менее требовательные к ресурсам чипа, способы адаптации маскировки к шифрам ГОСТ. Таким образом, основным недостатком маскировки является небольшой набор преобразований, которые могут быть ей защищены (по крайней мере с использованием ограниченных аппаратных ресурсов микрочипа), а, следовательно, ограниченное количество шифров, доступных для защиты этим методом. Кроме этого, для генерации случайной маски необходимо наличие смонтированного на кристалле чипа высококачественного генератора случайных чисел – такие чипы безусловно имеют б**о**льшую стоимость.

4.2.2 Описание метода программной балансировки

Альтернативой маскировке можно считать метод программной балансировки, его идея основана на особенностях архитектуры современных микрочипов – многоразрядности. Как неоднократно говорилось ранее, это означает, что все основные узлы чипа на каждом такте работы одновременно способны обрабатывать количество бит, равное разрядности чипа. Чрезвычайно распространёнными являются восьми разрядные микрочипы (хотя в последние годы приобретают популярность чипы с 16-ти, и даже 32-ух разрядной архитектурой). При этом, как показывалось в подразделе 2.5.3 (в частности, на рисунках 2.36 и в пояснениях к нему), амплитуда всплеска энергопотребления на снимаемых формах сигнала, будет изменяться пропорционально весу Хэмминга обрабатываемой комбинации (в том случае, если эта комбинация записывается в чистую ячейку памяти, в противном случае расстоянию Хэмминга между записываемой и перезаписываемой комбинациями). Таким образом на формах сигнала невозможно будет выделить отдельные скачки энергопотребления, обусловленные выработкой конкретного бита обрабатываемой комбинации. Поэтому программная балансировка предполагает поддержку постоянного количества единиц в обрабатываемых чипом комбинациях. При реализации балансировки на чипах с восьмиразрядной архитектурой, каждый обрабатываемый байт представляется двумя четырёх битовыми частями (ниббламы) – информационной и балансирующей. Информационная часть, является непосредственно частью обрабатываемой информации, балансирующая часть зависит от информационной части, и предназначена для поддержки одинакового количества единиц в обрабатываемом байте. Простейшим способом балансировки байта является использование в качестве балансирующего ниббла комбинации инвертированной относительно комбинации, составляющей информационный ниббл – такой подход обеспечит постоянное количество единиц в «склеенном» байте при любых возможных значениях информационных бит, а, следовательно, энергопотребление чипа будет постоянным при любом значении информационного ниббла. (По таблице 4.1 несложно проверить, что на любой из 16-ти возможных комбинаций бит информационных разрядов, балансирующие разряды будут принимать такие значения, что вес Хэмминга сбалансированного байта всегда будет равен четырём.) Авторами этого метода в 2013 году была предложена защищённая структура преобразований шифра AES, которая может быть выполнена на типовом 16-ти разрядном чипе [40]. В качестве примера работы балансировки, ниже будут рассмотрены защищённые операции сложения по mod2 подблоков сообщения и ключа, а также табличного нелинейного преобразования (S-box).

Таблица 4.1 – Возможные сочетания информационного и балансирующего ниббл, а также вес Хэмминга байта, «склеенного» их этих двух ниббл, при выборе в качестве балансирующего ниббла комбинации являющейся инвертированной относительно информационного ниббла

Вес Хэмминга «склеенного» байта	Информационный ниббл	Балансирующий ниббл
4	0000	1111
4	0001	1110
4	0010	1101
4	0011	1100
4	0100	1011
4	0101	1010
4	0110	1001
4	0111	1000
4	1000	0111
4	1001	0110
4	1010	0101
4	1011	0100
4	1100	0011
4	1101	0010
4	1110	0001
4	1111	0000

Для защиты от DPA операции сложения двух байт по mod2, в комбинации, представляющей результат сложения, необходимо обеспечить сбалансированность количества единиц, поэтому один ниббл (к примеру младший) следует выбрать информационным, а другой (старший) ниббл – балансирующим, а, следовательно, его значение должно быть инвертированным по отношению к информационному (рисунок 4.3).

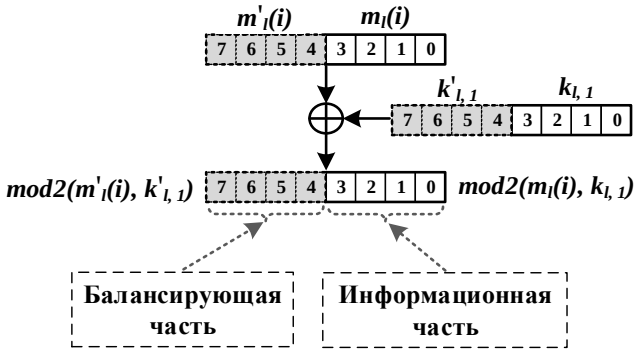


Рисунок 4.3 – Балансировка результата операции сложения по mod2

Отсюда понятно, что младшие нибблы сообщения и ключа, также должны быть информационными. Обозначим младшие нибблы сообщения, ключа и результата сложения как соответственно: $m_l(i)$, $k_{l,r}$, $t_{l,r}(i)$ (где i – номер итерации, r – номер раунда, l – номер подблока), а старшие нибблы как: $m'_l(i)$, $k'_{l,r}$, $t'_{l,r}(i)$, тогда комбинации $t_{l,r}(i)$ и $t'_{l,r}(i)$ будут получаться следующим образом (4.1)-(4.2):

$$t_{l,r}(i) = m_l(i) \oplus k_{l,r}, \tag{4.1}$$

$$t'_{l,r}(i) = m'_l(i) \oplus k'_{l,r}, \tag{4.2}$$

При этом необходимо найти такие значения $m'_l(i)$ и $k'_{l,r}$, при которых будет выполняться следующее неравенство:

$$t'_{l,r}(i) = \overline{t_{l,r}(i)}, \tag{4.3}$$

где $\overline{\cdot}$ – операция инвертирования бит (например если $t'_{l,r}(i) = 0001$, тогда $\overline{t_{l,r}(i)} = 1110$).

Известно, что инвертирование любой двоичной комбинации можно осуществить путём её сложения по mod2 с единичным вектором (это свойство операции сложения по mod2 легко проверяется на практике: $0 \oplus 1 = 1 = \bar{0}$, $1 \oplus 1 = 0 = \bar{1}$), отсюда будет справедливо следующее:

$$\overline{t_{l,r}(i)} = t_{l,r}(i) \oplus 1111, \tag{4.4}$$

Учитывая соотношения, приведённые в формулах (4.1)-(4.4), получим:

$$\overline{t_{l,r}(i)} = t_{l,r}(i) \oplus 1111 = m_l(i) \oplus k_{l,r} \oplus 1111 = m_l(i) \oplus \overline{k_{l,r}} = \overline{m_l(i)} \oplus k_{l,r}, \tag{4.5}$$

Отсюда понятно, что для получения в результате сложения по mod2 сбалансированного байта необходимо, чтобы старший ниббл одного из аргументов операции сложения (сообщения $m_l(i)$ или ключа $k_{l,r}$) был бы инвертированным по отношению к младшему ниббл, тогда как старший ниббл второго аргумента должен повторять его младший ниббл. Если инвертированный ниббл будет содержаться в ключе, то при повторной итерации операции сложения получим, что оба аргумента будут содержать инвертированные старшие нибблы и результат окажется несбалансированным:

$$\left(\overline{t_{l,r}(i)} \,\|\, t_{l,r}(i)\right) \oplus \left(\overline{k_{l,r+1}} \,\|\, k_{l,r+1}\right) = t_{l,r+1}(i) \,\|\, t_{l,r+1}(i), \tag{4.6}$$

где $\|$ – конкатенация (например если $t_{l,\,r}(i)=0001$, тогда $\overline{t_{l,\,r}(i)}\,\|\,t_{l,\,r}(i)=11100001$); $k_{l,\,r+1}$ – подблок ключа, использующийся на следующей итерации; $t_{l,\,r+1}(i)$ – результат сложения на следующей итерации.

Поэтому, логично инвертированный ниббл использовать в сообщении: $m_l(i)\to\overline{m_l(i)}\,\|\,m_l(i)$, а повторный ниббл в ключе: $k_{l,\,r}\to k_{l,\,r}\,\|\,k_{l,\,r}$. Такой подход позволит использовать сколь-угодно много итераций операции сложения по mod2, при этом каждая из них окажется защищённой от DPA:

$$\left(\overline{t_{l,\,r}(i)}\,\|\,t_{l,\,r}(i)\right)\oplus\left(k_{l,\,r+1}\,\|\,k_{l,\,r+1}\right)=\overline{t_{l,\,r+1}(i)}\,\|\,t_{l,\,r+1}(i),\qquad(4.7)$$

Также, важно отметить, что для исключения возможности совершения атаки DPA с использованием модели расстояния Хэмминга результат сложения необходимо сохранять в чистую ячейку памяти (то есть содержащую нули или единицы) или в ячейку памяти, в которой находился байт ключа, в последнем случае расстояние Хэмминга между записываемыми и перезаписываемыми данными (для любого сообщения $m_l(i)$) будет постоянным – равным весу Хэмминга сбалансированного байта сообщения, то есть четырём:

$$\begin{aligned}&HD\left[\left(\overline{t_{l,\,r}(i)}\,\|\,t_{l,\,r}(i)\right),\ \left(k_{l,\,r}\,\|\,k_{l,\,r}\right)\right]=HW\left[\left(\overline{t_{l,\,r}(i)}\,\|\,t_{l,\,r}(i)\right)\oplus\left(k_{l,\,r}\,\|\,k_{l,\,r}\right)\right]=\\&=HW\left[\left(\overline{m_l(i)}\,\|\,m_l(i)\right)\oplus\left(k_{l,\,r}\,\|\,k_{l,\,r}\right)\oplus\left(k_{l,\,r}\,\|\,k_{l,\,r}\right)\right]=HW\left[\overline{m_l(i)}\,\|\,m_l(i)\right]=4\end{aligned},\qquad(4.8)$$

где $HD[a,\,b]$ – расстояние Хэмминга между комбинациями a и b; $HW[a]$ – вес Хэмминга комбинации a.

Если результат сложения по mod2 будет сохраняться в ячейку памяти, содержащую байт сообщения, то расстояние Хэмминга между записываемыми и перезаписываемыми данными для любого сообщения $m_l(i)$ также будет постоянным, но равным не четырём (оно будет зависеть от веса Хэмминга байта ключа, а он не является сбалансированным:

$$\begin{aligned}&HD\left[\left(\overline{t_{l,\,r}(i)}\,\|\,t_{l,\,r}(i)\right),\ \left(\overline{m_l(i)}\,\|\,m_l(i)\right)\right]=HW\left[\left(\overline{t_{l,\,r}(i)}\,\|\,t_{l,\,r}(i)\right)\oplus\left(\overline{m_l(i)}\,\|\,m_l(i)\right)\right]=\\&=HW\left[\left(\overline{m_l(i)}\,\|\,m_l(i)\right)\oplus\left(k_{l,\,r}\,\|\,k_{l,\,r}\right)\oplus\left(\overline{m_l(i)}\,\|\,m_l(i)\right)\right]=HW\left[k_{l,\,r}\,\|\,k_{l,\,r}\right]\end{aligned},\qquad(4.9)$$

Теоретически, данная особенность никак не поможет злоумышленнику в реализации атаки DPA, однако она потенциально может быть использована при реализации атаки SPA (описанной в разделе 1.6).

Систематизируя описанное выше, можно предложить следующий алгоритм реализации операции сложения по mod2, защищённой балансировкой:

1. Предварительно, блоки шифруемого сообщения $m(i)$ и раундовые ключи k_r, разбиваются на подблоки $m_l(i)$ и $k_{l,r}$ длины $n/2$, где n – разрядность чипа (далее будет рассматриваться, случай когда $n=8$).
2. Каждый подблок раундового ключа $k_{l,r}$ дополняется до подблока длины n путём его конкатенации с ним же (рисунок 4.4а):

$$k_{l,r} \to k_{l,r} \| k_{l,r}, \tag{4.10}$$

3. Каждый подблок сообщения $m_l(i)$ дополняется до подблока длины n путём его конкатенации с ним же, но в инвертированном виде (рисунок 4.4б):

$$m_l(i) \to \overline{m_l(i)} \| m_l(i), \tag{4.11}$$

4. Подблоки $\overline{m_l(i)} \| m_l(i)$ и $k_{l,r} \| k_{l,r}$ складываются друг с другом по mod2 обычным способом (стандартной командой ассемблера), при этом, результат сложения должен помещаться в чистую ячейку памяти или перезаписывать ячейку памяти, в которой содержалось значение $k_{l,r} \| k_{l,r}$ (рисунок 4.4в):

$$\left(\overline{m_l(i)} \| m_l(i)\right) \oplus \left(k_{l,r} \| k_{l,r}\right) = \left(\overline{m_l(i) \oplus k_{l,r}}\right) \| \left(m_l(i) \oplus k_{l,r}\right), \tag{4.12}$$

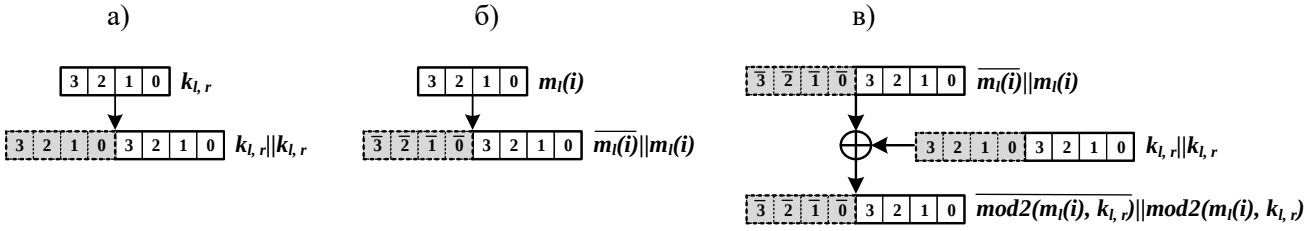


Рисунок 4.4 – Защищённая балансировкой реализация операции сложения по mod2: а) расширение подблока ключа; б) балансировка подблока сообщения; в) защищённая балансировкой операция сложения подблоков сообщения и ключа

Можно проверить, что при любых значениях $m_l(i)$ и $k_{l,r}$, вес Хэмминга результата сложения по (4.12) будет одинаковым (равняться $n/2$), то есть результат также будет состоять из двух нибблов – информационного (справа) и балансирующего (слева). Заметим, что операции

разделения сообщения на подблоки длиной $n/2$, и расширения каждого подблока до длины n, должны осуществляться для каждого обрабатываемого блока сообщения, тогда как разделение раундовых ключей на подблоки и их расширение может осуществляться единожды, при программировании чипа. Важно отметить, что байт ключа оказывается несбалансированным (это принципиально может быть использовано для реализации атаки SPA, хотя на практике, при достаточном уровне зашумления цепи электропитания чипа, успешная реализация такой атаки весьма сомнительна).

Защита S-box балансировкой основывается на тех же принципах, что и защита операции сложения по mod2 (рисунок 4.5). Это преобразование можно описать формулой:

$$\overline{p_{l,r}(i)} \,\|\, p_{l,r}(i) = S'\left(\overline{t_{l,r}(i)} \,\|\, t_{l,r}(i)\right), \tag{4.13}$$

где $\overline{t_{l,r}(i)} \,\|\, t_{l,r}(i)$ – сбалансированный байт входа защищённого S-box (далее S'-box); $S'(\cdot)$ – функция, описывающая преобразование на S'-box; $\overline{p_{l,r}(i)} \,\|\, p_{l,r}(i)$ – сбалансированный байт выхода S'-box, причём $p_{l,r}(i) = S\left(t_{l,r}(i)\right)$, где $S(\cdot)$ – преобразование на обычном S-box.

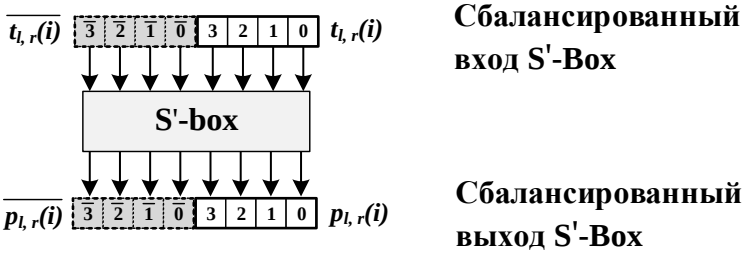


Рисунок 4.5 – Преобразование на защищённой балансировкой реализации S-box

Таблица преобразования S'-box описывается тривиально (таблица 4.2), где за основу берётся таблица преобразования обычного S-box, состоящая из двух столбцов: первый столбец – входные значения $s_{IN}(i)$, а второй столбец – выходные $s_{OUT}(i)$. Таблица преобразования S'-box получается конкатенацией соответствующих элементов таблицы обычного S-box со своими инвертированными значениями – элементы первого столбца будут рассчитываться как: $\overline{s_{IN}(i)} \,\|\, s_{IN}(i)$, элементы второго столбца: $\overline{s_{OUT}(i)} \,\|\, s_{OUT}(i)$.

Таблица 4.2 – Таблицы преобразования: а) обычного; б) защищённого S-box

а) б)

№	Вход S-box	Выход S-box		Вход S'-box	Выход S'-box
1	0000	0111		1111 0000	1000 0111
2	0001	0110		1110 0001	1001 0110
. . .					
16	1111	1000		0000 1111	0111 1000

Входы и выходы S'-box с описанной структурой очевидно окажутся сбалансированными при любых значениях $t_{l,r}(i)$. При реализации S'-box на чипе, результат преобразования необходимо записывать в пустую ячейку памяти, а не в ту ячейку, в которой хранился вход S'-box – в этом случае расстояние Хэмминга между записываемой и перезаписываемой комбинациями также окажется постоянным (равным $n/2$). Ассемблерная реализация сбалансированных S-box подробно описана в приложении К.

Необходимо подчеркнуть, что представленные реализации операции сложения по mod2 и преобразования на S'-box могут быть выполнены не только на специализированных чипах, но и на вполне бытовых микроконтроллерах – они требуют лишь незначительного увеличения объёма памяти программ (для хранения дополнительного кода программы, структуры S'-box и дополнительного ключа). В этом заключается качественное отличие балансировки от маскировки – для последней дополнительно потребуется генератор случайных чисел, смонтированный на кристалле чипа. Однако, в контексте защиты шифров ГОСТ, метод программной балансировки не имеет преимуществ над маскировкой – на сегодняшний день известны лишь три операции, поддающиеся защите программной балансировкой: описанные сложение по mod2 и табличное преобразование, а также циклический сдвиг (последний реализуем в защищённом виде лишь теоретически – известные автору чипы не смогут выполнить эту операцию в защищённом виде). Учитывая это, защита шифра «Магма» программной балансировкой окажется невозможной, ввиду того, что в нём используются операции сложения по $\mathrm{mod}2^{32}$ (при выполнении которой возникают переносы бит в соседние разряды, очевидно нарушающие балансировку) и циклического сдвига. Описание сложностей адаптации балансировки к шифру «Кузнечик» требует отдельного, детального разъяснения, которое будет приведено в следующем разделе.

4.3 Оценка возможности адаптации метода программной балансировкой для защиты шифра «Кузнечик»

Каждый из десяти раундов шифра «Кузнечик» можно условно разделить на три операции: сложение по mod2 сообщения с раундовым ключом, преобразование на S-box, L-преобразование (рисунки Л.1-Л.2 приложения Л). Описание защищённых реализаций первых двух операций было приведено в предыдущем разделе, отличия заключаются лишь в разрядности производимых преобразований – в «Кузнечике» используются восьмиразрядные S-box, следовательно, на вход защищённого S-box необходимо подавать 16-ти разрядную комбинацию, восемь младших бит которой будут информационными, а восемь старших – балансирующими, аналогичная ситуация наблюдается с L-преобразованием (которое будет подробно рассмотрено ниже). Поэтому возможность, защиты балансировкой, шифра «Кузнечик» (также, как и AES) можно рассматривать лишь на чипах с архитектурой на 16 разрядов или более.

Примем, что операции сложения по mod2 и преобразования на S-box защищены балансировкой, в этом случае на вход L-преобразования поступит 16 сдвоенных байт $a_l(i) = \overline{b_l(i)} \| b_l(i)$, $l = 0, \ldots, 15$ (младший байт – $b_l(i)$ является информационным, то есть представляет выход соответствующего S-box, а старший байт – $\overline{b_l(i)}$ – балансирующим, то есть инвертированным относительно информационного). Затем над каждым сдвоенным байтом производятся преобразования, описанные на рисунке Л.2 приложения Л, они заключаются в 13-ти операциях умножения в конечном поле с константами, сложений по mod2 и циклических сдвигах. Рассмотрим подробнее эти операции на примере преобразования двух младших сдвоенных байт (рисунок 4.6а). Реализацию умножения в конечном поле в защищённом виде проще всего осуществить при помощи таблицы замен, представляющей собой защищённый S'-box. Он будет отличаться от защищённого S'-box «Кузнечика», лишь реализуемыми преобразованиями – в S'-box L-преобразования они будут описывать таблицу умножения комбинации $b_l(i)$ на соответствующую константу (с дополнением результата умножения балансирующим байтом). В L-преобразовании понадобится семь различных «умножающих» S'-box (по одному на каждую константу): $S_{CONST}(\cdot)$, $CONST = 148, 32, 133, 16, 194, 192, 251$, к примеру выход $S_{148}(\cdot)$ будет следующим: $S_{148}(a_1(i)) = a_1(i) \cdot 148 = \overline{b_1(i) \cdot 148} \| b_1(i) \cdot 148$. Из рисунка 4.6б видно, что выход такого S'-box окажется сбалансированным, то есть защищённым от DPA. Затем, в соответствии с алгоритмом L-преобразования, комбинацию $S_{148}(a_1(i))$ необходимо сложить по mod2 с комбинацией $a_0(i) = \overline{b_0(i)} \| b_0(i)$, при этом требуется, чтобы

результатом этой операции стала сбалансированная комбинация $c_1(i) = \overline{\left(b_1(i)\cdot 148 \oplus b_0(i)\right)} \| \left(b_1(i)\cdot 148 \oplus b_0(i)\right)$.

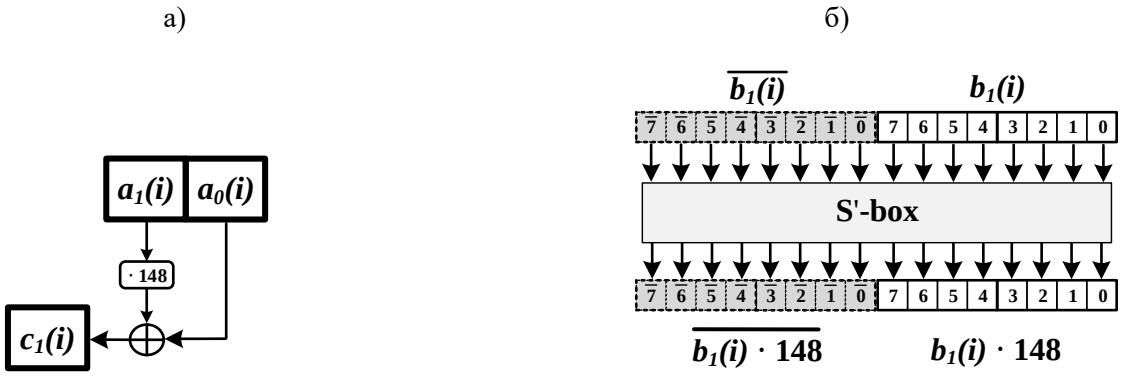


Рисунок 4.6 – а) Часть L-преобразования шифра «Кузнечик»; б) защищённая балансировкой реализация операции умножения в конечном поле

Заметим, что в соответствии с формулой (4.4) операция инвертирования двоичной комбинации x эквивалентна операции сложения по mod2 с единичной комбинацией: $\bar{x} = x \oplus 11...1$, отсюда будет справедливо следующее:

$$S_{148}\left(a_1(i)\right) = \overline{b_1(i)\cdot 148} \| b_1(i)\cdot 148 = \left(b_1(i)\cdot 148 \oplus 1111\,1111\right) \| \left(b_1(i)\cdot 148\right), \tag{4.14}$$

$$a_0(i) = \overline{b_0(i)} \| b_0(i) = \left(b_0(i) \oplus 1111\,1111\right) \| b_0(i), \tag{4.15}$$

$$\begin{aligned} c_1(i) &= S_{148}\left(a_1(i)\right) \oplus a_0(i) = \\ &= \left[\left(b_1(i)\cdot 148 \oplus \cancel{1111\,1111}\right) \oplus \left(b_0(i) \oplus \cancel{1111\,1111}\right)\right] \| \left[\left(b_1(i)\cdot 148\right) \oplus b_0(i)\right] = \\ &= \left(b_1(i)\cdot 148\right) \oplus b_0(i) \| \left(b_1(i)\cdot 148\right) \oplus b_0(i) \end{aligned}, \tag{4.16}$$

Из формул (4.14)-(4.16) видно, что результатом сложения окажется несбалансированная комбинация. Как ранее объяснялось (формулы (4.6)-(4.7)), результат операции сложения по mod2 $c_1(i)$ окажется сбалансированным, если один аргумент (сдвоенный байт) будет состоять из информационной и балансирующей частей, а другой аргумент из информационной части и повторения информационной части (то есть младший и старший байты должны дублировать друг друга). В противном случае старший байт результата сложения не будет инвертирован по отношению к младшему, а будет повторять его, в результате чего балансировка нарушится.

Таким образом, единственный способ обеспечить сбалансированность комбинации $c_1(i)$ заключается в устранении балансировки одного из аргументов. Комбинация $a_0(i)$ является выходом 16-го S-box – её несбалансированность позволит классическим образом реализовать атаку DPA с использованием выхода S-box, а следовательно восстановить восемь бит младшего разряда раундового ключа. Операция умножения в конечном поле является однозначной, поэтому несбалансированность результата умножения $a_1(i)$ на 148 также весьма просто позволит восстановить 8-15 биты раундового ключа «Кузнечика». Поэтому нарушение балансировки любого аргумента операции сложения по mod2 приведёт к раскрытию части раундового ключа «Кузнечика».

Пренебрежём этим, и примем, что комбинация c_1 окажется несбалансированной, тогда на следующем шаге она складывается по mod2 со сбалансированной комбинацией $a_2(i)\cdot 32$ (рисунок Л.2 приложения Л):

$$
\begin{aligned}
&c_2(i) = c_1(i) \oplus \left(a_2(i)\cdot 32\right) = \\
&= \left[\left(b_1(i)\cdot 148\right) \oplus b_0(i) \,\|\, \left(b_1(i)\cdot 148\right) \oplus b_0(i)\right] \oplus \left[\overline{b_2(i)\cdot 32} \,\|\, b_2(i)\cdot 32\right] = \\
&= \left[\left(\left(b_1(i)\cdot 148\right) \oplus b_0(i)\right) \oplus \left(\overline{b_2(i)\cdot 32}\right)\right] \| \left[\left(\left(b_1(i)\cdot 148\right) \oplus b_0(i)\right) \oplus \left(b_2(i)\cdot 32\right)\right] = \\
&= \left[\left(\left(b_1(i)\cdot 148\right) \oplus b_0(i)\right) \oplus \left(b_2(i)\cdot 32\right) \oplus 1111\,1111\right] \| \left[\left(\left(b_1(i)\cdot 148\right) \oplus b_0(i)\right) \oplus \left(b_2(i)\cdot 32\right)\right] = \\
&= \left[\overline{\left(\left(b_1(i)\cdot 148\right) \oplus b_0(i)\right) \oplus \left(b_2(i)\cdot 32\right)}\right] \| \left[\left(\left(b_1(i)\cdot 148\right) \oplus b_0(i)\right) \oplus \left(b_2(i)\cdot 32\right)\right]
\end{aligned}
, \quad (4.17)
$$

Результат $c_2(i)$ в (4.17) очевидно окажется сбалансированным, отсюда понятно, что результаты сложения по mod2 в L-преобразовании поочерёдно будут принимать сбалансированное и несбалансированное значения вплоть до последней операции сложения по mod2 с комбинацией $a_{15}(i)\cdot 148$. Атака на несбалансированные значения $c_1(i)$, $c_3(i)$, …, $c_{15}(i)$ окажется нетривиальной но вполне возможной, поэтому в целом подобную реализацию шифра «Кузнечик» нельзя считать защищённой от DPA, отсюда можно сделать вывод, что на практике, к российским стандартам шифрования, нельзя напрямую адаптировать ни маскировку, ни балансировку.

Из приведённого в разделах 4.2-4.3 описания двух программных методов защиты, становится очевидным значительное ограничение, накладываемое на разработчиков методов защиты от DPA – они должны находить способы защиты операций, составляющих алгоритм шифра, с возможностью изменения лишь подходов к реализации этих операций, но не их самих. При этом постулируется, что получаемые на защищённом чипе криптограммы должны в

точности соответствовать получаемым на незащищённом чипе. Такой подход вполне понятен, так как никакие изменения алгоритма стандартного шифра очевидно недопустимы. Однако, как было ранее показано, это жёстко ограничивает возможности разработчика по защите шифра лишь набором используемых в нём преобразований, которые (как например, операции, составляющие шифры ГОСТ) или вовсе не поддаются защите или требуют значительных вычислительных ресурсов защищаемого чипа. Отсюда возникает идея разработки нового подхода к защите, который с одной стороны не изменял бы структуру защищаемого шифра, а с другой – дал бы исследователям б**о**льшую свободу при разработке конкретных методов защиты. При этом, к разрабатываемому методу защиты можно сформировать следующие базовые требования:

- Дешевизна реализации (желательно возможность реализации на существующих чипах без необходимости аппаратных доработок или с незначительными доработками).
- Минимальное снижение скорости выполнения защищаемого алгоритма.
- Неизменность защищаемого алгоритма шифра (одно и то же сообщение, на защищённом и незащищённом чипах должно преобразовываться в одну и ту же криптограмму). В крайнем случае возможно допустить изменение алгоритма шифра, но с гарантией сохранения устойчивости к любым методам криптоанализа.
- Высокая стойкость к атаке DPA, доказанная теоретически.

Разработке метода защиты, обладающего приведёнными характеристиками посвящён следующий раздел.

4.4 Обобщённая структура универсального метода защиты от DPA

Основная цель любого программного метода защиты от DPA заключается в устранении зависимости между обрабатываемыми чипом данными и его энергопотреблением. При этом, до сегодняшнего дня, они предполагали модификацию не самого алгоритма шифра, а лишь его программной реализации (то есть криптограммы, получаемые на фиксированном сообщении и секретном ключе при использовании защищённого и незащищённого шифров, оказываются идентичными). Этот подход абсолютно логичен и обоснован – если будет предложен метод защиты от DPA, требующий изменения (пусть даже весьма незначительные), самого алгоритма шифра, например, таблиц подстановок (S-box), или структуры перестановок бит в конце раунда, или любой иной операции, заданной соответствующим стандартом, тогда, даже несмотря на возможную незначительность введённых изменений, получится совершенно другой шифр. В

отношении «нового» шифра, окажутся неприменимыми все исследования стойкости стандартного шифра к множеству существующих атак, проведённые колоссальным количеством специалистов. Кроме того, на модифицированный шифр не будет распространяться сертификация соответствующих государственных структур, а что наиболее важно – к нему не будет доверия у пользователей. По этим причинам разработчики методов защиты от DPA ограничены лишь поиском способов защиты операций, составляющих шифр. Это в конце концов приводит к снижению защищённости чипа, значительным требованиям к его вычислительным ресурсам (что является причиной увеличения стоимости чипа) или значительному снижению скорости шифрования. Поэтому возникает идея разработки нового подхода к защите, который, с одной стороны, позволил бы сохранить сертификацию шифра, а с другой стороны, не зависел бы от составляющих его операций. В качестве такой идеи можно предложить весьма простой, универсальный и потенциально надёжный поход, предполагающий использование дополнительных преобразований над шифруемым сообщением (предобработки) и полученной криптограммой (постобработки) (рисунок 4.7).

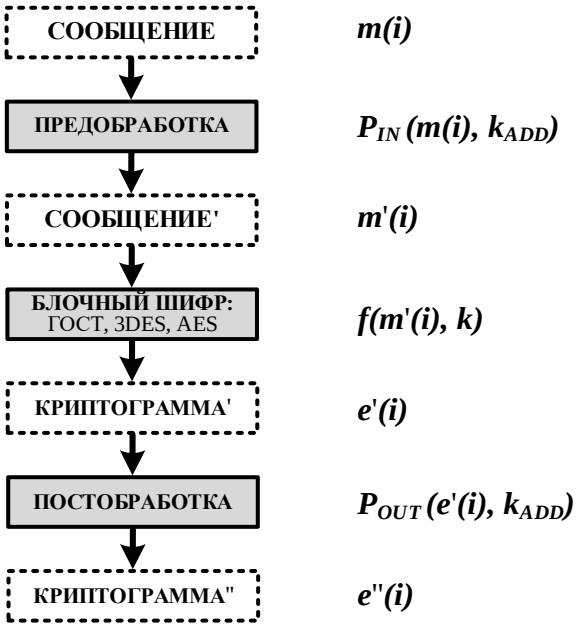


Рисунок 4.7 – Обобщённая структура универсального метода защиты от DPA

Идея такого подхода заключается в том, чтобы для i-го шифруемого сообщения $m(i)$, сначала произвести некоторую предварительную обработку, реализация которой оказывается защищённой от DPA: $m'(i) = P_{IN}\left(m(i),\, k_{ADD}\right)$, где $P_{IN}(\cdot)$ – функция, описывающая преобразования в узле предобработки; k_{ADD} – дополнительный ключ, использующийся на узле предобработки. Далее так обработанное сообщение преобразуется алгоритмом шифрования

(реализованным обычным способом): $e'(i) = f\left(m'(i),\ k\right)$, где $f(\cdot)$ – функция, описывающая преобразования на любом блочном шифре; k – секретный ключ шифра; $e'(i)$ – выход шифра (криптограмма) при входе $m'(i)$. Затем полученная криптограмма $e'(i)$ подвергается постобработке: $e''(i) = P_{OUT}\left(e'(i),\ k_{ADD}\right)$. (По аналогии с предобработкой, необходимость её использования обусловлена возможностью реализации атаки DPA, как от начала шифрования, с использованием известных шифруемых сообщений, так и от конца шифрования, с использованием известных криптограмм), причём результат постобработки и будет являться выходом защищённого шифра.

В простейшем случае алгоритмы предобработки и постобработки выбираются идентичными (далее будет рассматриваться только этот случай): $P_{IN}(\cdot) \sim P_{OUT}(\cdot)$, эти алгоритмы должны удовлетворять следующим основным требованиям:

1. Защищённость преобразований $P_{IN}(\cdot)$, $P_{OUT}(\cdot)$ от атак DPA.
2. Зависимость $m'(i)$ и $e'(i)$ от такого количества бит ключа k_{ADD}, которое не позволит рассчитать их только по известным $m(i)$ и $e''(i)$ за реализуемое время.
3. Обратимость и однозначность преобразований (фиксированная комбинация на входе, должна преобразовываться в одну и ту же комбинацию на выходе; различные комбинации на входе, должны преобразовываться в различные комбинации на выходе). Причём, сложность обратного преобразования должна быть сопоставима с прямым.

Является очевидным, что при выполнении этих условий злоумышленник окажется не в состоянии за реализуемое время найти биты входа $m'(i)$ или выхода $e'(i)$ защищаемого шифра, а, следовательно, он не сможет произвести расчёт результата выполнения каких-либо промежуточных операций шифра, путём перебора небольших частей его секретного ключа. Поэтому атака DPA окажется невозможной.

Очевидной особенностью такого метода защиты будет отличие криптограммы, полученной на защищённом шифре, от криптограммы, полученной, для незащищённой реализации шифра. Однако, поскольку УМЗ не предполагает внесения каких-либо изменений в алгоритм защищаемого шифра, то, в случае использования в узлах предобработки и постобработки дополнительного ключа (не зависящего от секретного ключа шифра), стойкость получаемой модификации шифра к любым методам криптоанализа оказывается принципиально *не ниже,* чем у оригинального (незащищённого) шифра. Более того, дополнительные преобразования могут даже увеличить стойкость шифра. Использование УМЗ даёт значительно больший простор для

разработчиков методов защиты – ведь теперь они не ограничены операциями, входящими в алгоритм защищаемого шифра. Поэтому в узлах предобработки и постобработки можно использовать преобразования, наилучшим образом поддающиеся защите от DPA. Строго говоря узлы предобработки и постобработки можно представить в виде отдельных (не зависящих от защищаемого шифра) блочных шифров, структура которых предполагает их защищённость от атаки DPA. Между ними может помещаться любой блочный шифр – очевидно, что единственный его параметр, который имеет значение при реализации УМЗ это длина блоков сообщения и криптограммы. Кроме того, при условии разработки оптимального метода защиты, описанный подход, может быть стандартизован и использоваться как модификация блочного шифра, с возможностью, в случае необходимости, получения сертификации от соответствующих государственных структур.

Предлагаемый подход не является строго определённым методом защиты от DPA. В широком смысле, УМЗ это целое семейство методов защиты, причём подразумевается, что разработчики могут предлагать различные структуры УМЗ, состоящие из оптимального сочетания защищённых от DPA преобразований, наилучшим образом подходящих для реализации на конкретном чипе. В следующем разделе, для демонстрации работоспособности этой идеи, будет предложена структура УМЗ, в основе которой лежат операции, защищённые, описанным ранее, методом программной балансировки.

4.5 Описание деталей предлагаемой структуры УМЗ

Приведённая выше структура УМЗ не конкретизирует используемые в нём защищённые преобразования. Из известных методов защиты от DPA, строго доказанную защищённость имеют, описанные ранее, метод маскировки и программной балансировки. Использование в узлах предобработки и постобработки, операций, защищённых маскировкой возможно, причём можно выбирать совокупность операций, наиболее просто поддающихся защите маскировкой. Однако, маскировка требует наличия в чипе высококачественного генератора случайных чисел, то есть предъявляются дополнительные требования к его аппаратной составляющей. В отличие от этого, операции, защищённые балансировкой, можно реализовать, практически, на любом бытовом микроконтроллере. Эта особенность приводит к большей универсальности балансировки, реализацию которой можно считать экономически более оправданной, так как появляется возможность обеспечения надёжной защиты от DPA на более дешёвых чипах.

Ниже будет рассмотрено, каким образом в структуре узлов предобработки/постобработки можно использовать защищённые балансировкой реализации операций сложения по mod2 и преобразования на S-box.

На предварительном этапе, поступающее сообщение необходимо разбить на подблоки по четыре бита, затем каждый подблок дополнить до байта балансирующими битами (инвертированными по отношению к информационным; рисунок М.1 приложения М). После этого, каждый сбалансированный байт, обрабатывается по определённому алгоритму.

Интуитивно, простейшей реализацией узла предобработки является многократное выполнение защищённой от DPA операции сложения по mod2 каждого подблока сообщения с подблоками дополнительного ключа (рисунок 4.8).

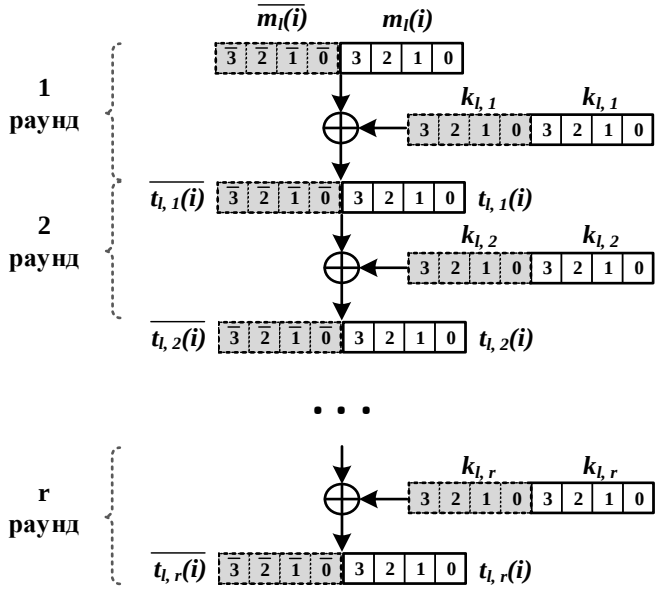


Рисунок 4.8 – Простейшая реализация узла предобработки

В этом случае, выходом узла предобработки будет являться комбинация, зависящая от $4r$ бит дополнительного ключа. Вместе с этим, является очевидным, что многократное сложение по mod2 может быть легко аппроксимировано одним сложением:

$$m_l(i) \oplus k_{l,1} \oplus k_{l,2} \oplus \ldots \oplus k_{l,r} = m_l(i) \oplus k_{ALL}, \quad (4.18)$$

где $k_{ALL} = k_{l,1} \oplus k_{l,2} \oplus \ldots \oplus k_{l,r}$

Поэтому для нахождения четырёх младших бит выхода узла предобработки, злоумышленнику потребуется перебрать лишь 16 возможных вариантов ключа k_{ALL}. Отсюда, при реализации DPA к примеру, на шифр «Магма», для восстановления младшего четырёх битового подблока секретного ключа шифра, злоумышленнику потребуется перебрать не 16, а 256

вариантов ключа (то есть к четырём битам младшего подблока раундового ключа первого раунда шифра, добавится четыре бита дополнительного ключа узла предобработки) что конечно не представляет значимой вычислительной сложности.

Поэтому можно предложить над каждым сбалансированным подблоком сообщения последовательно применять операцию сложения по mod2 и преобразования на S'-box (рисунок 4.9). (Заметим, что таблицы преобразования S'-box могут быть выбраны отличными для разных раундов.) При этом таблица преобразования последнего S'-box реализуется таким образом, чтобы балансирующая часть байта оказывалась равной нулю (то есть последний S'-box оказывается незащищённым). Это, как будет показано впоследствии, позволит достаточно просто «склеить» информационные части полученных байт, которые в дальнейшем поступают на вход шифра.

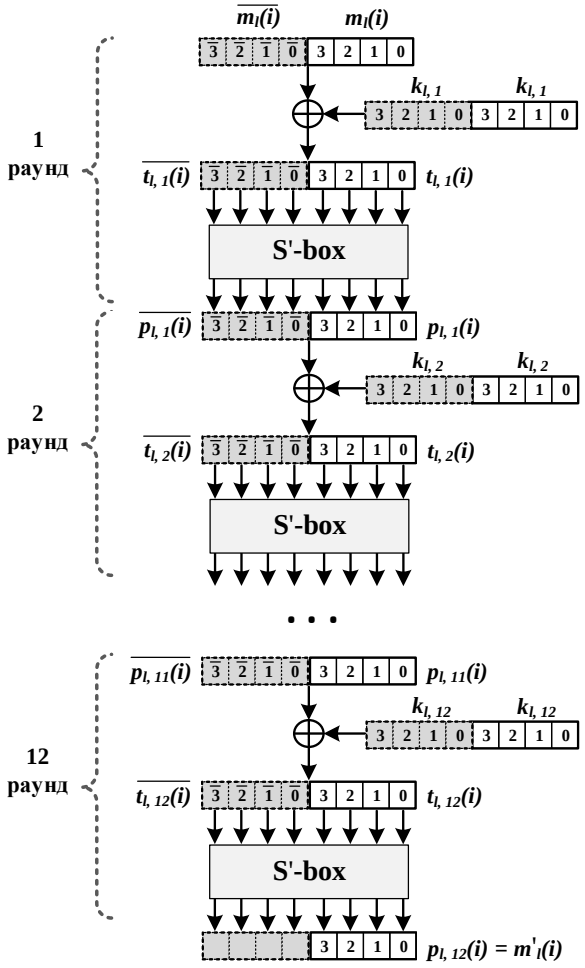


Рисунок 4.9 – Реализация защищённого от DPA узла предобработки (на примере обработки четырёх бит сообщения)

Предлагаемый подход уже не позволит произвести аппроксимацию комплекса преобразований узла предобработки неким простым преобразованием. Атаку DPA на выход последнего раунда узла предобработки будет возможно реализовать по той причине, что злоумышленнику будут известны сообщения, поступающие на узел предобработки, а выход узла

предобработки оказывается несбалансированным (то есть не защищённым от DPA). Однако при реализации атаки DPA злоумышленнику придётся совершить перебор дополнительного ключа. В приведённом на рисунке 4.9 примере узел предобработки состоит из 12-ти раундов, на каждом из которых расходуется по четыре бита дополнительного ключа. Общая длина дополнительного ключа, в совокупности по 12-ти раундам, составляет 48 бит. Поэтому, атака DPA будет предполагать совершение перебора 2^{48} (то есть больше 280 триллионов) вариантов дополнительного ключа, на каждом из которых, необходимо будет рассчитать корреляционный вектор (с использованием некоторого массива снятых форм сигнала), а затем произвести сравнение каждого из 2^{48} векторов, и определить тот на котором происходит наибольший всплеск значений. Подобную задачу, можно считать весьма сложной даже в теории. На практике, при качественной программной реализации алгоритма DPA даже на весьма мощном ПК (производительность процессора которого может быть грубо оценена в 200-300 гигафлопс), вряд ли удастся совершать перебор более 100 вариантов ключа в секунду (при использовании 1 000 форм сигнала, каждая из которых состоит из 500 отсчётов – что является очень хорошими условиями). Поэтому за год будет возможно перебрать разве лишь чуть больше 3 млрд. комбинаций – для перебора 2^{48} вариантов ключа понадобится порядка ста тысяч лет.

Необходимо заметить, что предложенный подход имеет одну весьма очевидную уязвимость – в представленной реализации отсутствуют блоки перестановки бит (структура узла предобработки не является разветвлённой). Следовательно четыре бита выхода узла предобработки $m'_l(i)$ однозначно зависят от четырёх бит сообщения $m_l(i)$. Поэтому при обработке одинаковых подблоков сообщения $m_l(i)$, на выходе узла предобработки будут появляться одинаковые подблоки $m'_l(i)$. Тогда можно представить 12 итераций узла предобработки подблока сообщения, как «чёрный ящик», и составить таблицу, в которой каждому значению подблока $m_l(i)$ сопоставляется значение $m'_l(i)$. Такая таблица будет иметь два столбца и 16 строк (таблица 4.3).

Таблица 4.3 – Пример таблицы сочетаний комбинаций $m_l(i)$ и $m'_l(i)$

$m_l(i)$	$m'_l(i)$
0000	1000
0001	1001
0010	1010
0011	1011
0100	1100
0101	1101
0110	1110
0111	1111
1000	0000
1001	0001
1010	0010
1011	0011
1100	0100
1101	0101
1110	0110
1111	0111

Количество возможных вариантов таких таблиц будет равно количеству возможных перестановок элементов столбца $m'_l(i)$, то есть 16! = 20 922 789 888 000 (более 20 триллионов вариантов) это число находится между 2^{44} и 2^{45}. Отсюда следует, что реализация более чем 12-ти раундов преобразований в УМЗ окажется излишней – злоумышленнику будет проще реализовать DPA, перебирая все возможные варианты таблиц, чем дополнительный ключ. Перебор такого количества таблиц займёт значительно меньшее время, чем перебор ключа, но оно всё равно окажется нереализуемо большим – порядка 7 000 лет. Однако, если принять во внимание производительность наиболее мощного на 2016 год суперкомпьютера Sunway TaihuLight около 100 петафлопс [22], то при помощи него, перебор всех вариантов таких таблиц можно совершить за неделю (стоимость подобной машины исчисляется сотнями миллионов долларов США, а занимаемая площадь имеет порядок сотен м^2). Для сравнения, перебор всех вариантов таблиц на самом высокопроизводительном (2.57 петафлопс) отечественном суперкомпьютере Ломоносов-2 [17], займёт больше полугода. Необходимо подчеркнуть, что представленные оценки сложности реализации DPA на предлагаемую реализацию узла предобработки, являются весьма грубыми, но даже приведённой поверхностной оценки, достаточно для формулирования вывода, что описанный метод защиты окажется уязвимым к атакам, реализуемым государственными службами, а также крупными частными компаниями (в

разделе 4.6 будут предложены способы усиления защищённости предлагаемой структуры УМЗ от атак реализуемых с использованием чрезвычайно мощных вычислительных ресурсов).

Заметим, что представленная на рисунке 4.9 модель узла предобработки позволит преобразовать лишь четыре бита сообщения. Поэтому при защите шифра «Магма» (длина блоков сообщения/криптограммы которого составляет 64 бита) сообщение будет необходимо разбить на 16 четырёх битовых подблоков (то есть $l = 1, ..., 16$), каждый из которых необходимо сбалансировать, затем над каждым из них произвести, описанные рисунком 4.9, преобразования. Для шифра «Кузнечик», блок шифруемого сообщения имеет длину 128 бит – его предобработка займёт в два раза больше времени, так как в данном случае необходимо произвести предобработку уже 32-х четырёх битовых подблоков.

Также необходимо обратить внимание на то, что УМЗ не предполагает модификации реализации алгоритма защищаемого шифра, таким образом, комбинация с выхода узла предобработки, должна быть избавлена от балансирующих бит – состоять лишь из информационных, которые впоследствии поступают на обычную реализацию шифра (то же самое относится и к узлу постобработки, выход которого будет представлять собой выход защищённого шифра). В зависимости от особенностей конкретной модели защищаемого чипа могут быть предложены разные способы исключения балансирующих бит и «склеивания» четырёх битовых подблоков информационных бит. Наиболее универсальный подход предполагает избавление от балансирующих нибблов, при помощи операции логического «И» с маской (комбинацией «0000 1111» или «1111 0000»), и «склеивание» при помощи операции сложения по mod2 (рисунок 4.10).

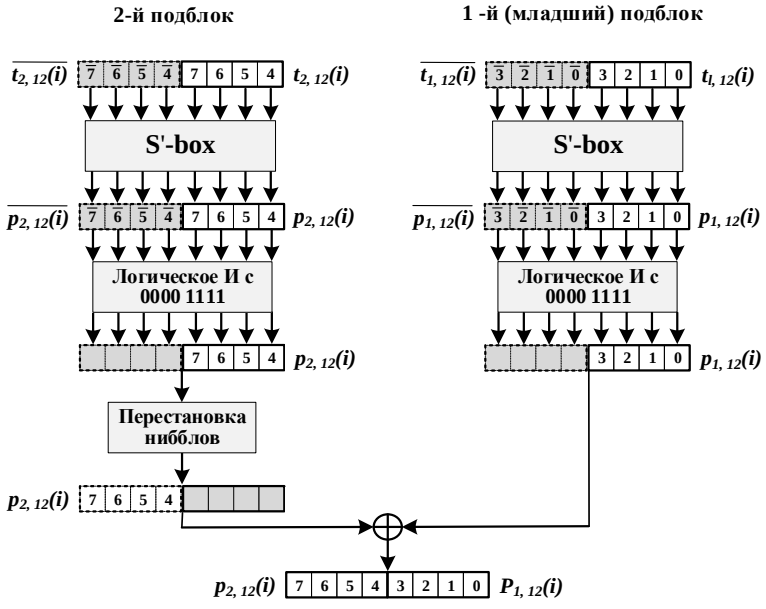


Рисунок 4.10 – Избавление от балансирующих нибблов с использованием операций логического «И» и перестановки нибблов

Описанным выше методом «склеиваются» все 16 (для «Магмы») или 32 (для «Кузнечика») подблоков, после чего они подаются на вход защищаемого шифра.

Необходимо обратить внимание на важную особенность предлагаемой структуры узлов предобработки и постобработки – операции, составляющие их преобразования (сложение по mod2 и преобразование на S'-box) являются обратимыми, следовательно, преобразования предобработки и постобработки также окажутся обратимыми. Рассмотрим этот вопрос подробнее на примере узла постобработки шифра «Магма». Вначале опишем реализацию алгоритма постобработки при шифровании сообщения: с выхода шифра, на вход узла постобработки подаётся комбинация $e'(i)$ длиной 64 бита, далее она разбивается на четырёх битовые подблоки: $e'(i) = e'_{16}(i) \| ... \| e'_1(i) \| e'_0(i)$. Затем производится балансировка каждого подблока: $e'_l(i) \to \overline{e'_l(i)} \| e'_l(i)$, тем самым их длина увеличивается до байта. Затем над каждым из 16-ти подблоков реализуется операция постобработки (идентичная предобработке; рисунок 4.9), после чего, полученные четырёх битовые подблоки $e''_i(i)$ «склеиваются» в 64-ёх битовую выходную комбинацию: $e''(i) = e''_{16}(i) \| ... \| e''_1(i) \| e''_0(i)$.

При расшифровывании сообщения, выходная комбинация $e''(i)$ также разбивается на четырёх битовые подблоки $e''(i) = e''_{16}(i) \| ... \| e''_1(i) \| e''_0(i)$, после чего каждый подблок балансируется. Затем, над каждым подблоком совершаются преобразования, описанные рисунком 4.9, но в обратном порядке (рисунок 4.11). При этом, является очевидным, что результатом такого преобразования окажется четырёх битовый подблок $e'_l(i)$ идентичный тому, который получался при шифровании. Далее подблоки «склеиваются» в блок $e'(i)$ – он расшифровывается в блок $m'(i)$ обычной реализацией шифра. Над блоком $m'(i)$ совершается преобразование, обратное предобработке (по полной аналогии со схемой на рисунке 4.11), таким образом получается искомое расшифрованное сообщение $m(i)$.

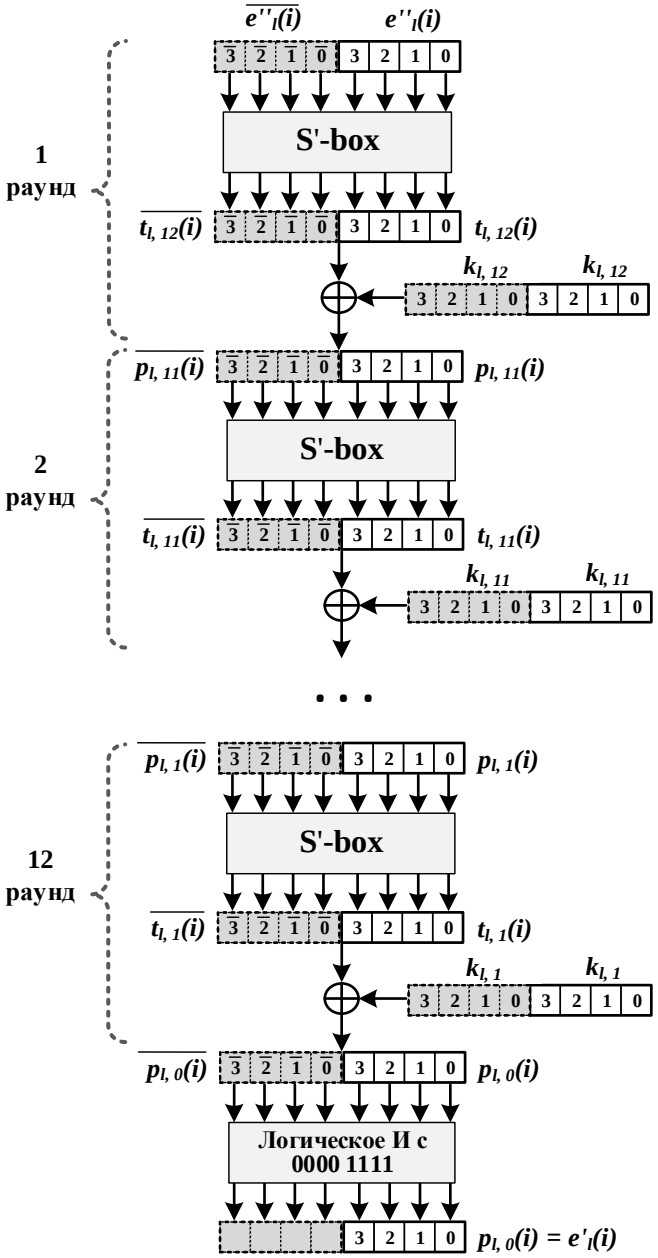


Рисунок 4.11 – Реализация, защищённого от DPA, узла постобработки при расшифровывании сообщения

По той причине, что все, описанные рисунком 4.11, операции совершаются над сбалансированными байтами – атака DPA с использованием любых промежуточных значений (кроме последнего раунда) окажется невозможной. В том случае, если злоумышленнику известны подблоки $e_l''(i)$ криптограммы, то ближайшие к ним несбалансированные промежуточные значения шифра, представляют собой вход узла постобработки (подблоки $e_l'(i)$, на рисунке 4.11), при реализации атаки можно использовать перебор дополнительного ключа или таблиц – сложность окажется такой же (перебор 2^{48} вариантов ключа или 16!

вариантов таблиц), как и при атаке на узел предобработки (при шифровании сообщений). Такая же сложность будет и при использовании подблоков расшифрованного сообщения.

Оценка дополнительных ресурсов чипа, требующихся для реализации предлагаемого метода защиты приведена в приложении К. В произведённой автором реализации УМЗ на микроконтроллере PIC, для защиты шифра «Кузнечик», скорость шифрования снизилась не более чем на 50%, по сравнению с незащищённой реализацией «Кузнечика», причём потребовался весьма небольшой объём памяти для хранения дополнительного кода программы.

4.6 Характеристики предлагаемой структуры УМЗ

При рассмотрении предлагаемой структуры узлов предобработки и постобработки, чрезвычайно важным является вопрос распределения дополнительного ключа по каждому их раунду. В рамках преобразования каждого четырёх битового подблока сообщения, раундовые ключи на каждом из 12-ти раундов предобработки очевидно должны отличаться – это сделает бессмысленным их перебор (по сравнению с перебором вариантов таблиц). Однако возникает вопрос о том следует ли использовать разные дополнительные ключи для предобработки разных подблоков сообщения. Как говорилось ранее, длина блока шифруемого сообщения шифра «Магма» составляет 64 бита, следовательно, необходимо предобработать 16 подблоков по четыре бита каждый, использование разных дополнительных ключей при предобработке каждого подблока приведёт к необходимости хранения в чипе 768-ми бит (96 байт) ключа – по меркам современных чипов это небольшой объём памяти. Однако встаёт вопрос об эффективности применения разных ключей, по сравнению с одним 48-ми битовым (шесть байт) ключом. При использовании одинаковых дополнительных ключей для предобработки всех подблоков сообщения, злоумышленнику будет достаточно восстановить 48 бит дополнительного ключа, при помощи которого преобразуется один четырёх битовый подблок сообщения (или подобрать соответствующую таблицу сочетаний из 16! возможных), для того, чтобы узнать результат предобработки всех других подблоков (то есть полностью определить комбинацию, подающуюся на вход шифра). В том случае если ключи будут выбраны разными, то злоумышленнику придётся находить ключ (или подбирать таблицу) для каждого подблока. К примеру, для того, чтобы узнать 64-х битовую комбинацию, поступающую на вход шифра «Магма» будет необходимо перебрать ключи каждого из 16-ти подблоков, что потребует в 16 раз больше времени, то есть совершить перебор $16 \cdot 2^{48} = 2^4 \cdot 2^{48} = 2^{52}$ вариантов ключа, или $16 \cdot 16! = 334\,764\,638\,208\,000$ вариантов таблиц (это число находится между 2^{48} и 2^{49}). Длина

блока сообщения шифра «Кузнечик» в два раза больше «Магмы» и поэтому для взлома ключа узла предобработки шифра «Кузнечик», понадобится в два раза больше времени, по сравнению с «Магмой». Приведённые расчёты показывают, что использование различных ключей при преобразовании разных подблоков не приводит к принципиальному увеличению защищённости. Атака, реализуемая частными лицами и криминальными группами, окажется невозможной и при использовании 48-ми бит дополнительного ключа, а усложнение атаки в 16 раз, вводимое использованием 768-ми бит дополнительного ключа, всё равно не позволит обеспечить принципиальную защищённость на сколько-нибудь длительную перспективу от государственных структур и крупных частных компаний. Поэтому использование 768-ми бит дополнительного ключа выглядит разумным лишь для защиты от перспективных (потенциальных) атак на предлагаемую реализацию УМЗ (с использованием ограниченных ресурсов), и только в том случае, если требование 90 дополнительных байт в ПЗУ не приведёт к значительному удорожанию защищаемого чипа. Использование разных ключей в узлах предобработки и постобработки также не внесёт принципиального усложнения в атаку, в этом случае атака разделяется на три этапа:

- Злоумышленник реализует атаку на дополнительный ключ узла предобработки (или постобработки), и восстанавливает его.
- Злоумышленник реализует атаку DPA на незащищённый шифр и восстанавливает секретный ключ шифра.
- Неизвестный ключ постобработки (предобработки) восстанавливается за несколько минут обычным перебором.

В том случае, если УМЗ применяется вместе с другими методами защиты от DPA, можно признать опасным использование секретного ключа шифра в качестве дополнительного ключа УМЗ. Таким образом взлом УМЗ не приведёт к взлому секретного ключа шифра. Вместе с этим, можно признать вполне допустимым вычисление дополнительного ключа УМЗ как результата применения криптостойкой бесключевой хэш-функции к секретному ключу защищаемого шифра – в этом случае взлом ключа УМЗ не приведёт к компрометации ключа шифра.

Помимо дополнительного ключа, важным вопросом является структура таблиц преобразований S'-box, использующихся в узлах предобработки и постобработки, но вначале необходимо пояснить некоторые принципиальные особенности последних. Предлагаемую структуру узла предобработки можно рассматривать как симметричную, блочную криптосистему (с интегрированной защитой от побочных атак по цепям электропитания). Однако, в отличие от традиционных шифров, в отношении узла предобработки злоумышленнику оказывается известна входная комбинация (рисунок 4.7), но не известна выходная (для узла

постобработки – наоборот: известна выходная, но неизвестна входная). Вместе с этим, реализация криптографических атак на шифры требует знания определённого количества пар сообщение/криптограмма (применительно к атакам на УМЗ корректно говорить о парах входной/выходной последовательностей узла предобработки или постобработки). Отсюда становится понятно, что на узлы предобработки и постобработки невозможна практическая реализация известных атак, например, таких как: линейный, дифференциальный криптоанализ, атаки по методу максимального правдоподобия, атаки по методу решения систем нелинейных уравнений и подобных. Из известных атак, оказывается возможной лишь реализация атак по сторонним каналам, в частности DPA, где в качестве рассчитываемого промежуточного значения будет использоваться выход узла предобработки или вход узла постобработки. При реализации DPA необходимо совершать перебор всех вариантов ключа или перебор таблиц. Усиление защиты от перебора таблиц будет подробно рассмотрено позже, а для защиты от перебора дополнительного ключа очевидно необходим ключ достаточной длины (как говорилось выше, реализация 12-ти раундов позволяет достичь максимальной полезной длины ключа), и невозможность аппроксимации ключа неким меньшим ключом как это показано на рисунке 4.8 и формулой (4.18). Для защиты от последнего, в структуру узлов предобработки/постобработки и вводятся нелинейные преобразования в виде S'-box.

В структуре узлов предобработки и постобработки можно предложить использовать одинаковые S'-box, таблица преобразования которых, описывается следующим соотношением:

$$S\left(x\right)=x^{-1}\Rightarrow S'\left(\overline{x}\,\|\,x\right)=\overline{x^{-1}}\,\|\,x^{-1},\tag{4.19}$$

где x – элемент конечного поля $GF\left(2^4\right)$; x^{-1} – обратный элемент к элементу x.

Данная структура преобразований уже не позволит тривиально аппроксимировать ключ. Более того, упрощается процесс расшифровывания сообщения – в этом случае используется та же таблица преобразований, что и для шифрования. Кроме этого, из теории криптографии известно, что преобразования на таких S-box, имеют наименьшую вероятность аппроксимаций линейными равенствами [4, с. 63-64]. Поэтому узлы предобработки и постобработки окажутся менее восприимчивыми к линейному и дифференциальному криптоанализу – данное свойство сыграет роль дополнительной защиты от перспективных атак, которые могут появиться в отношении предлагаемого метода защиты. Необходимо заметить, что принципиально, данный метод защиты позволяет использовать разные S'-box на каждом раунде узлов предобработки и постобработки, более того таблицы преобразования S'-box могут выступать в качестве дополнительного секрета (вместе с ключом), однако практический выигрыш от такого шага весьма сомнителен.

Как говорилось ранее, наиболее слабым местом предлагаемой структуры УМЗ является, то, что при использовании чрезвычайно мощных вычислительных ресурсов, остаётся возможность реализации в отношении неё атаки DPA. Длину дополнительного ключа, использующегося при предобработке подблока сообщения, можно увеличить за счёт добавления дополнительных раундов, например, использование вместо 12-ти раундов 20-ти позволит увеличить длину ключа с 48-ми бит до 80-ти. Однако данный шаг является абсолютно бессмысленным, потому, что вместо перебора ключа, злоумышленник имеет возможность перебирать таблицы, количество вариантов, которых строго определяется разрядностью информационной части обрабатываемых чипом комбинаций, а в приведённом примере она оказывается равной четырём. Можно предложить два пути модификации узла предобработки, позволяющих значительно усложнить реализацию перебора таблиц: разветвление структуры узлов предобработки/постобработки и увеличение разрядности защищаемых чипов.

Идея подхода с разветвлением заключается в том, чтобы на каждом из 12-ти раундов узлов предобработки/постобработки, помимо операций сложения по mod2 и преобразования на S-box, ввести операцию обмена битами между подблоками (рисунок 4.12). Обмен может быть реализован в виде перестановки бит между заданными позициями или, к примеру, циклическим сдвигом комбинации, полученной с выходов всех S-box на количество бит некратное восьми. Такой шаг может позволить обеспечить зависимость каждого бита выхода узлов предобработки/постобработки от абсолютно всех бит их входа (к примеру, для «Магмы» это 64 бита).

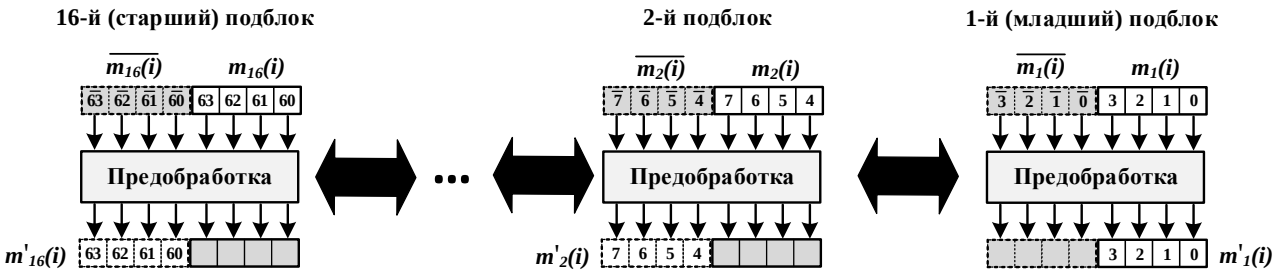


Рисунок 4.12 – Демонстрация идеи разветвления структуры узла предобработки (двухсторонняя стрелка описывает обмен битами при преобразовании различных подблоков сообщения)

Необходимо подчеркнуть очевидный факт – при добавлении операций ветвления должна сохраняться защищённость обрабатываемых байт от DPA (то есть их сбалансированность). Вместе с этим, ассемблеры микрочипов обладают весьма ограниченным набором команд, и для большей части из них реализация защищённых ветвлений, стандартным набором команд, окажется невозможной. Поэтому данный подход в большинстве случаев будет требовать доработки аппаратной части чипа, что противоречит концепции УМЗ.

Второй путь повышения защищённости от DPA – увеличение требуемой разрядности защищаемого чипа. Положим, что используется чип не с восьми, а с 16-ти разрядной архитектурой, следовательно, за один такт он способен обработать два байта данных одновременно, примем восемь младших байт информационными, а восемь старших – балансирующими (рисунок 4.13а). Преобразования в узлах предобработки/постобработки будут строго повторять оные для восьмиразрядной архитектуры, отличия будут заключаться лишь в разрядности преобразований – операции сложения по mod2 и преобразования на S'-box будут осуществляться над 16-ти разрядными комбинациями (пример для первого раунда узла предобработки приведён на рисунке 4.13б).

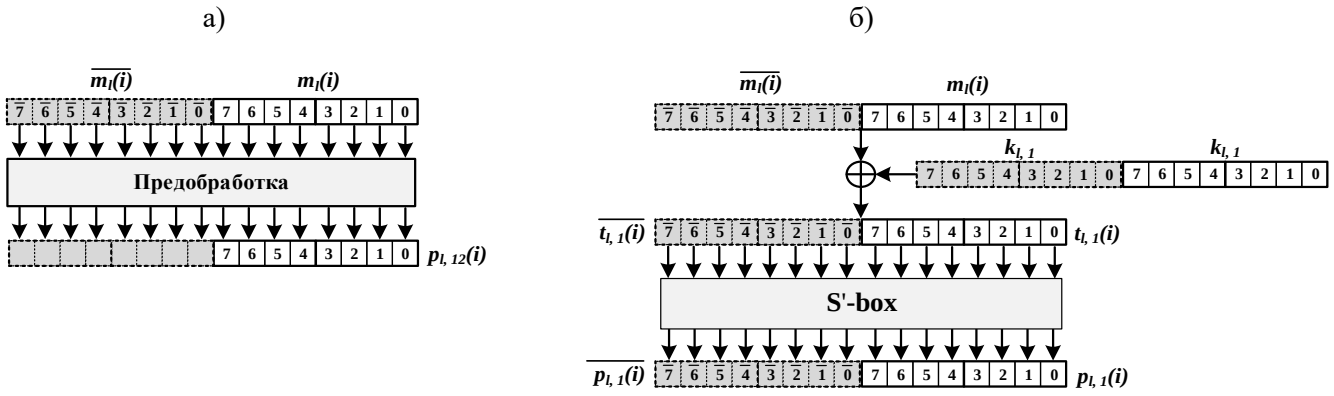


Рисунок 4.13 – Демонстрация преобразований узла предобработки, реализованного на 16-ти разрядной архитектуре: а) обобщённое представление преобразования; б) преобразование на первом раунде предобработки

В данном случае, также как и для восьмиразрядной архитектуры, информационная часть выхода узла предобработки $p_{l,\,12}(i)$ однозначно зависит от информационной части сбалансированного подблока сообщения $m_l(i)$. При использовании одинаковых подблоков $m_l(i)$, подблоки $p_{l,\,12}(i)$ также будут одинаковыми, следовательно можно составить таблицу, в которой каждому значению комбинации $m_l(i)$ сопоставляется значение $p_{l,\,12}(i)$, однако такая таблица будет иметь 256 строк вместо 16-ти. Количество возможных вариантов таких таблиц будет равно 256! (что эквивалентно приблизительно 2^{1684}) – это число является невообразимым для перебора даже при наличии неограниченных ресурсов у злоумышленника. Отсюда можно сделать вывод, что атака с перебором таблиц в отношении 16-ти (или более) разрядной архитектуры окажется нереализуемой и стойкость предлагаемого метода защиты будет определяться лишь длиной дополнительного ключа.

В приложении К показано, что защищённые S-box на ассемблере реализуются аналогично обычным S-box – с использованием подхода «чтение со смещением», а значения таблицы преобразования S'-box необходимо записывать по определённым адресам памяти EEPROM, которые также, должны быть сбалансированными. В случае, использования 16-ти разрядных S'-box окажется необходимым резервирование под таблицу преобразования 64 Кбайт памяти EEPROM, что для многих современных чипов окажется чрезвычайно большой величиной. Вместе с этим, получаемый запас устойчивости к перебору таблиц оказывается чрезвычайно большим, даже с учётом появления перспективных вычислительных систем. Поэтому вполне допустимым является усечение разрядности преобразований на узлах предобработки/постобработки до 10 бит (при использовании 16-ти битовой архитектуры чипа, шесть старших бит будут просто приниматься нулевыми). В этом случае для хранения таблицы преобразования будет необходимо резервировать 1 Кбайт памяти EEPROM, что является вполне допустимым значением для чипов среднего уровня.

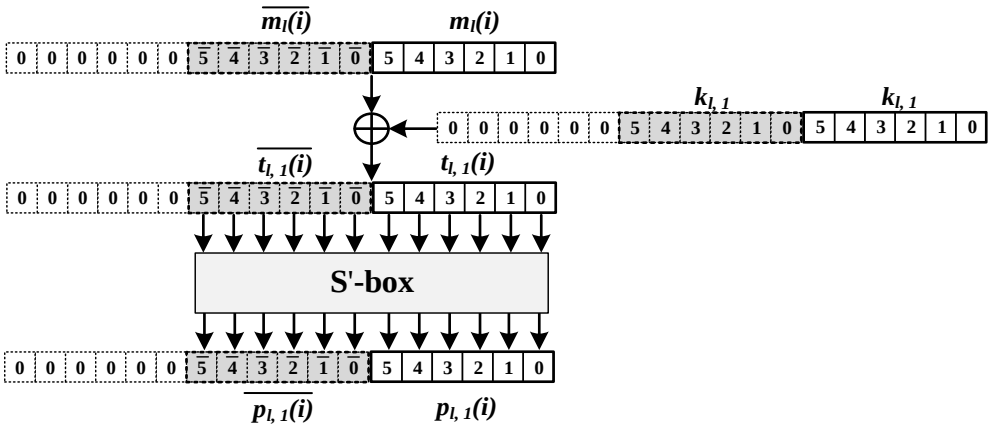


Рисунок 4.14 – Демонстрация преобразования на первом раунде узла предобработки, реализованного на 16-ти разрядной архитектуре с усечением до 10 бит

Таблица преобразования в этом случае будет состоять из 32-х строк, а количество возможных вариантов таких таблиц будет составлять 32! (что эквивалентно значению между 2^{117} и 2^{118}) – такое количество возможных комбинаций является не переборным для любых современных и перспективных вычислительных систем.

Необходимо отметить, что при использовании 10-ти разрядных преобразований на каждом раунде предобработки подблока сообщения расходуется пять бит раундового ключа, поэтому использование в узлах предобработки/постобработки 16-20-ти раундов преобразований обеспечит устойчивость к DPA в случае использования злоумышленником вычислительных систем любой мощности.

4.7 Выводы по главе

Учитывая результаты приведённых выше теоретических выкладок можно сделать следующие основные выводы:

- Предложен новый, универсальный метод защиты от DPA любых защитных блоковых преобразований (независимо от используемых в них операций).
- Предложенная структура УМЗ, использующая сбалансированные операции сложения по mod2 и преобразования на S'-box, является вполне простой в реализации, универсальной и не требующей дополнительного периферийного оборудования (например, генератора случайных чисел или второго процессора). При этом обеспечивается строго доказанный уровень защищённости от DPA, определяемый сложностью перебора дополнительного ключа. Она вполне реализуема на практике на микроконтроллерах PIC фирмы Microchip, причём не произойдёт значительного снижения скорости выполнения защищённого алгоритма, по сравнению с незащищённой реализацией.
- Недостаток УМЗ, заключающийся в отличии криптограммы, получаемой защищённой реализацией шифра от незащищённой, можно считать допустимой «платой» за описанные выше достоинства. Особенно, учитывая тот факт, что устойчивость шифра к любым иным методам криптоанализа оказывается доказуемо не ниже по сравнению с незащищённой реализацией шифра.
- Предложенная структура УМЗ допускает потенциал наращивания защищённости от DPA, за счёт увеличения разрядности выполняемых операций, увеличения количества обрабатываемых раундов, а также допускается возможность добавления дополнительных преобразований (помимо сложения по mod2 и S'-box).
- Подход с использованием УМЗ не является строго определённым методом защиты от DPA – допускается использовать различный набор защищённых операций, оптимально реализуемых на конкретном чипе. В совокупности УМЗ может позволить упростить, удешевить и унифицировать реализации защищённых от DPA преобразований на чипах. Более того, предлагаемый метод, по мере дальнейшей проработки и обсуждения может быть изменён или дополнен другими преобразованиями, обеспечивающими лучшую защищённость от атак DPA или большую скорость выполнения.

Основное содержание главы и полученных научных результатов изложено в работах автора [24; 25; 26; 33; 34].

Заключение

В настоящей диссертации была решена научно-техническая задача по разработке метода, позволяющего обеспечить защиту аппаратных реализаций защитных блоковых преобразований от побочных атак по цепям электропитания.

В процессе решения частных задач исследования и, в соответствии с целевой установкой, были получены следующие основные результаты:

1. Предложен аппаратно-программный комплекс (в основе которого лежит АЦП), позволяющий производить анализ «утечек» информации по цепи электропитания чипа и связанных с этим уязвимостей, реализованных на нём алгоритмов. Причём стоимость такой установки оказалась значительно ниже существующих решений (базирующихся на использовании осциллографа), при обеспечении высокого качества снимаемых данных об энергопотреблении.
2. Произведено экспериментальное исследование «утечек» информации по цепи электропитания анализируемого чипа в зависимости от выполняемых им операций. Дана оценка эффективности реализации DPA в отношении различных операций. На основе детального анализа результатов исследования, был сделан ряд выводов об особенностях энергопотребления анализируемого чипа, в контексте реализации атаки DPA. К наиболее важным результатам можно отнести следующее:

- Показано, что зависимости энергопотребления чипа от обрабатываемых им данных весьма далеки от идеальных теоретических моделей, они имеют значительно более сложный характер, чем представлялось при изучении известной литературы.
- Детально описана «утечка» информации по цепи электропитания чипа при выполнении разных операций. Показано, что объём «утекающей» информации может на порядки отличаться, в зависимости от выполняемой операции. Экспериментально доказано что эффективность реализации DPA в отношении различных операций, будет существенно отличаться.
- Установлено, что для ряда операций энергопотребление чипа не зависит от предварительного заполнения ячейки, в которую сохраняется результат выполнения операции, что прямо противоречит моделям, приведённым во многих литературных источниках.
- Показано, что зависимость энергопотребления чипа от обрабатываемых данных существует не только на тех тактах, на которых выполняется рассматриваемая операция, но и на 5-7 последующих тактах. Эти «остаточные» всплески представляют собой дополнительный

шум, принципиально не устранимы и они оказывают негативное влияние на возможность успешного анализа чипа.

3. Произведена сравнительная оценка эффективности подходов к реализации DPA, предполагающих расчёт ковариационных и корреляционных векторов. В результате чего выяснилось, что при рассмотрении идеализированных теоретических моделей энергопотребления чипа, разница между этими подходами оказывается незначительной. Однако, при использовании форм сигнала, полученных от реального чипа, эффективность применения подхода с расчётом корреляции, оказывается значительно выше.
4. Описана теоретическая модель атаки DPA на аппаратную реализацию шифра «Магма», после чего, её эффективность была подтверждена на практике, с использованием экспериментальных данных. В результате было доказано, что задача восстановление ключа «Магмы» не представляет значительной сложности и может быть выполнена за обозримое время (в случае если на чипе не реализованы специальные защитные механизмы).
5. Произведён сравнительный анализ сложности применения DPA к аппаратным реализациям шифров «Магма» и «Кузнечик». Показано, что при их реализации на чипах с восьмью (и более) разрядной архитектурой, «Магма» окажется более устойчивым к DPA по сравнению с «Кузнечиком».
6. Предложен новый метод защиты преобразований на интегральных чипах от побочных атак по цепям электропитания. В частности, предлагается дополнение алгоритма защитного преобразования двумя защищёнными от атаки DPA преобразованиями с дополнительным ключом: перед выполнением защитного преобразования и после него. Такая структура преобразований предъявляет низкие требования к дополнительным аппаратным ресурсам чипа, обладает простотой реализации и невысокой стоимостью. (При этом не требуется внесения изменений в аппаратную реализацию защищаемого чипа.) Обеспечивается строго доказанный уровень защищённости, определяемый вычислительной сложностью перебора дополнительного ключа. В отличие от других программных методов защиты, предлагаемый метод является универсальным, то есть он применим к защите любых аппаратно-реализованных защитных блоковых преобразований, в том числе и для российских шифров ГОСТ.

Решения поставленных частных задач исследования могут быть обобщены в следующие три основных научных результата, выносимых на защиту:

1. Архитектура средства моделирования побочных атак по цепям электропитания. Модель «утечки» информации, обрабатываемой интегральным чипом, по цепи электропитания, основанная на результатах проведённого комплекса измерений.

2. Доказательство возможности успешного применения побочных атак по цепям электропитания к аппаратным реализациям шифров ГОСТ.
3. Метод защиты преобразований на интегральных чипах от побочных атак по цепям электропитания.

Необходимо подчеркнуть, что наиболее важным научным результатом, является предложенный метод защиты от побочной атаки по цепям электропитания. Для его разработки потребовалось детальное изучение как теоретических, так и практических аспектов реализации атак этого типа (основные результаты которого сформулированы в первых трёх главах). Предложенный метод защиты обладает важной практической значимостью, заключающейся в обеспечении безопасности аппаратных реализаций, как зарубежных, так и российских защитных преобразований (прежде всего отечественных шифров, описанных стандартом ГОСТ Р 34.12-2015).

Полученные результаты являются достоверными, обладают необходимой степенью новизны, имеют теоретическую ценность и практическую значимость, а также они апробированы и опубликованы в 13-ти научных трудах.

Совокупность полученных научных результатов свидетельствует о достижении поставленной цели исследования – повышена безопасность информационно-коммуникационных технологий в части реализации защитных преобразований, выполняемых интегральными чипами.

Список сокращений и условных обозначений

ИС	интегральная схема
КМОП	комплементарная структура металл-оксид-полупроводник
ЛВ	логический вентиль (*введено автором*)
ОЗУ	оперативное запоминающее устройство (оперативная, энергозависимая память)
ОС	операционная система
ПЗУ	постоянное запоминающее устройство (энергонезависимая память)
ПК	персональный компьютер
Э/М	электромагнитное
DPA	differential power analysis (дифференциальный анализ мощности)
$E(\cdot)$	математическое ожидание (*введено автором*)
$HD(A)$	вес Хэмминга двоичной комбинации A (*введено автором*)
$HW(A, B)$	расстояние Хэмминга между двоичными комбинациями A и B (*введено автором*)
NFC	near field communication (ближняя бесконтактная связь)
SMA	sub-miniature version A (формат разъёма для подключения коаксиального кабеля)
SNR	отношение сигнал/шум
SPA	simple power analysis (простой анализ мощности)
$U_{\text{ДАН}}$	составляющая снятой формы сигнала, характеризующая энергопотребление, затрачиваемое чипом на выработку каждой единицы обрабатываемой комбинации (*введено автором*)
$Var(\cdot)$	дисперсия (*введено автором*)
$\sigma(\cdot)$	среднеквадратическое отклонение (СКО)

Словарь терминов

Атака: действия, направленные на реализацию угроз несанкционированного доступа к информации, воздействия на нее или на ресурсы автоматизированной информационной системы с применением программных и (или) технических средств [14].

Аутентификация (субъекта доступа): действия по проверке подлинности субъекта доступа в автоматизированной информационной системе [14].

Безопасность информации (при применении информационных технологий): состояние защищенности информационной технологии, обеспечивающее безопасность информации, для обработки которой она применяется, и информационную безопасность автоматизированной информационной системы, в которой она реализована [14].

Злоумышленник: субъект, пытающийся получить несанкционированный доступ к информации о секретных параметрах (нарушить секретность) системы, в частности невскрываемого чипа (*введено автором*).

Идентификация: действия по присвоению субъектам и объектам доступа идентификаторов и (или) по сравнению предъявляемого идентификатора с перечнем присвоенных идентификаторов [14].

Интегральный чип (схема): электронная схема, изготовленная на полупроводниковой подложке.

Информационная система: совокупность содержащейся в базах данных информации и обеспечивающих ее обработку информационных технологий и технических средств [12].

Конфиденциальность (информации [ресурсов автоматизированной информационной системы]): состояние информации [ресурсов автоматизированной информационной системы], при котором доступ к ней [к ним] осуществляют только субъекты, имеющие на него право [14].

Несанкционированный доступ (к информации [ресурсам автоматизированной информационной системы]); НСД: доступ к информации [ресурсам автоматизированной информационной системы], осуществляемый с нарушением установленных прав и (или) правил доступа к информации [ресурсам автоматизированной информационной системы] [14].

Ниббл: четыре бита (половина байта).

Протокол: совокупность действий (инструкций, команд, вычислений, алгоритмов), выполняемых в заданной последовательности двумя или более субъектами с целью достижения определенного результата [6].

Такт: промежуток времени между двумя импульсами тактового генератора, анализируемого чипа. (В зависимости от контекста, данный термин также употребляется в работе

для определения участка снятой формы сигнала, характеризующей энергопотребление чипа в процессе его работы на протяжении этого промежутка времени.) (*введено автором*)

Утечка (информации) по техническому каналу: неконтролируемое распространение информации от носителя защищаемой информации через физическую среду до технического средства, осуществляющего перехват информации [14].

Целостность (информации [ресурсов автоматизированной информационной системы]): состояние информации [ресурсов автоматизированной информационной системы], при котором ее [их] изменение осуществляется только преднамеренно субъектами, имеющими на него право [14].

СПИСОК ЛИТЕРАТУРЫ

Книги

1. Белов, А.В. Микроконтроллеры AVR в радиолюбительской практике [Текст] / А.В. Белов. – СПб. : Наука и техника, 2007. – 352 с. : ил. – ISBN 978-5-94387-365-2.
2. Ван дер Варден, Б.Л. Математическая статистика [Текст] / Б.Л. Ван дер Варден; пер. с немецкого Большева Л.Н. – М. : Издательство иностранной литературы, 1960. – 436 с. : ил.
3. Кацтен, С. PIC микроконтроллеры. Полное руководство [Текст] / С. Катцен; пер. с англ. Евстифеева А.В. – М. : Додэка-XXI, 2010. – 656 с. : ил. (Серия «Программируемые системы»). Доп. тит. л. англ. – ISBN 978-5-94120-218-8.
4. Коржик, В.И. Основы криптографии: учебное пособие [Текст] / В.И. Коржик, В.А. Яковлев. – СПб. : ИЦ Интермедия, 2016. – 296 с. : ил. – ISBN 978-5-89160-097-3.
5. Микушин, А.В. Цифровые устройства и микропроцессоры: учеб. пособие [Текст] / А.В. Микушин, А.М. Сажнев, В.И. Сединин. – СПб. : БХВ-Петербург, 2010. – 832 с.: ил. – (Учебная литература для вузов). – ISBN 978-5-9775-0417-1.
6. Молдовян, А.А. Криптография: скоростные шифры [Текст] / А.А. Молдовян, Н.А. Молдовян, Б.В. Изотопов, Н.Д. Гуц. – СПб. : БХВ-Петербург, 2002. – 495 с.
7. Ревич, Ю.В. Практическое программирование микроконтроллеров Atmel AVR на языке ассемблера [Текст] / Ю.В. Ревич. – 2-е изд., испр. – СПб. : БХВ-Петербург, 2011. – 352 с.: ил. – (Электроника). – ISBN 978-5-9775-0657-1.
8. Топильский, В.Б. Схемотехника аналого-цифровых преобразователей. Учебное издание [Текст] . В.Б. Топильский. – М.: Техносфера, 2014. – 288 с. – ISBN 978-5-94836-383-7
9. Шнайер, Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си [Текст] / Б. Шнайер; пер. с англ. – М. : Триумф, 2002. – 816 с. – ISBN 5-89392-055-4.
10. Mangard, S. Power Analysis Attaks: Revealing the Secrers of Smart Card. / S. Mangard, E. Oswald, T. Popp. – [USA] : Springer US, 2017. – 338 p. – ISBN 978-0-387-30857-9.
11. Peeters, E. Advanced DPA Theory and Practice: Towards the Security Limits of Secure Embedded Circuits / E. Peeters. – New York : Springer-Verlag New York, 2013. – 139 p. – ISBN 978-1-4614-6782-3.

Нормативно-правовые акты

12. Федеральный закон от 27.07.2006 N 149-ФЗ (ред. от 31.12.2014) «Об информации, информационных технологиях и о защите информации» (с изм. и доп., вступ. в силу с 01.09.2015). – 2015.

Стандарт

13. Информационная технология. Криптографическая защита информации. Блочные шифры [Текст]: ГОСТ Р 34.12-2015. введ. 19.06.2015.
14. Информационные технологии. Основные термины и определения в области технической защиты информации [Текст]: ГОСТ Р 50.1.053-2005. введ. 06.04.2005.

Отчёты о научно-исследовательской работе

15. Тихонов, С.В. Исследование основных направлений создания средств и систем защиты электронных документов нового поколения: отчёт о НИР / М.М. Позднышев, А.Г. Бородинов, С.В. Тихонов [и др.] . – СПб.: АО «Научные приборы», 2014; инв. №231-АО от 16.11.2015 г. – 323 с.

Электронные ресурсы

16. Пробники от А до Я. Учебное пособие [Электронный ресурс]: Контрольно-измерительное оборудование | Tektronix. – Режим доступа: http://ru.tek.com/document/primer/abcs-probes (дата обращения: 15.05.2017).
17. Суперкомпьютер «Ломоносов-2» [Электронный ресурс]: Т-Платформы | T-Platforms. – Режим доступа: http://www.t-platforms.ru/solutions/completed/lomonosov2-supercomputer (дата обращения: 15.05.2017).
18. УСТРОЙСТВО МИКРОКОНТРОЛЛЕРОВ AVR [Электронный ресурс]: myROBOT - Роботы, робототехника, микроконтроллеры, программирование AVR. – Режим доступа: https://myrobot.ru/stepbystep/mc_architecture.php (дата обращения: 15.05.2017).
19. ASIC basics tutorial [Электронный ресурс]: Radio-Electronics.com: resources, analysis & news for electronics engineers. – Режим доступа: http://www.radio-electronics.com/info/data/semicond/asic/asic.php (дата обращения: 17.08.2017).
20. ATmega16A [DATASHEET] [Электронный ресурс]: Atmel Corporation - Microcontrollers, 32-bit, and touch solutions. – Режим доступа: http://www.atmel.com/images/atmel-8154-8-bit-avr-atmega16a_datasheet.pdf (дата обращения: 15.05.2017).
21. Hnath, W. Differential Power Analysis Side-Channel Attacks in Cryptography [Электронный ресурс] / W. Hnath, J. Pettengill. – 2010. – 42 с. – Режим доступа: https://pdfs.semanticscholar.org/f6bc/06ad389ccd63614f9e2d882ca1d8b9042cae.pdf (дата обращения: 10.05.2017).
22. National Supercomputing Center in Wuxi [Электронный ресурс]: 国家超级计算无锡中心. – Режим доступа: http://demo.wxmax.cn/wxc/introduction.php?word=introduction&i=34 (дата обращения: 15.05.2017).

Статьи

23. Тихонов, С.В. О возможности взлома аппаратной реализации шифра ГОСТ / В.И. Коржик, С.В. Тихонов // Проблемы информационной безопасности. Компьютерные системы. – 2012. – № 3. – С. 53-62.
24. Тихонов, С.В. Исследование методов защиты шифра ГОСТ от побочной атаки, использующей контроль потребляемой мощности / В.И. Коржик, С.В. Тихонов // Актуальные проблемы инфотелекоммуникаций в науке и образовании (АПИНО-2013): сборник научных статей II Международной научно-технической и научно-методической конференции. – 2013. – С. 829-831.
25. Тихонов, С.В. Метод защиты аппаратной реализации шифра ГОСТ от атаки измерения потребляемой мощности в цепи питания / С.В. Тихонов, В.И. Коржик // Проблемы информационной безопасности. Компьютерные системы. – 2013. – № 3. – С. 62-72.
26. Тихонов, С.В. Разработка алгоритма предобработки аппаратно-реализованных блоковых шифров, обеспечивающих защищённость от атак DPA, при минимальных требованиях к аппаратным ресурсам / С.В. Тихонов // Актуальные проблемы инфотелекоммуникаций в науке и образовании (АПИНО-2014): сборник научных статей III Международной научно-технической и научно-методической конференции. – 2014. – С. 364-368.
27. Тихонов, С.В. Вторая революция в криптографии: миф или реальность? / В.И. Коржик, В.А. Яковлев, С.В. Тихонов // Актуальные проблемы инфотелекоммуникаций в науке и образовании (АПИНО-2015): сборник научных статей IV Международной научно-технической и научно-методической конференции [Текст] : в 2 т. – 2015. – С. 10-17. – 1 т.
28. Тихонов, С.В. Экспериментальное исследование взлома аппаратно реализованных блоковых шифров по цепям электропитания / С.В. Тихонов // Методы и технические средства обеспечения безопасности информации: Материалы 24-й научно-технической конференции 29 июня – 02 июля 2015 года. – 2015. – С. 100-102.
29. Тихонов, С.В. Бесключевая криптосистема с обеспечением секретности на физическом уровне: миф или реальность? / В.И. Коржик, В.А. Яковлев, С.В. Тихонов // Проблемы информационной безопасности. Компьютерные системы. – 2015. – № 4. – С. 79-89.
30. Тихонов, С.В. Экспериментальное исследование утечек информации, обрабатываемой интегральными чипами, по цепи электропитания / С.В. Тихонов // Проблемы информационной безопасности. Компьютерные системы. – 2016. – № 1. – С. 82-93.
31. Тихонов, С.В. Экспериментальное исследование возможности взлома по цепям электропитания аппаратно реализованных блоковых шифров / С.В. Тихонов // Проблемы информационной безопасности. Компьютерные системы. – 2016. – № 2. – С. 106-117.

32. Тихонов, С.В. Исследование практической возможности взлома по цепям электропитания аппаратно реализованных шифров на примере ГОСТ 28147-89 / С.В. Тихонов // Актуальные проблемы инфотелекоммуникаций в науке и образовании (АПИНО-2016): сборник научных статей V Международной научно-технической и научно-методической конференции [Текст] : в 3 т. – 2016. – С. 534-539. – 1 т.
33. Тихонов, С.В. Универсальный метод защиты блочных шифров от побочных атак по цепям питания / С.В. Тихонов // Актуальные проблемы инфотелекоммуникаций в науке и образовании (АПИНО-2017): сборник научных статей V Международной научно-технической и научно-методической конференции [Текст] : в 3 т. – 2017. – 2 т.
34. Тихонов, С.В. Универсальный метод защиты блочных шифров от побочных атак по цепям питания / С.В. Тихонов // Проблемы информационной безопасности. Компьютерные системы. – 2017. – № 3.
35. Akkar, M. A generic protection against high-order differential power analysis / M. Akkar, L. Goubin // Johansson T. (eds) Fast Software Encryption. FSE 2003. Lecture Notes in Computer Science. – 2003. – Vol. 2887. – P. 192-205.
36. Akkar, M. An implementation of DES and AES secure against some attacks / M. Akkar, C. Giraud // Koç Ç.K., Naccache D., Paar C. (eds) Cryptographic Hardware and Embedded Systems — CHES 2001. CHES 2001. Lecture Notes in Computer Science. – 2001. – Vol. 2162. – P. 309-318.
37. Akkar, M. Power Analysis, What Is Now Possible... / M. Akkar, R. Bevan, P. Dischamp, D. Moyart // Okamoto T. (eds) Advances in Cryptology — ASIACRYPT 2000. ASIACRYPT 2000. Lecture Notes in Computer Science. – 2000. – Vol. 1976 – P. 489–502.
38. Akkar, M. Two Power Analysis Attacks against One-Mask Methods / M. Akkar, R. Bévan, L. Goubin // Roy B., Meier W. (eds) Fast Software Encryption. FSE 2004. Lecture Notes in Computer Science. – 2004. – Vol. 3017. – P. 332-347.
39. Ambrose, J.A. Multiprocessor information concealment architecture to prevent power analysis-based side channel attacks / J.A. Ambrose, R.G. Ragel, S. Parameswaran, A. Ignjatovic // IET Computers & Digital Techniques. – 2011. – Vol. 5. – Iss.1. – P. 1-15.
40. Arora, A. A double-width algorithmic balancing to prevent power analysis Side Channel Attacks in AES / A. Arora, J.A. Ambrose, J. Peddersen, S. Parameswaran // IEEE Computer Society Annual Symposium on VLSI. – 2013. – P. 76-83.
41. Balasch, J. Power Analysis of Atmel CryptoMemory – Recovering Keys from Secure EEPROMs / J. Balasch, B. Gierlichs, R. Verdult, L. Batina, I. Verbauwhede // Dunkelman O. (eds) Topics in Cryptology – CT-RSA 2012. CT-RSA 2012. Lecture Notes in Computer Science. – 2012. – Vol. 7178. – P. 19-34.

42. Brier, E. Correlation Power Analysis with a Leakage Model / E. Brier, C. Clavier, F. Olivier // Joye M., Quisquater JJ. (eds) Cryptographic Hardware and Embedded Systems - CHES 2004. CHES 2004. Lecture Notes in Computer Science. – 2004. –Vol. 3156. – P. 16-29.
43. Clavier, C. Differential Power Analysis in the Presence of Hardware Countermeasures / C. Clavier, JS. Coron, N. Dabbous // Koç Ç.K., Paar C. (eds) Cryptographic Hardware and Embedded Systems — CHES 2000. CHES 2000. Lecture Notes in Computer Science. – 2000. – Vol. 1965 – P. 252-263.
44. Eisenbarth, T. On the Power of Power Analysis in the Real World: A Complete Break of the KeeLoq Code Hopping Scheme / T. Eisenbarth, T. Kasper, A. Moradi, C. Paar, M. Salmasizadeh, M.T.M. Shalmani // Wagner D. (eds) Advances in Cryptology – CRYPTO 2008. CRYPTO 2008. Lecture Notes in Computer Science. – 2008. – Vol. 5157. – P. 203-220.
45. Guilley, S. Differential Power Analysis Model and Some Results / S. Guilley, P. Hoogvorst, R. Pacalet // Quisquater JJ., Paradinas P., Deswarte Y., El Kalam A.A. (eds) Smart Card Research and Advanced Applications VI. IFIP International Federation for Information Processing. – 2004. – Vol. 153. – P. 127-142.
46. Jaffe, J. More Differential Power Analysis: Selected DPA Attacks / J. Jaffe // ECRYPT Summerschool on Cryptographic Hardware, Side Channel and Fault Analysis. – 2006.
47. Joye, M. Side-Channel Analysis / M.Joye, F.Olivier // H. van Tilborg, Ed., Encyclopedia of Cryptography and Security. – 2005. – P. 571-576.
48. Kocher, P. Differential Power Analysis / P. Kocher, J. Jaffe, B. Jun // Wiener M. (eds) Advances in Cryptology — CRYPTO' 99. CRYPTO 1999. Lecture Notes in Computer Science. – 1999. – Vol. 1666. – P. 388-397.
49. Lv, J. On Two DES Implementations Secure against Differential Power Analysis in Smart-Cards / J. Lv // Information and Computation. – 2006. – Vol. 204. – Iss. 7. – P. 1179–1193.
50. Mace, F. A dynamic current mode logic to counteract power analysis atacks / F. Mace, F.-X. Standaert, I. Hassoune, J.-D. Legat, J.-J. Quisquater // Proceedings of DCIS 2004. – 2004. – P. 186-191.
51. Mangard, S. A Simple Power-Analysis (SPA) Attack on Implementations of the AES Key Expansion / S. Mangard // Lee P.J., Lim C.H. (eds) Information Security and Cryptology – ICISC 2002. ICISC 2002. Lecture Notes in Computer Science. – 2003. – Vol 2587. – P. 343-358.
52. Masoumi, M. Novel Approach to Protect Advanced Encryption Standard Algorithm Implementation Against Differential Electromagnetic and Power Analysis / M. Masoumi, M.H. Rezayati // IEEE Transactions on Information Forensics and Security. – 2015. – Vol. 10. – Vol. 2. – P. 256-265.
53. May, D. Random register renaming to foil DPA / D. May, H.L. Muller, N.P. Smart // Koç Ç.K., Naccache D., Paar C. (eds) Cryptographic Hardware and Embedded Systems — CHES 2001. CHES 2001. Lecture Notes in Computer Science. – 2001. – Vol 2162. – P. 28-38.

54. Mayer-Sommer, R. Smartly Analyzing the Simplicity and the Power of Simple Power Analysis on Smartcards / R. Mayer-Sommer // Koç Ç.K., Paar C. (eds) Cryptographic Hardware and Embedded Systems — CHES 2000. CHES 2000. Lecture Notes in Computer Science. – 2000. –Vol 1965. – P. 78-92.
55. Messerges, T. Investigations of power analysis attacks on smartcards / T. Messerges, T. Dabbish, R. Sloan // Proceedings of the USENIX Workshop on Smartcard Technology. – 1999. – P. 151-162.
56. Moore, S. W. Improving smart card security using self-timed circuits / S.W. Moore, R.D. Mullins, P.A. Cunningham, R.J. Anderson, G.S. Taylor // Proceedings Eighth International Symposium on Asynchronous Circuits and Systems (ASYNC '02). – 2002. – P. 211-218.
57. Moradi, A. On the vulnerability of FPGA bitstream encryption against power analysis attacks: extracting keys from xilinx Virtex-II FPGAs / A. Moradi, A. Barenghi, T. Kasper, H. Görtz, C. Paar // Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS 2011). – 2011. – P. 111-124.
58. Neve, A. Feasibility of Smart Cards in Silicon-On-Insulator (SOI) Technology / A. Neve, D. Flandre, J.-J. Quisquater // Proceedings of the USENIX Workshop on Smartcard Technology. – 1999. – P. 1-7.
59. Oswald, D. Breaking Mifare DESFire MF3ICD40: Power Analysis and Templates in the Real World / D. Oswald, C. Paar // Preneel B., Takagi T. (eds) Cryptographic Hardware and Embedded Systems – CHES 2011. CHES 2011. Lecture Notes in Computer Science. – 2011. – Vol 6917. – P. 207-222.
60. Ratanpal, G.B. An on-chip signal suppression countermeasure to power analysis attacks / G.B. Ratanpal, R.D. Williams, T.N. Blalock // IEEE Transactions on Dependable and Secure Computing. – 2004. – Vol. 1. – Iss. 3. – P. 179–189.
61. Skorobogatov, S.P. Semi-invasive attacks – A new approach to hardware security analysis. / S.P. Skorobogatov // Technical Report UCAM-CL-TR-630. University of Cambridge. Computer Laboratory. – 2005. – 144 p.
62. Tiri, K. A dynamic and differential CMOS logic with signal independent power consumption to withstand differential power analysis on smart cards / K. Tiri, M. Akmal, I. Verbauwhede // Proceedings of ESSCIRC 2002. – 2002. – P. 403-406.
63. Tunstall, M. Correlation power analysis of large word sizes / M. Tunstall, N. Hanley, R. McEvoy, C. Whelan, C.C. Murphy, W.P. Marnane // IET Irish Signals and Systems Conference (ISSC) 2007. – 2007.
64. Tunstall, M. Efficient Use of Random Delays in Embedded Software / M. Tunstall, O. Benoit // Sauveron D., Markantonakis K., Bilas A., Quisquater JJ. (eds) Information Security Theory and Practices. Smart Cards, Mobile and Ubiquitous Computing Systems. WISTP 2007. Lecture Notes in Computer Science. – 2007. – Vol. 4462. – P. 27-38.

Приложение А. Краткое описание принципа работы параллельных АЦП

На рисунке А.1 показана принципиальная схема параллельного АЦП с восемью уровнями квантования. Принцип действия таких АЦП очень прост [5, с. 297-299; 8, с. 61-63]: на вход подаётся опорное напряжение $V_{\text{ОП}}$ и аналоговый сигнал $V_{\text{СИГН}}$, который необходимо преобразовать в цифровой код. Опорное напряжение, при помощи резистивного делителя, состоящего из резисторов одинакового номинала, делится на семь одинаковых уровней. Преобразуемый сигнал поступает на неинвертирующие входы компараторов, опорный сигнал, через делители напряжения подаётся на инвертирующие входы компараторов. Компараторы сравнивают входной сигнал АЦП с опорным напряжением – если напряжение на неинвертирующем входе компаратора превышает напряжение на его инвертирующем входе, то на выходе компаратора формируется напряжение логической единицы, иначе – логического нуля.

Если напряжение, преобразуемого сигнала на неинвертирующем входе АЦП окажется меньше всех опорных напряжений, подаваемых на инвертирующие входы компараторов, то на выходах всех компараторов будут нулевые логические уровни сигнала. Повышая уровень входного сигнала до напряжения, превышающего напряжение на инвертирующем входе нижнего компаратора, на выходе последнего сформируется уровень логической единицы, а на выходе АЦП сформируется код 001. При дальнейшем увеличении уровня измеряемого сигнала, код будет принимать значения: 010, 011, и так далее. Максимальное значение на выходе АЦП 111, сформируется при превышении входным сигналом значения опорного сигнала верхнего компаратора.

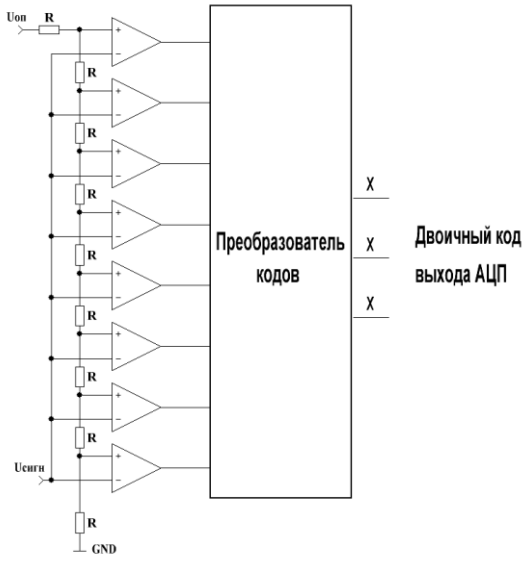


Рисунок А.1 – Принципиальная схема параллельного АЦП

АЦП, представленный на рисунке А.1, стоит из восьми компараторов, он позволяет представить в двоичном коде (цифровом виде) напряжение от 0 до $V_{\text{ОП}}$ восьмью уровнями с шагом $\frac{1}{7}\cdot V_{\text{ОП}}$: 0, $\frac{1}{7}\cdot V_{\text{ОП}}$, $\frac{2}{7}\cdot V_{\text{ОП}}$, , $\frac{4}{7}\cdot V_{\text{ОП}}$, $\frac{5}{7}\cdot V_{\text{ОП}}$, $\frac{6}{7}\cdot V_{\text{ОП}}$, $\frac{7}{7}\cdot V_{\text{ОП}} = V_{\text{ОП}}$. Напряжение будет описываться трёхзначной двоичной комбинацией от 000 до 111 – это трёхразрядный АЦП.

Приведённый пример хорошо иллюстрирует погрешность АЦП, называемую ошибкой квантования. Квантуемый сигнал, описывается всего восьмью уровнями напряжения, поэтому значения сигнала, находящиеся между двумя уровнями, округляются до одного из них. Эта ошибка определяется шагом квантования, в приведённом примере он составляет $\frac{1}{7}\cdot V_{\text{ОП}}$. Ошибки квантования являются следствием ограниченного разрешения АЦП, присущи всем АЦП и принципиально не устранимы.

Данное приложение является справочным, оно основано на выкладках, приведённых в следующих работах: [5; 8].

Приложение Б. Краткое описание принципа работы АЦП конвейерного типа

Блок-схема АЦП конвейерного типа приведена на рисунке Б.1.

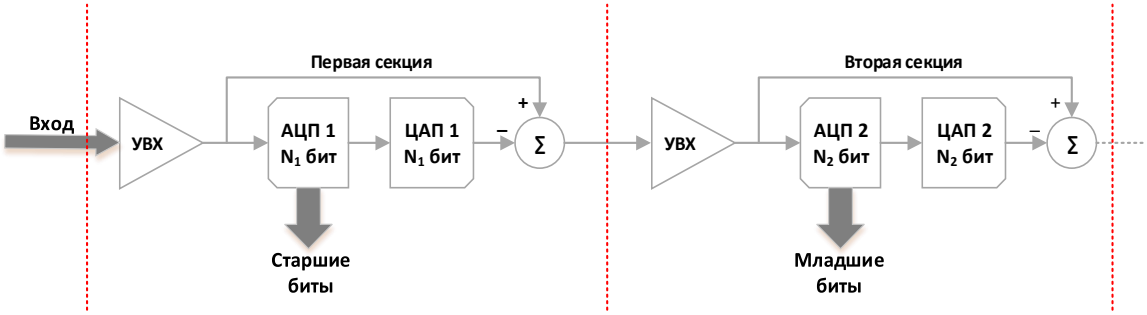


Рисунок Б.1 – Блок схема АЦП конвейерного типа

В АЦП конвейерного типа [5, с. 299-302; 8, с. 63-68] дискретное значение сигнала запоминается в устройстве выборки-хранения (УВХ), затем оно преобразуется в код при помощи параллельного АЦП 1 первой секции, при этом данный АЦП имеет относительно малую разрядность N_1. После этого, полученный код, при помощи цифро-аналогового преобразователя преобразуется обратно в напряжение представляющее очень грубое приближение к реальному сигналу. Которое вычитается из точного значения напряжения с УВХ, поступающего на сумматор. Полученная разность усиливается и запоминается в УВХ второй секции, затем при помощи АЦП второй секции с разрядностью N_2 преобразуется в код. Таким образом имея две восьми разрядные секции (по 255 компараторов в каждой), можно получить 16-ти разрядный АЦП. Отсюда АЦП конвейерного типа, состоящий из 510 компараторов, обеспечивает то же разрешение, что и параллельный АЦП, состоящий из 65 535 компараторов. Частота квантования таких АЦП конечно будет ниже, чем параллельных, однако на сегодняшний день для 16-ти битовых АЦП конвейерного типа возможна работа на частотах 200-400 МГц. Стоит заметить, что количество компараторов может быть снижено ещё сильнее, за счёт увеличения числа секций, однако это неминуемо приведёт к снижению частоты квантования.

Данное приложение является справочным, оно основано на выкладках, приведённых в следующих работах: [5; 8].

Приложение В. Краткое описание принципа работы микроконтроллера Atmel ATmega16

Из технической документации микроконтроллера Atmel ATmega16 [20] известно, что он имеет Гарвардскую архитектуру, то есть имеется отдельная память и шина адреса для программ и для данных, следовательно, за каждый такт (начиная с прихода нарастающего фронта тактового сигнала) осуществляется выполнение текущей инструкции и выборка из памяти программ следующей (рисунок В.1).

Рисунок В.1 – Этапы выполнения инструкций микроконтроллером Atmel ATmega 16

Память программ (Flash память) микроконтроллера предназначена для хранения последовательности команд, управляющих его функционированием, и имеет 16-ти битовую организацию. Число циклов стирание/запись – не менее 10 тыс [18].

Память данных разделена на три области: регистровая память, ОЗУ и EEPROM.

Регистровая память [Там же] включает 32 рабочих регистра, и служебные регистры портов ввода/вывода. Все они расположены в адресном пространстве ОЗУ, но не являются его частью, они имеют байтовый формат. В области регистров ввода/вывода расположены различные служебные регистры (регистры управления микроконтроллером, регистры статуса и тому подобные), а также регистры управления периферийными устройствами, входящими в состав микроконтроллера. По сути, управление микроконтроллером заключается в управлении этими регистрами.

Для долговременного хранения различной информации, которая может изменяться в процессе функционирования микроконтроллера, используется EEPROM-память. Этот тип памяти, удобен для хранения промежуточных данных, различных констант, коэффициентов, серийных номеров, ключей и подобного, EEPROM может быть записана как при программировании микроконтроллера, так и в процессе его функционирования. Количество циклов стирание/запись – не менее 100 тыс. [Там же].

ОЗУ используется для оперативного хранения данных. Число циклов чтения и записи в ОЗУ не ограничено, но при отключении питающего напряжения вся информация теряется. Отличие

между рабочими регистрами и оперативной памятью состоит в том, что с регистрами можно производить любые операции, а в отношении оперативной памяти доступны лишь операции чтения/записи данных из/в регистры [Там же].

Все команды обрабатывают данные (операнды) в виде байтов, содержащихся в любом из 32-ух рабочих регистров микроконтроллера, в которые они могут быть загружены и из которых они могут быть выгружены в ОЗУ и EEPROM. Непосредственное выполнение команд осуществляется в АЛУ. После загрузки программы микроконтроллера в память программ, в специальный регистр, называемый счётчиком команд, записывается адрес ячейки памяти программ, содержащей первую команду. В процессе выполнения программы, на первом такте, АЛУ считывает из счётчика команд адрес ячейки памяти программ (производится выборка первой инструкции), и инкремент счётчика команд, а на втором такте производится выполнение первой команды, и одновременно с этим производится чтение второй команды по адресу, на который указывает счётчик команд, также производится инкремент счётчика команд, и так далее. Бо**л**ьшая часть команд выполняется микроконтроллерами Atmel за один такт, команды обращения к ОЗУ, ПЗУ, а также команды передачи управления, как-правило выполняются за 2-3 такта. В документации к микроконтроллеру ATmega16 [20] дан пример выполнения «простой» команды, выполняемой за один такт (рисунок В.2).

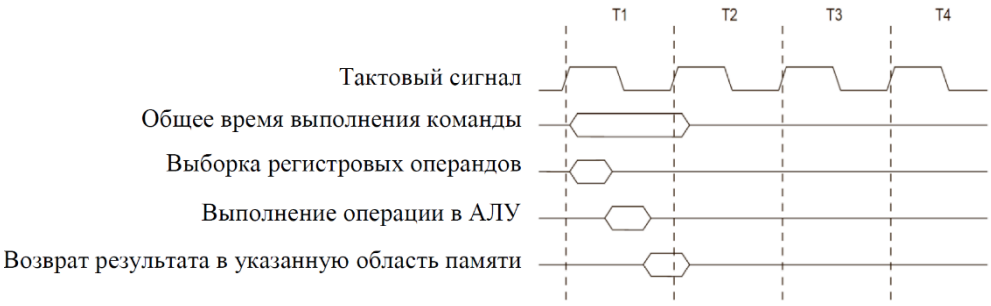


Рисунок В.2 – Этапы выполнения инструкции микроконтроллером Atmel ATmega 16

Таким образом реализуется, к примеру операция сложения по mod2 (XOR), она может быть осуществлена только с байтами, находящимися в двух рабочих регистрах (в рабочие регистры, данные, предварительно могут быть загружены из ОЗУ или ПЗУ, при помощи соответствующих команд). Подчеркнём, что параллельно с выполнением команды, схематически показанным на рисунке В.2, осуществляется чтение из памяти программ следующей команды. (Данная особенность гарвардской архитектуры приведёт к наложению алгоритмического шума на рассматриваемые отсчёты, так как на значения напряжения, характеризующие энергопотребление, в процессе выполнения интересующей команды, будут накладываться значения напряжения, относящиеся к следующей команде.) Итак, на первом этапе, из двух

регистров производится выборка байт, которые необходимо сложить по mod2 – они подаются на АЛУ, на втором этапе АЛУ осуществляет операцию сложения по mod2, а на третьем этапе, результат поступает с АЛУ в один из участвующих в операции рабочих регистров, перезаписывая одно из слагаемых (в дальнейшем результат может быть оставлен в регистре или скопирован в ОЗУ/ПЗУ, при помощи соответствующих команд).

Данное приложение является справочным, оно основано на выкладках, приведённых в следующих работах: [1; 3; 7; 18; 20].

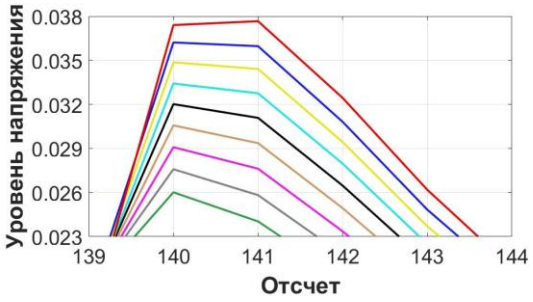
Приложение Г. Характеристики отсчётов форм сигнала при выполнении операции записи в чистую ячейку ОЗУ

Таблица Г.1 – Зависимость среднего значения и СКО 140-го и 190-го отсчётов от веса Хэмминга байта данных пересылаемого из ячейки рабочего регистра в чистую ячейку ОЗУ

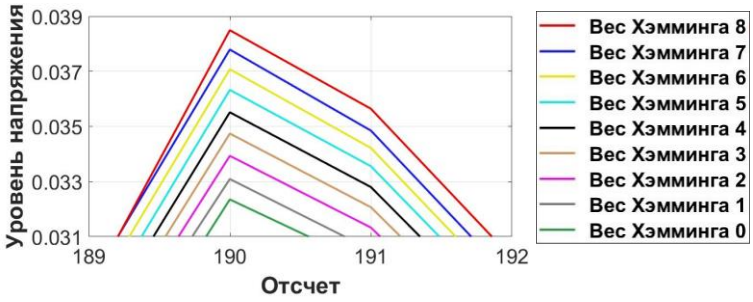
	140-й отсчёт		**190-й отсчёт**	
Вес Хэмминга	Среднее значение	СКО	Среднее значение	СКО
0	0.02601 (46.8 мВ)	0.00099 (1.78 мВ)	0.03234 (58.2 мВ)	0.00148 (2.67 мВ)
1	0.02759 (49.7 мВ)	0.00097 (1.75 мВ)	0.03309 (59.6 мВ)	0.00159 (2.87 мВ)
2	0.02908 (52.4 мВ)	0.00108 (1.95 мВ)	0.03393 (61.1 мВ)	0.00163 (2.95 мВ)
3	0.03057 (55.0 мВ)	0.00122 (2.19 мВ)	0.03474 (62.5 мВ)	0.00163 (2.95 мВ)
4	0.03202 (57.6 мВ)	0.00135 (2.43 мВ)	0.03551 (63.9 мВ)	0.00164 (2.97 мВ)
5	0.03342 (60.1 мВ)	0.00153 (2.75 мВ)	0.03632 (65.4 мВ)	0.00169 (3.04 мВ)
6	0.03486 (62.8 мВ)	0.00164 (2.96 мВ)	0.03707 (66.7 мВ)	0.00164 (2.96 мВ)
7	0.03622 (65.2 мВ)	0.00191 (3.43 мВ)	0.03779 (68.0 мВ)	0.00174 (3.14 мВ)
8	0.03740 (67.3 мВ)	0.00206 (3.71 мВ)	0.03849 (69.3 мВ)	0.00150 (2.71 мВ)

Приложение Д. Графики, характеризующие зависимости энергопотребления анализируемого микроконтроллера от выполняемых операций и обрабатываемых данных

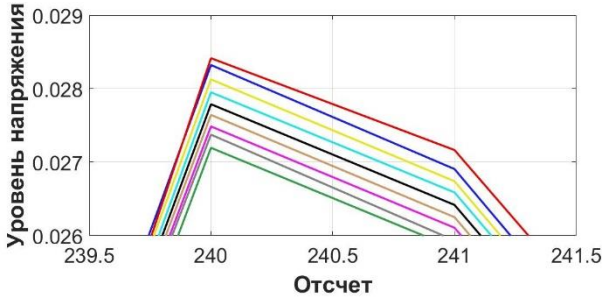
а)



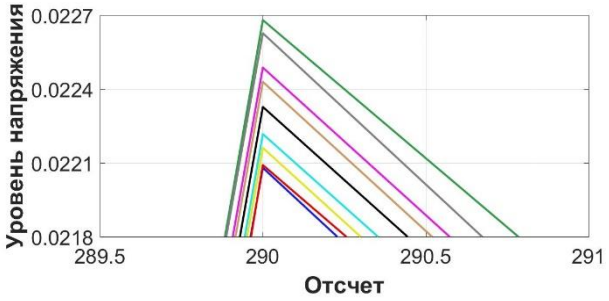
б)



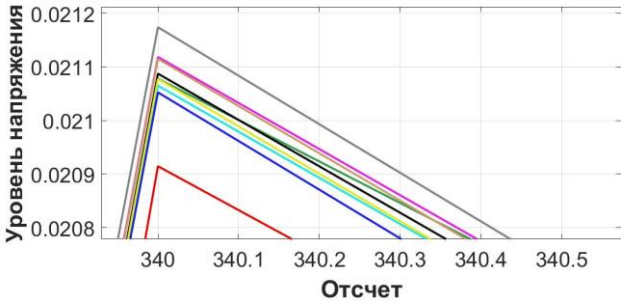
в)



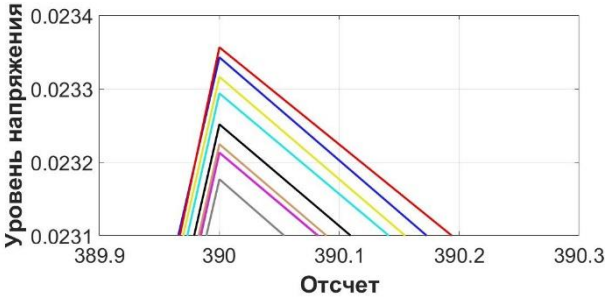
г)



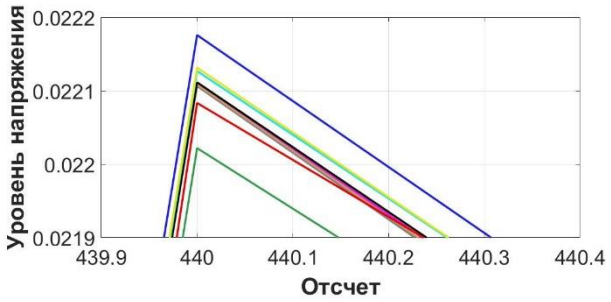
д)



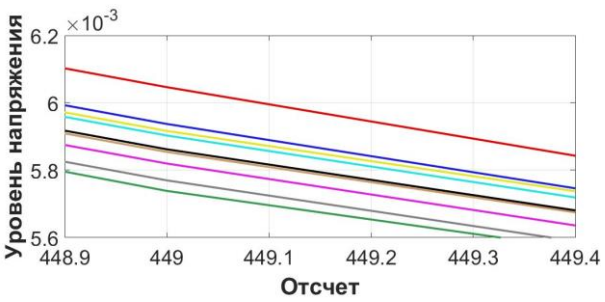
е)



ж)



и)



к) л)

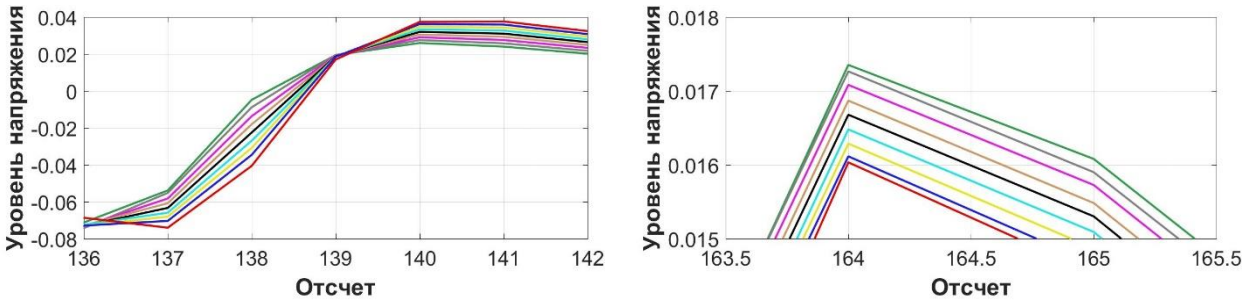
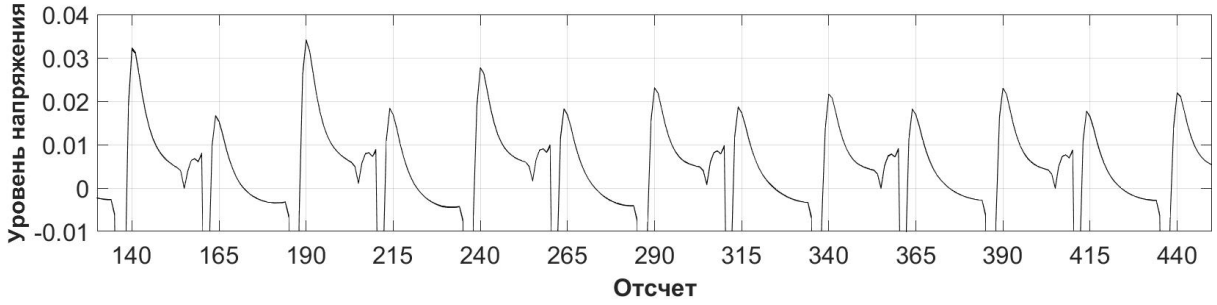
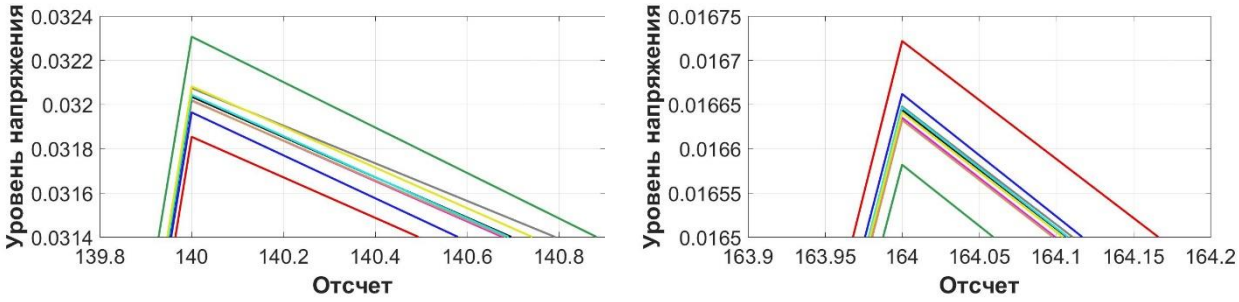


Рисунок Д.1 – Увеличенные области усреднённых форм сигнала, характеризующих зависимость энергопотребления микроконтроллера ATmega 16 от веса Хэмминга комбинации, записываемой в чистую ячейку ОЗУ

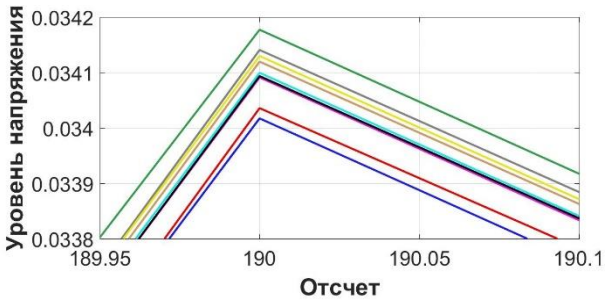
а)



б) в)



г)



д)

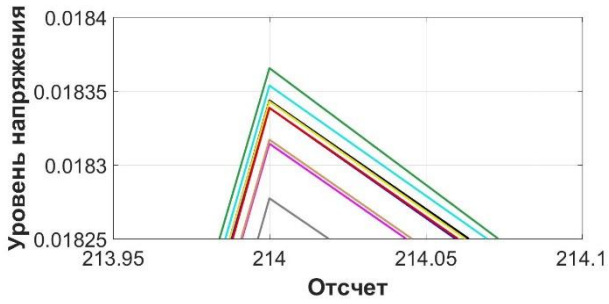
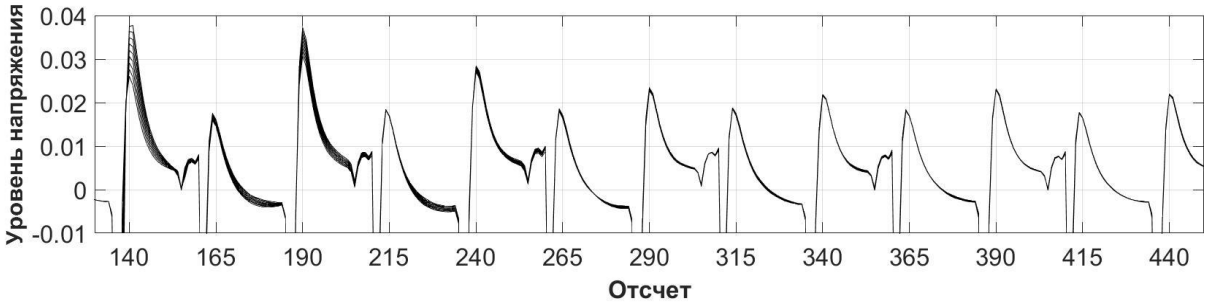
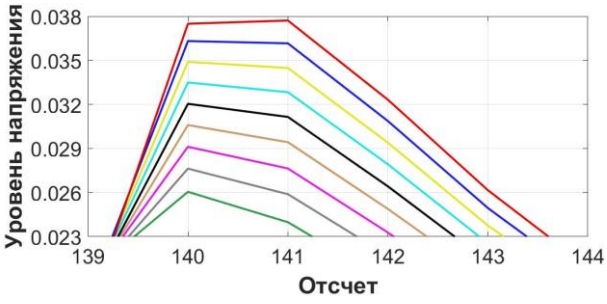


Рисунок Д.2 – Увеличенные области усреднённых форм сигнала, характеризующих зависимость энергопотребления микроконтроллера ATmega 16 от расстояния Хэмминга между записываемой и перезаписываемой комбинациями, при выполнении операции записи в предварительно заполненную ячейку ОЗУ

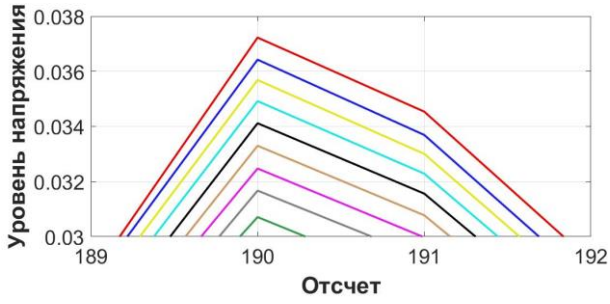
а)



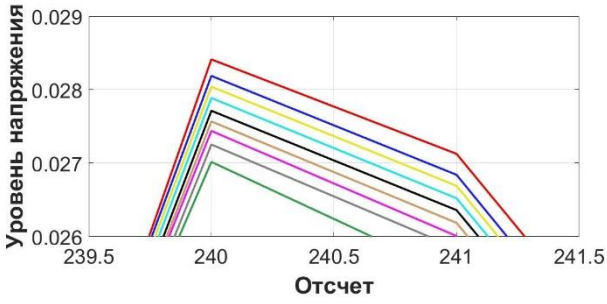
б)



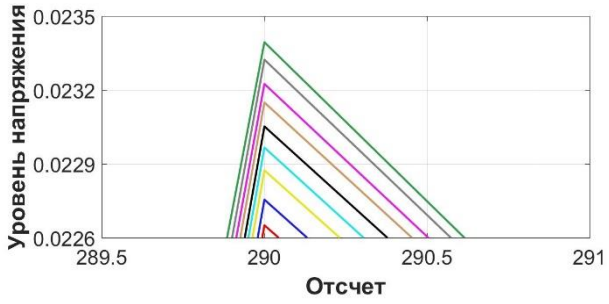
в)



г)



д)



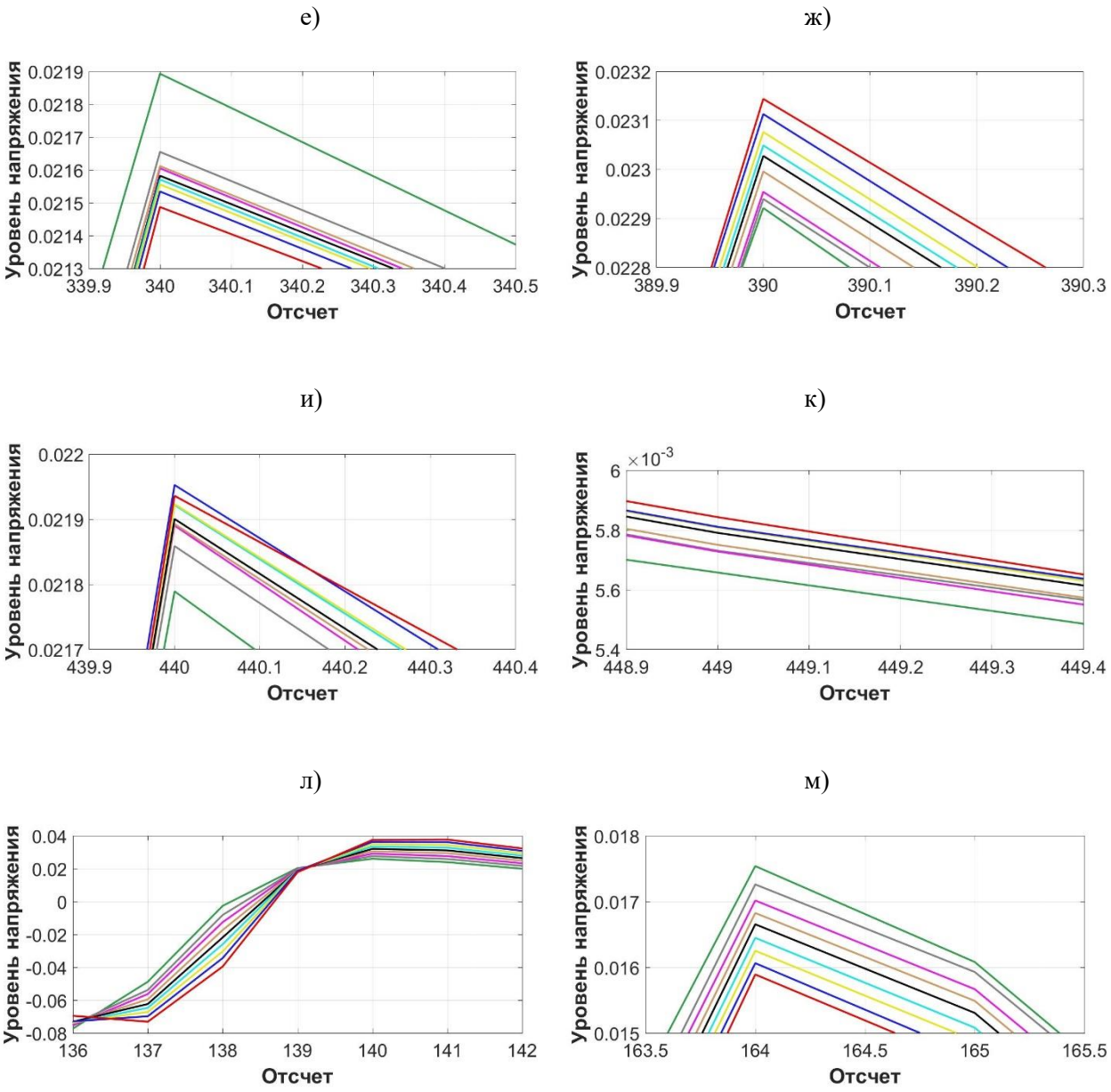
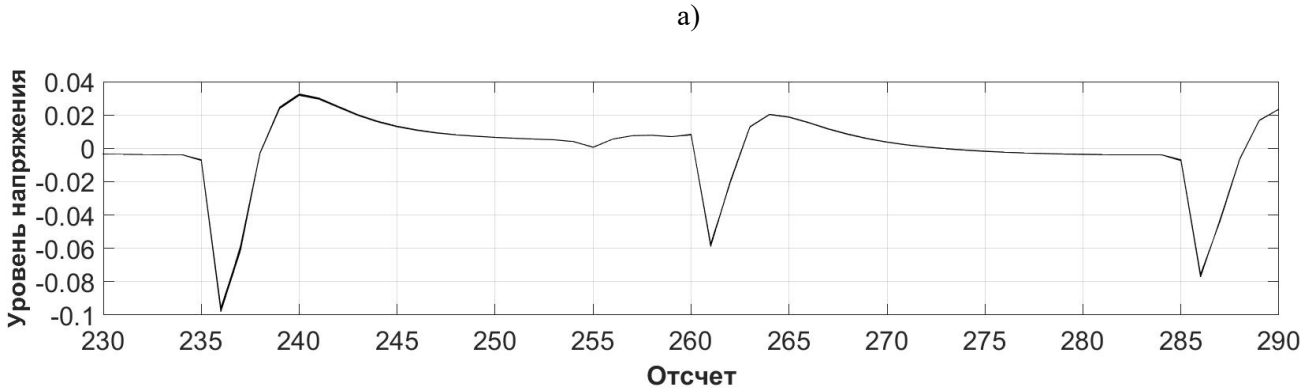


Рисунок Д.3 – Увеличенные области усреднённых форм сигнала, характеризующих зависимость энергопотребления микроконтроллера ATmega 16 от веса Хэмминга комбинации, записываемой в предварительно заполненную ячейку ОЗУ



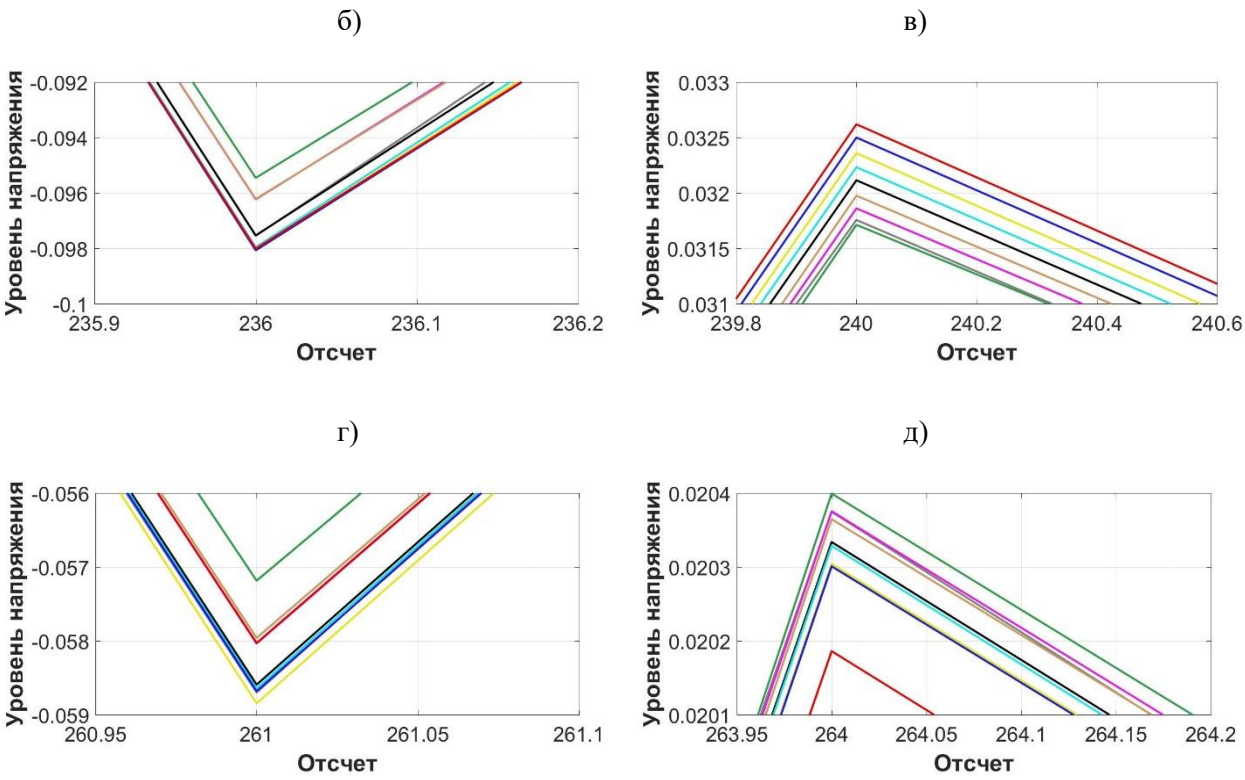
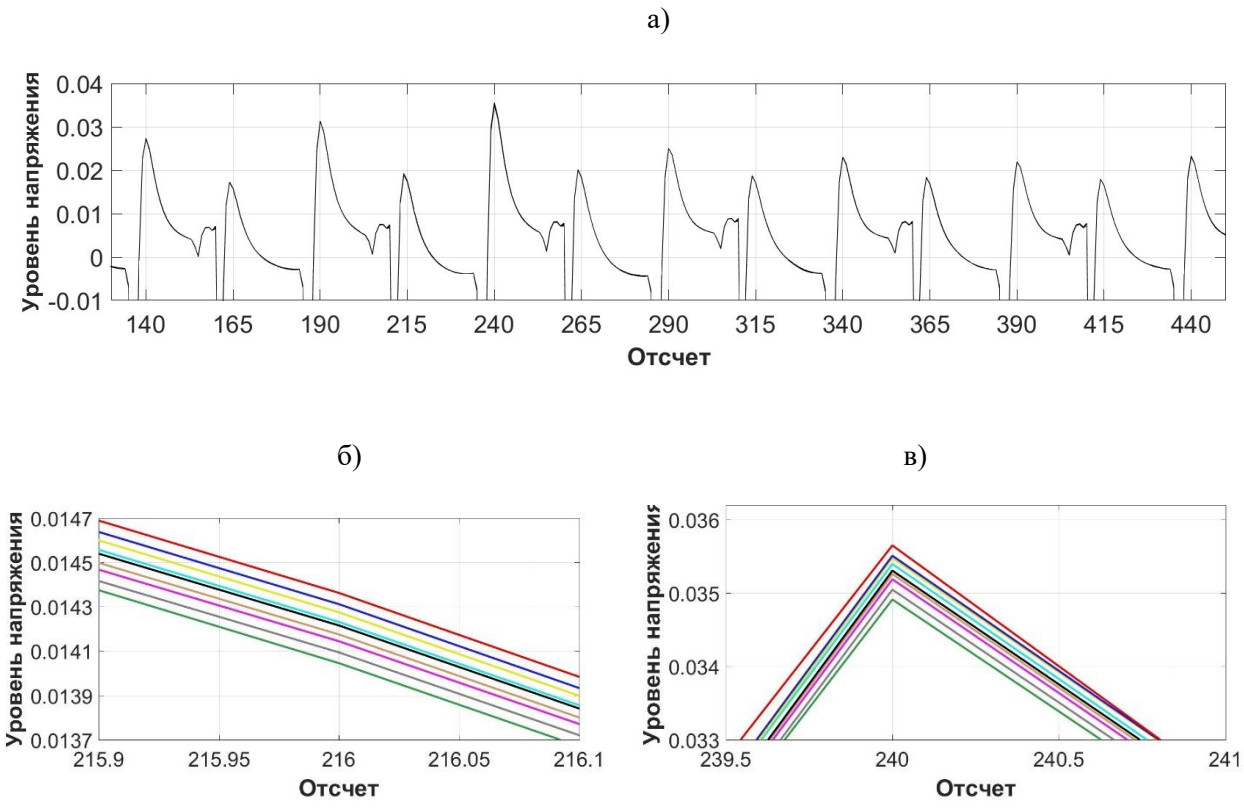
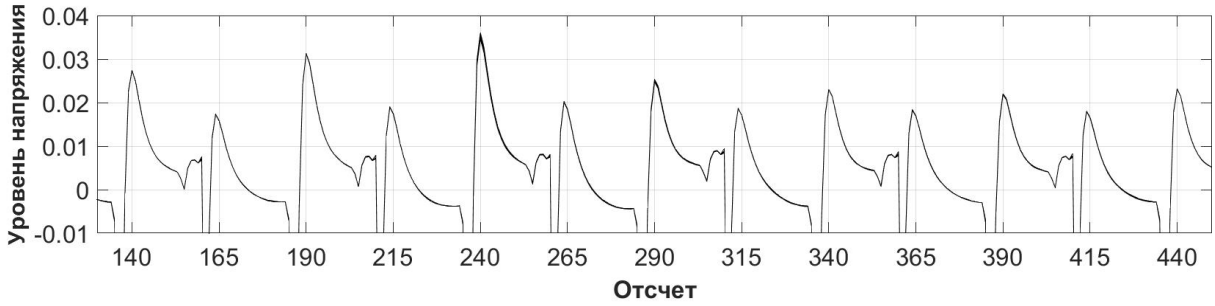


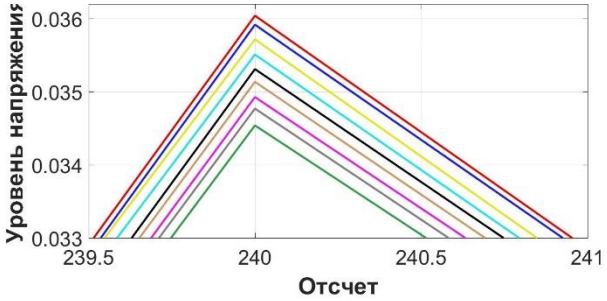
Рисунок Д.4 – Увеличенные области усреднённых форм сигнала, характеризующих зависимость энергопотребления микроконтроллера ATmega 16 от веса Хэмминга комбинации, считываемой из ячейки ОЗУ в чистую ячейку рабочего регистра



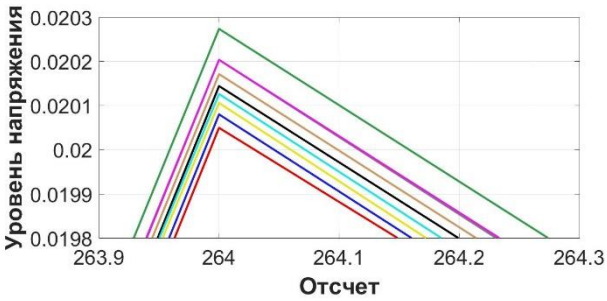
г)



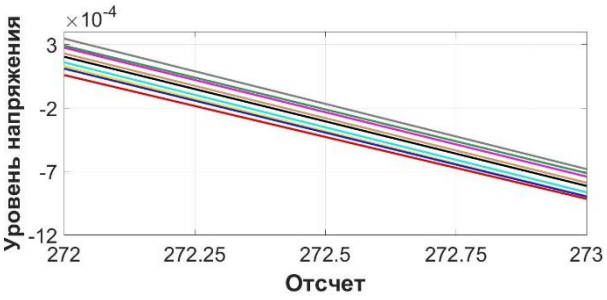
д)



е)



ж)



и)

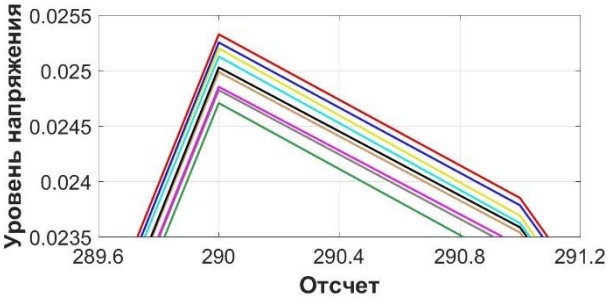
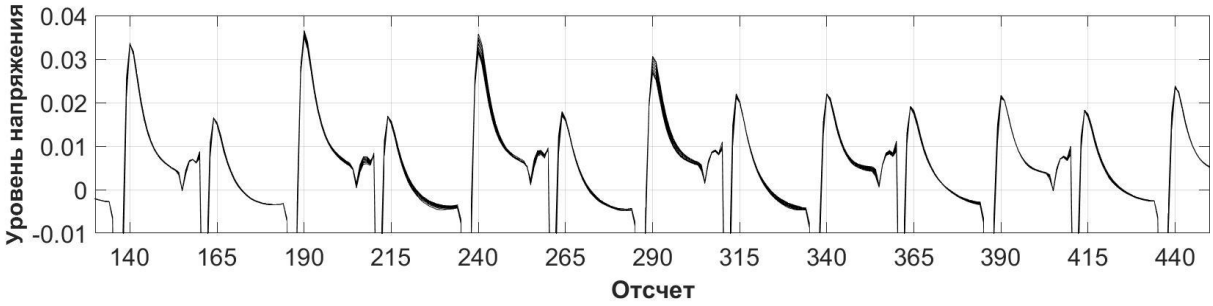
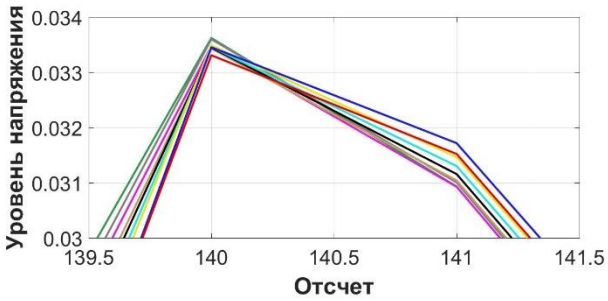


Рисунок Д.5 – Увеличенные области усреднённых форм сигнала, полученных при выполнении операции чтения из ячейки ОЗУ в предварительно заполненную ячейку рабочего регистра, и характеризующих зависимость энергопотребления микроконтроллера ATmega 16 от: а)-в) веса Хэмминга считываемой комбинации; г)-и) расстояния Хэмминга между записываемой и перезаписываемой комбинациями

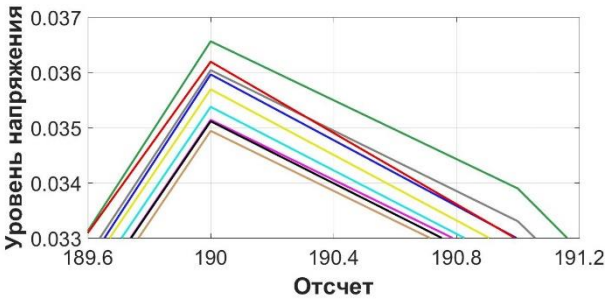
а)


Уровень напряжения
0.04
0.03
0.02
0.01
0
-0.01
140 165 190 215 240 265 290 315 340 365 390 415 440
Отсчет

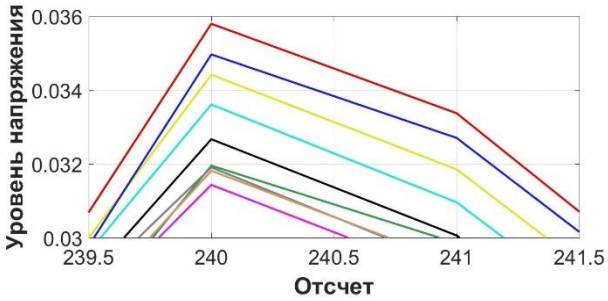
б)


Уровень напряжения
0.034
0.033
0.032
0.031
0.03
139.5 140 140.5 141 141.5
Отсчет

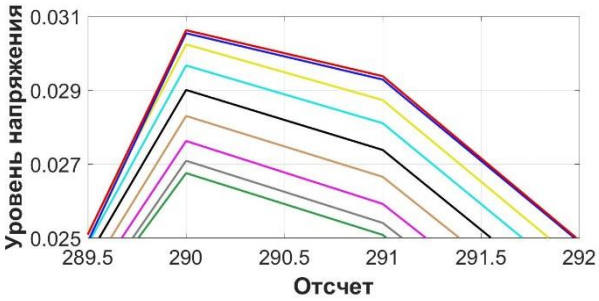
в)


Уровень напряжения
0.037
0.036
0.035
0.034
0.033
189.6 190 190.4 190.8 191.2
Отсчет

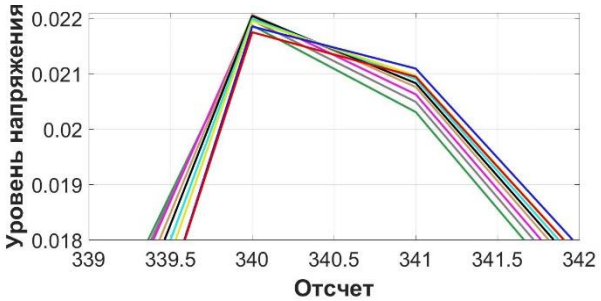
г)


Уровень напряжения
0.036
0.034
0.032
0.03
239.5 240 240.5 241 241.5
Отсчет

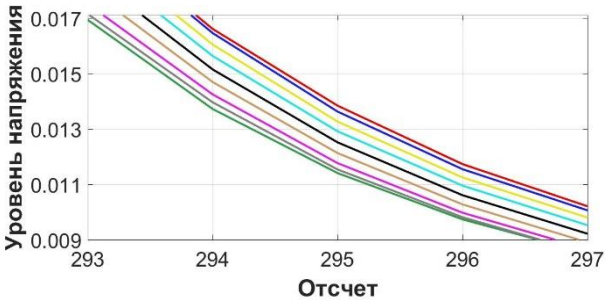
д)


Уровень напряжения
0.031
0.029
0.027
0.025
289.5 290 290.5 291 291.5 292
Отсчет

е)


Уровень напряжения
0.022
0.021
0.02
0.019
0.018
339 339.5 340 340.5 341 341.5 342
Отсчет

ж)


Уровень напряжения
0.017
0.015
0.013
0.011
0.009
293 294 295 296 297
Отсчет

и)

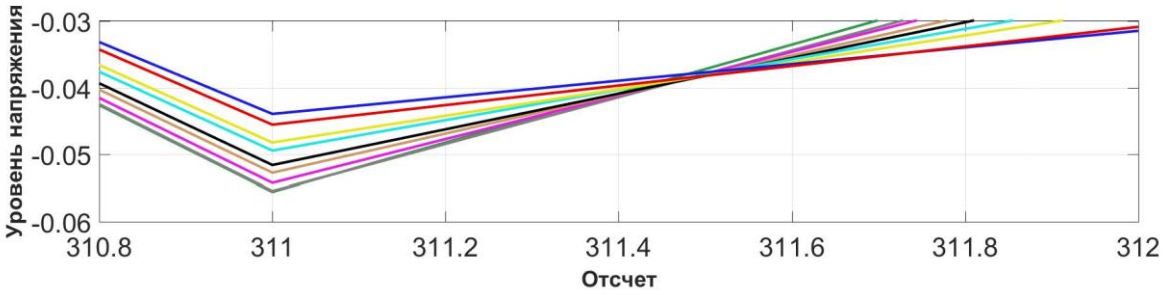
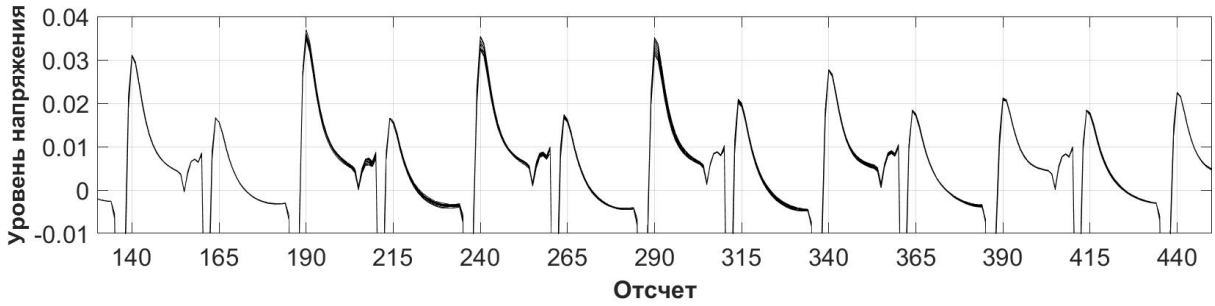
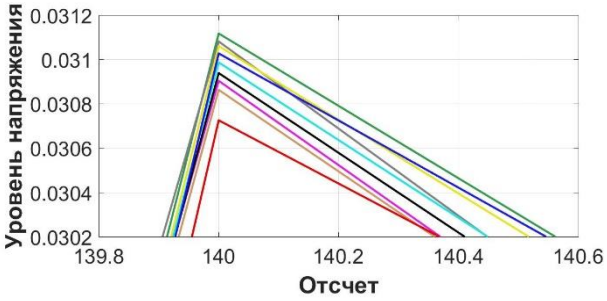


Рисунок Д.6 – Увеличенные области усреднённых форм сигнала, характеризующих зависимость энергопотребления микроконтроллера ATmega 16 от веса Хэмминга комбинации, считываемой из ячейки ПЗУ в чистую ячейку рабочего регистра

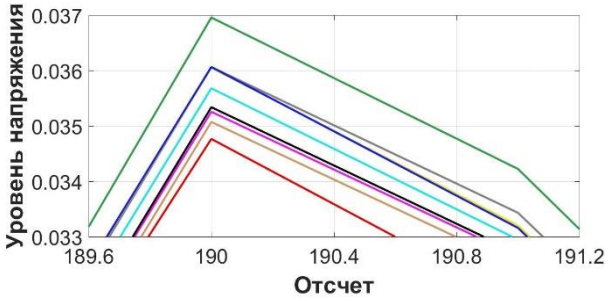
а)



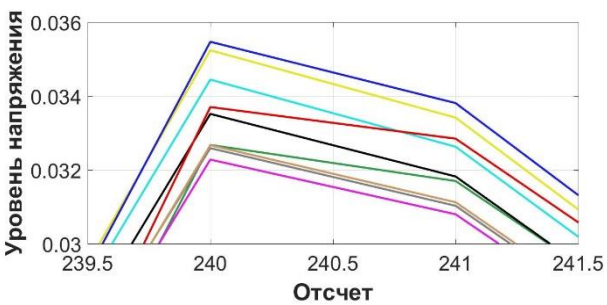
б)



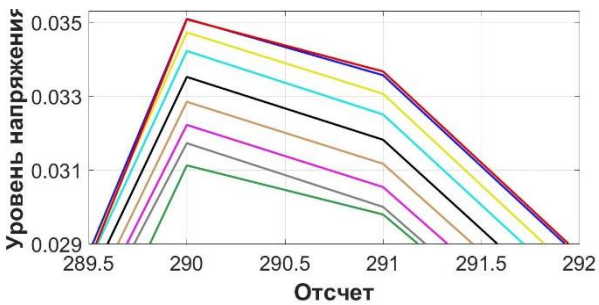
в)



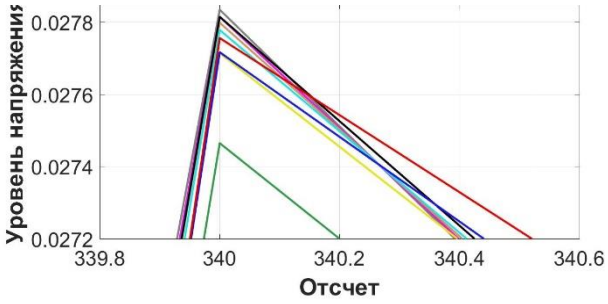
г)



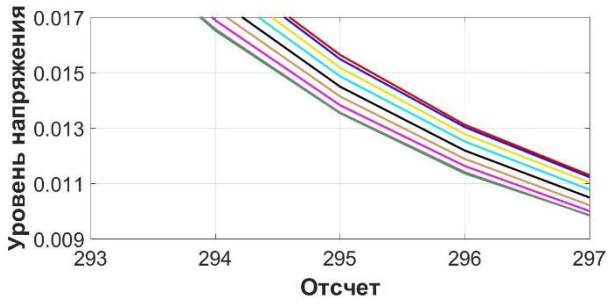
д)



е)



ж)



и)

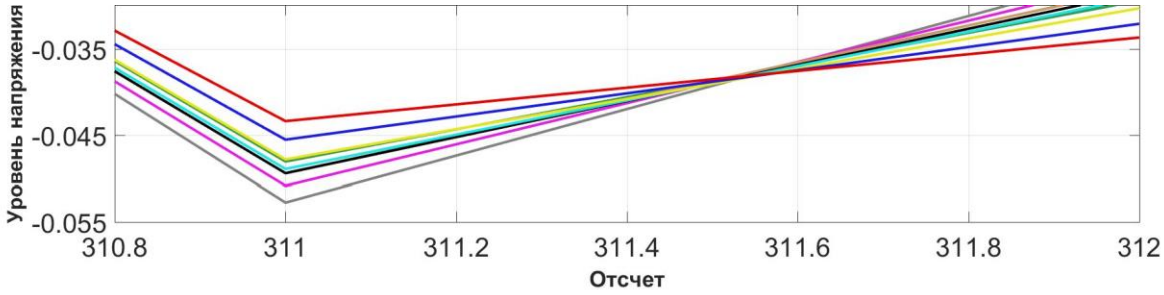
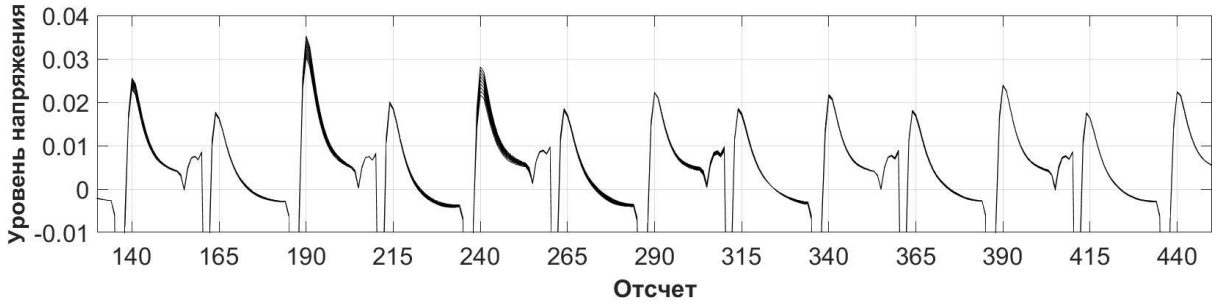
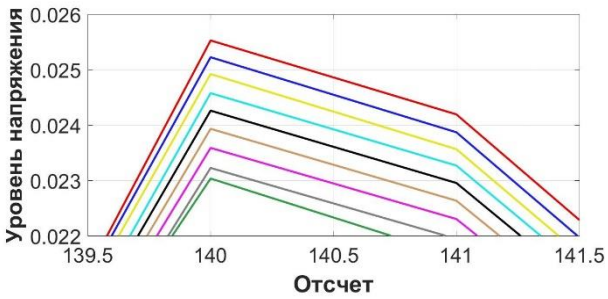


Рисунок Д.7 – Увеличенные области усреднённых форм сигнала, характеризующих зависимость энергопотребления микроконтроллера ATmega 16 от веса Хэмминга комбинации, считываемой из ячейки ПЗУ в предварительно заполненную ячейку рабочего регистра

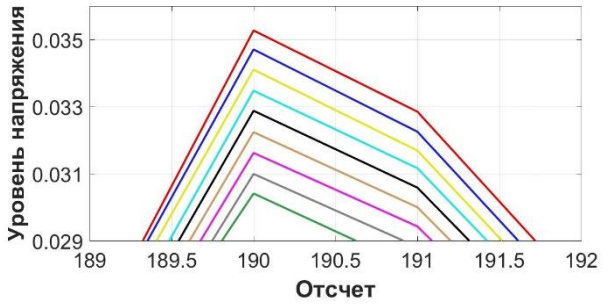
а)



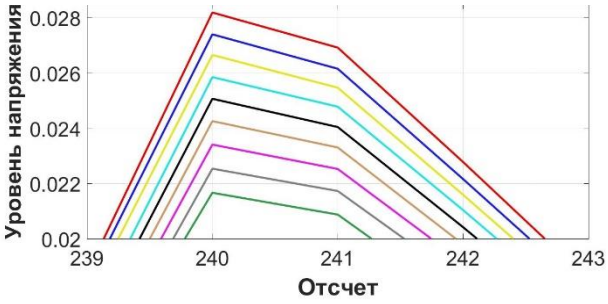
б)



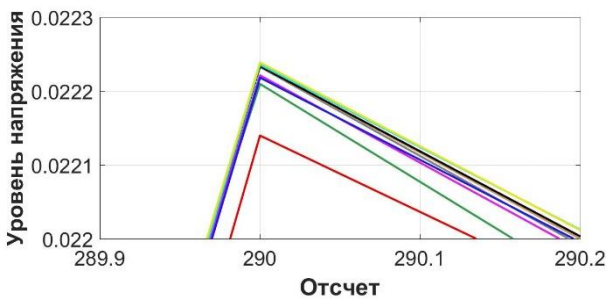
в)



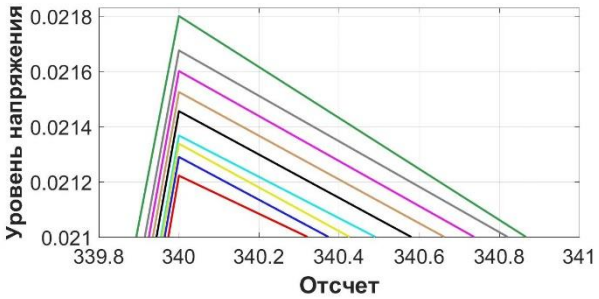
г)



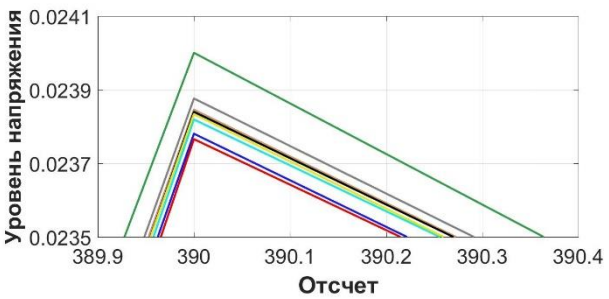
д)



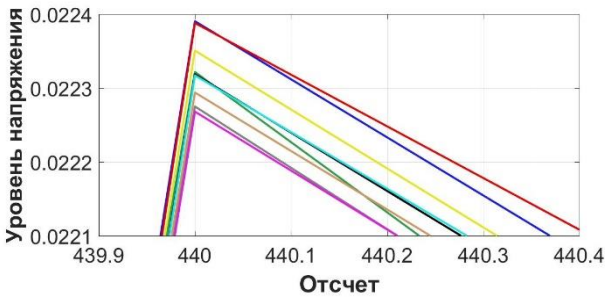
е)



ж)



и)



к)

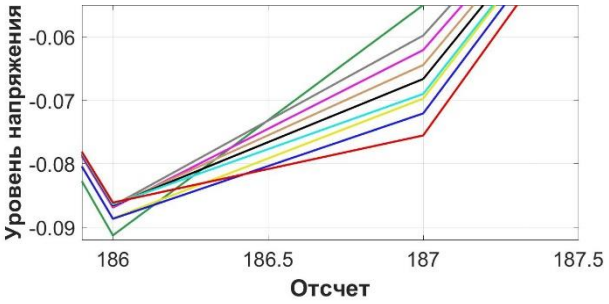
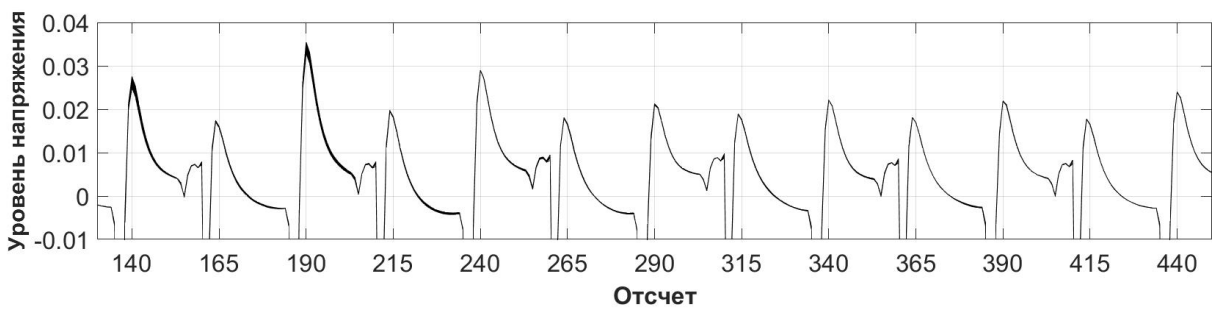
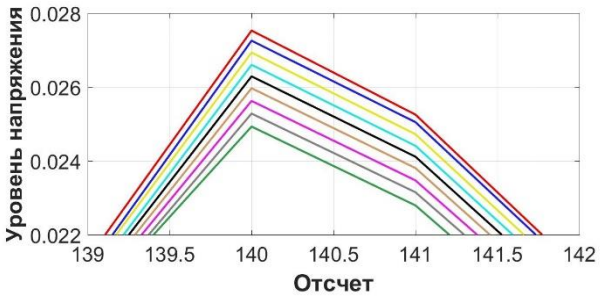


Рисунок Д.8 – Увеличенные области усреднённых форм сигнала, характеризующих зависимость энергопотребления микроконтроллера ATmega 16 от веса Хэмминга комбинации, копируемой из одного рабочего регистра в другой, предварительно очищенный регистр

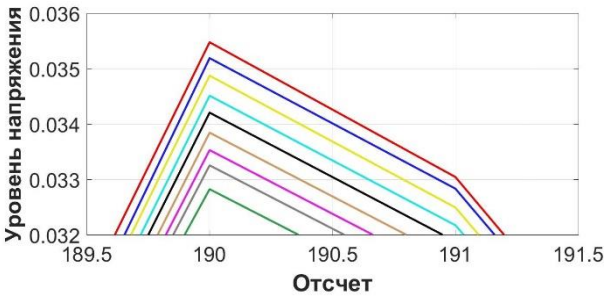
а)



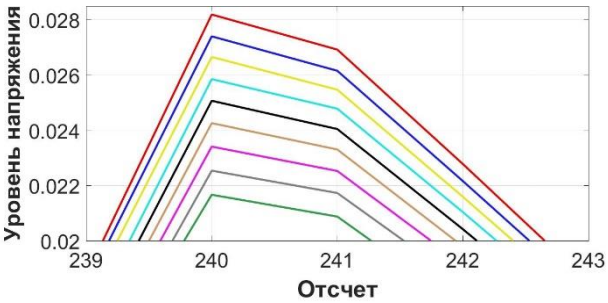
б)


Уровень напряжения
0.028
0.026
0.024
0.022
139
139.5
140
140.5
141
141.5
142
Отсчет

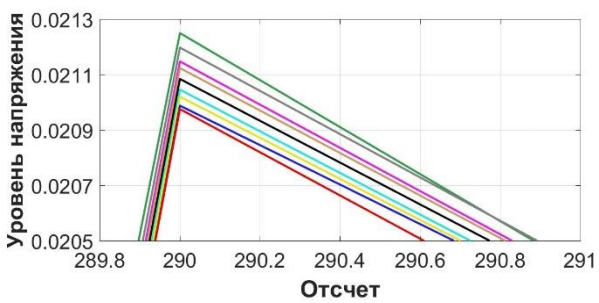
в)


Уровень напряжения
0.036
0.035
0.034
0.033
0.032
189.5
190
190.5
191
191.5
Отсчет

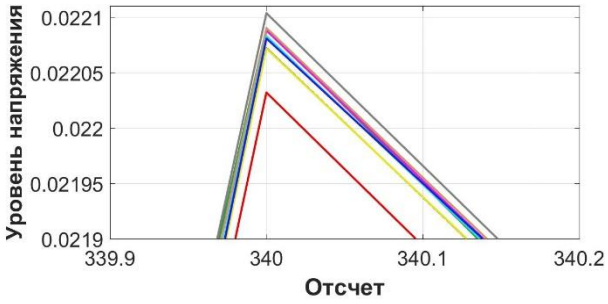
г)


Уровень напряжения
0.028
0.026
0.024
0.022
0.02
239
240
241
242
243
Отсчет

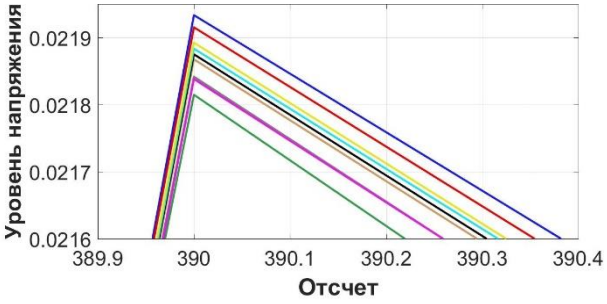
д)


Уровень напряжения
0.0213
0.0211
0.0209
0.0207
0.0205
289.8
290
290.2
290.4
290.6
290.8
291
Отсчет

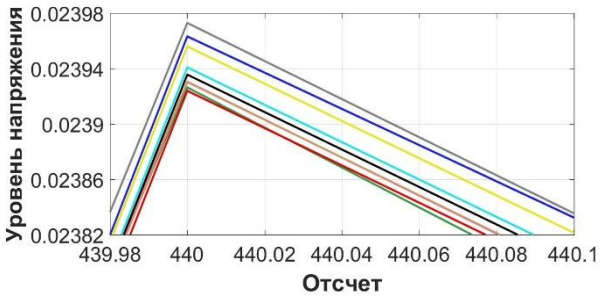
е)


Уровень напряжения
0.0221
0.02205
0.022
0.02195
0.0219
339.9
340
340.1
340.2
Отсчет

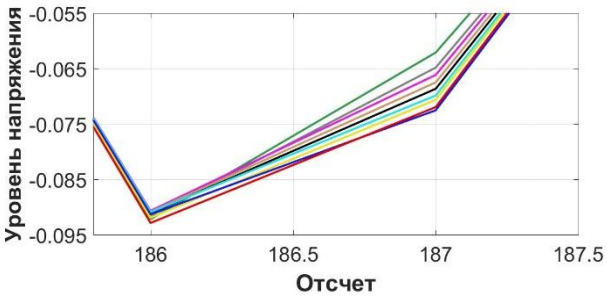
ж)


Уровень напряжения
0.0219
0.0218
0.0217
0.0216
389.9
390
390.1
390.2
390.3
390.4
Отсчет

и)


Уровень напряжения
0.02398
0.02394
0.0239
0.02386
0.02382
439.98
440
440.02
440.04
440.06
440.08
440.1
Отсчет

к)


Уровень напряжения
-0.055
-0.065
-0.075
-0.085
-0.095
186
186.5
187
187.5
Отсчет

л)

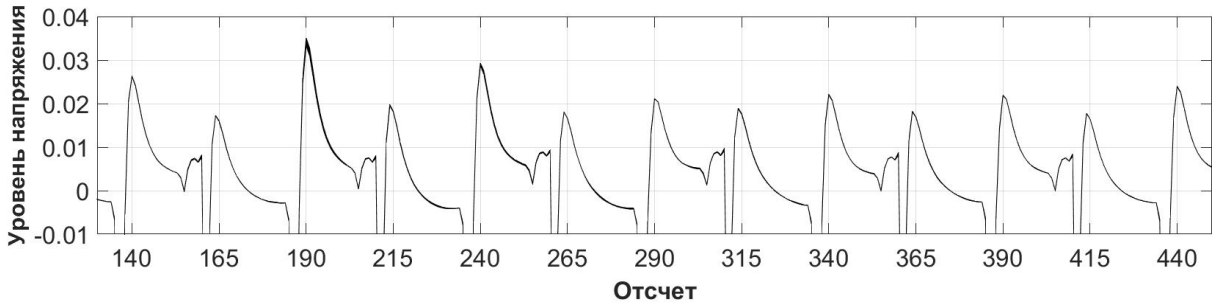


Рисунок Д.9 – Увеличенные области усреднённых форм сигнала, полученных при выполнении операции копирования данных из одного рабочего регистра в другой, предварительно заполненный регистр, и характеризующих зависимость энергопотребления микроконтроллера ATmega 16 от: а)-к) веса Хэмминга копируемой комбинации; л) расстояния Хэмминга между записываемой и перезаписываемой комбинациями

Приложение Е. Таблицы, характеризующие зависимости энергопотребления анализируемого микроконтроллера от выполняемых операций и обрабатываемых данных

Таблица Е.1 – Отсчёты с наилучшими для злоумышленника значениями $U_{\text{ДАН}}$, СКО и SNR в каждой из четырёх областей тактов, на которых существует зависимость энергопотребления от веса Хэмминга обрабатываемых данных, при выполнении операции записи в чистую ячейку ОЗУ

Такт	Область	Зависимость	Отсчёт	$U_{\text{ДАН}}$, мВ	СКО, мВ	SNR	Особые отметки
1	1	Обратная	138	7.155	26.51	0.17515	
	2	Прямая	141	2.727	2.55	2.87130	
	3	Обратная	162	0.462	25.34	0.00070	*
	3	Обратная	163	0.337	8.50	0.00380	*
	4	Обратная	172	0.438	1.62	0.17858	
2	1	Обратная	187	3.479	56.94	0.00772	
	1	Прямая	189	0.944	11.00	0.02308	*
	2	Прямая	190	1.229	2.92	0.47183	
	3	Зависимость сильно нарушена					
	4	Обратная	228	0.353	1.56	0.11022	
	4	Обратная	230	0.333	1.52	0.11265	
3	1	Обратная	237	1.782	47.20	0.00269	*
	2	Прямая	240	0.244	1.74	0.05361	
	2	Прямая	242	0.271	2.98	0.01648	
	3	Зависимость сильно нарушена					
	4	Прямая	264	0.194	1.71	0.01802	
	4	Обратная	284	0.140	1.54	0.01906	
4	1	Обратная	289	0.214	9.31	0.00095	*
	2	Обратная	290	0.118	1.97	0.01355	*
	2	Обратная	293	0.146	2.26	0.00873	*
	3	Зависимость сильно нарушена					
	4	Прямая	316	0.108	2.39	0.00400	*
	4	Прямая	323	0.092	1.64	0.00997	
5	1	Зависимость сильно нарушена					
	2	Обратная	352	0.084	1.52	0.00781	
	2	Обратная	355	0.106	2.76	0.00266	
	3	Зависимость сильно нарушена					
	4	Прямая	384	0.063	1.53	0.00387	
6	1	Зависимость сильно нарушена					
	2	Прямая	390	0.029	1.94	0.00138	*
	2	Прямая	391	0.039	2.47	0.00106	*

Таблица Е.2 – Отсчёты с наилучшими для злоумышленника значениями $U_{\text{ДАН}}$, СКО и SNR в каждой из четырёх областей тактов, на которых существует зависимость энергопотребления от веса Хэмминга обрабатываемых данных, при выполнении операции записи в предварительно заполненную ячейку ОЗУ

Такт	Область	Зависимость	Отсчёт	$U_{\text{ДАН}}$, мВ	СКО, мВ	SNR	Особые отметки
1	1	Обратная	138	7.392	26.46	0.17880	
	2	Прямая	141	2.746	2.50	3.01080	
	3	Обратная	162	0.868	25.07	0.00125	
	3	Обратная	163	0.469	8.05	0.00518	
	4	Обратная	169	0.398	1.87	0.11952	
	4	Обратная	172	0.386	1.69	0.15749	
2	1	Обратная	187	4.201	52.54	0.01091	
	1	Прямая	189	0.893	10.30	0.02391	
	2	Прямая	190	1.303	1.76	1.3356	
	3	Зависимость сильно нарушена					
	4	Обратная	225	0.327	1.63	0.09434	
	4	Обратная	228	0.290	1.62	0.10290	
3	1	Обратная	237	2.426	46.23	0.00376	
	2	Прямая	240	0.279	1.82	0.04716	
	2	Прямая	241	0.333	2.34	0.03441	
	3	Зависимость сильно нарушена					
	4	Прямая	264	0.117	1.71	0. 01693	*
	4	Прямая	266	0.172	2.57	0. 00783	
4	1	Обратная	287	1.032	36.35	0. 00066	*
	1	Обратная	289	0.329	9.21	0. 00158	
	2	Обратная	290	0.149	1.81	0. 01634	
	3	Зависимость сильно нарушена					
	4	Прямая	316	0.135	2.54	0. 00466	
	4	Прямая	322	0.113	1.73	0. 01035	
5	1	Зависимость сильно нарушена					
	2	Обратная	353	0.084	1.59	0. 00779	
	2	Обратная	355	0.141	2.88	0. 00377	*
	3	Обратная	363	0.274	8.07	0. 00095	
	4	Прямая	378	0. 087	1.60	0. 00326	
	4	Прямая	382	0.072	1.63	0. 00387	
6	1	Зависимость сильно нарушена					
	2	Прямая	391	0.084	2.19	0.00200	

Таблица Е.3 – Отсчёты с наилучшими для злоумышленника значениями $U_{\text{ДАН}}$, СКО и SNR в каждой из четырёх областей третьего такта, на котором существует зависимость энергопотребления от веса Хэмминга обрабатываемых данных, при выполнении операции чтения из ячейки ОЗУ в чистую ячейку рабочего регистра

Такт	Область	Зависимость	Отсчёт	$U_{\text{ДАН}}$, мВ	СКО, мВ	SNR	Особые отметки
3	1	Обратная	237	0.328	52.56	0.00023	*
	1	Прямая	239	0.189	10.07	0.00128	*
	2	Прямая	240	0.182	1.43	0.04962	
	3	Зависимость сильно нарушена					
	4	Обратная	268	0.055	1.98	0. 00117	*
	4	Обратная	272	0.029	1.41	0. 00170	*

Таблица Е.4 – Отсчёты с наилучшими для злоумышленника значениями $U_{\text{ДАН}}$, СКО и SNR в каждой из четырёх областей тактов, на которых существует зависимость энергопотребления от веса Хэмминга обрабатываемых данных, при выполнении операции чтения из ячейки ПЗУ в чистую ячейку рабочего регистра

Такт	Область	Зависимость	Отсчёт	$U_{\text{ДАН}}$, мВ	СКО, мВ	SNR	Особые отметки
1	1	Обратная	137	2.823	44.67	0.01439	*
	1	Обратная	138	1.789	26.42	0.01547	*
	2	Прямая	143	0.138	3.249	0.01226	*
	2	Прямая	160	0.015	11.32	0.00118	
	3	Обратная	162	1.924	27.16	0.01602	*
	3	Обратная	163	0.651	8.94	0.01608	*
	4	Прямая	167	0.156	2.33	0.00896	*
2	1	Обратная	188	1.101	27.32	0.00572	*
	2	Обратная	206	0.344	2.46	0.04665	
	2	Обратная	208	0.328	1.607	0.10739	
	3	Обратная	212	1.990	27.16	0.01788	*
	3	Обратная	213	0.672	8.93	0.01853	*
	4	Прямая	220	0.268	1.91	0.02829	
	4	Обратная	225	0.248	1.71	0.03715	
3	1	Обратная	237	5.749	58.97	0.03718	
	2	Обратная	256	0.365	11.30	0.00014	
	2	Обратная	257	0.242	1.74	0.04753	
	3	Обратная	263	0.835	8.96	0.02883	
	4	Обратная	264	0.187	1.95	0.04585	*

Продолжение таблицы Е.4

Такт	Область	Зависимость	Отсчёт	$U_{ДАН}$, мВ	СКО, мВ	SNR	Особые отметки
4	1	Обратная	287	4.869	51.85	0.03268	*
	2	Прямая	290	0.774	2.48	0.41989	
	2	Прямая	291	0.860	3.17	0.30085	
	3	Прямая	311	2.00	57.54	0.00553	
	3	Обратная	313	0.875	10.66	0.02090	*
	4	Обратная	327	0.203	1.63	0.05190	
5	1	Обратная	339	0.628	9.59	0.01572	*
	2	Прямая	354	0.340	2.59	0.04029	
	2	Прямая	360	0.459	11.38	0.00316	
	3	Обратная	362	1.81	27.06	0.01403	*
	4	Прямая	365	0.181	2.13	0.02663	*
6	1	Обратная	387	2.486	36.38	0.01576	*
	1	Обратная	388	0.998	10.94	0.02309	
	2	Обратная	390	0.174	2.02	0.02312	*
	2	Прямая	410	0.405	11.34	0.00171	
	3	Обратная	412	1.826	26.99	0.01514	*
	4	Прямая	417	0.204	2.244	0.02878	*

Таблица Е.5 – Отсчёты с наилучшими для злоумышленника значениями $U_{ДАН}$, СКО и SNR в каждой из четырёх областей тактов, на которых существует зависимость энергопотребления от веса Хэмминга обрабатываемых данных, при выполнении операции чтения из ячейки ПЗУ в предварительно заполненную ячейку рабочего регистра

Такт	Область	Зависимость	Отсчёт	$U_{ДАН}$, мВ	СКО, мВ	SNR	Особые отметки
1	1	Обратная	137	0.579	38.65	0.00393	*
	1	Обратная	138	0.365	25.35	0.00486	*
	2	Зависимость сильно нарушена					
	3	Прямая	161	0.945	52.30	0.00125	*
	3	Обратная	162	0.575	27.97	0.00372	*
	4	Прямая	167	0.054	2.37	0.00145	*
2	1	Зависимость сильно нарушена					
	2	Прямая	208	0.3221	1.68	0.10804	
	3	Обратная	211	0.843	52.37	0.00098	*
	3	Обратная	213	0.191	9.23	0.00449	*
	4	Прямая	224	0.233	1.77	0.02050	
	4	Прямая	225	0.231	1.72	0.02329	

Продолжение таблицы Е.5

Такт	Область	Зависимость	Отсчёт	$U_{ДАН}$, мВ	СКО, мВ	SNR	Особые отметки
3	1	Обратная	237	1.727	55.58	0.01788	*
	2	Обратная	256	0.243	2.82	0.02089	*
	2	Обратная	257	0.207	1.77	0.02866	
	3	Обратная	262	0.819	28.03	0.00553	*
	3	Обратная	263	0.351	9.17	0.00930	*
	4	Обратная	264	0.161	1.93	0.03955	*
4	1	Обратная	287	2.843	60.43	0.01203	*
	2	Прямая	290	0.791	3.94	0.15812	
	3	Прямая	311	0.937	63.22	0.00122	*
	3	Обратная	313	0.382	11.04	0.00516	*
	4	Обратная	314	0.202	2.14	0.01226	
	4	Обратная	326	0.170	1.79	0.03936	*
5	1	Обратная	337	1.095	50.37	0.00350	*
	1	Обратная	338	0.328	17.60	0.00484	*
	2	Прямая	355	0.1957	2.61	0.00833	
	2	Прямая	359	0.142	1.77	0.02821	*
	3	Прямая	361	1.021	52.17	0.00156	*
	3	Обратная	362	0.504	27.86	0.00284	*
	4	Обратная	384	0.126	1.68	0.01431	
6	1	Обратная	387	0.881	37.34	0.00386	*
	1	Обратная	388	0.332	10.31	0.00770	*
	2	Обратная	390	0.111	2.13	0.01383	*
	3	Прямая	411	0.935	51.89	0.00135	*
	3	Обратная	412	0.559	27.86	0.00334	*
	4	Прямая	420	0.096	1.83	0.01259	*
	4	Прямая	417	0.119	2.35	0.01131	*

Таблица Е.6 – Отсчёты с наилучшими для злоумышленника значениями $U_{\text{ДАН}}$, СКО и SNR в каждой из четырёх областей тактов, на которых существует зависимость энергопотребления от веса Хэмминга обрабатываемых данных, при выполнении операции копирования данных между рабочими регистрами

Такт	Область	Зависимость	Отсчёт	$U_{\text{ДАН}}$, мВ	СКО, мВ	SNR	Особые отметки
1	1	Обратная	137	2.468	36.84	0.00714	
	2	Прямая	140	0.498	1.690	0.24612	
	2	Прямая	141	0.513	2.12	0.14572	
	3	Обратная	163	0.263	8.36	0.00067	
	4	Обратная	164	0.105	1.94	0.00331	
2	1	Обратная	187	4.11	53.42	0.00995	
	1	Прямая	189	0.574	10.44	0.01265	
	2	Прямая	190	0.973	1.60	0.96911	
	3	Обратная	213	0.308	10.11	0.00079	
	4	Обратная	222	0.173	1.67	0.02390	
	4	Обратная	225	0.158	1.540	0.02998	
3	1	Обратная	237	4.251	40.23	0.02112	
	1	Прямая	239	0.930	9.95	0.02761	
	2	Прямая	240	1.306	2.03	1.03140	
	3	Обратная	263	0.267	8.32	0. 00090	
	4	Обратная	273	0.205	1.60	0. 04725	
	4	Обратная	277	0.209	1.55	0. 04315	
4	1	Обратная	288	0.354	13.43	0. 00051	*
	2	Прямая	308	0.159	1.53	0. 02707	
	2	Прямая	310	0.205	11.32	0. 00047	
	3	Зависимость сильно нарушена					
	4	Обратная	334	0.108	1.52	0. 01298	
	4	Обратная	335	0.176	11.92	0. 00016	
5	1	Обратная	338	0.302	9.70	0.00115	
	2	Обратная	340	0.116	1.88	0.00878	
	3	Зависимость сильно нарушена					
	4	Прямая	366	0.119	2.60	0.00307	
	4	Прямая	368	0.099	1.93	0.00623	
6	1	Зависимость сильно нарушена					
	2	Обратная	400	0.063	1.55	0.00386	
	2	Обратная	405	0.120	2.67	0.00202	
	3	Зависимость сильно нарушена					
	4	Обратная	414	0.057	1.94	0.00073	

Таблица Е.7 – Отсчёты с наилучшими для злоумышленника значениями $U_{\text{ДАН}}$, СКО и SNR в каждой из четырёх областей тактов, на которых существует зависимость энергопотребления от расстояния Хэмминга между записываемой и перезаписываемой комбинациями, при выполнении операции копирования данных из одного рабочего регистра в другой, предварительно заполненный регистр

Такт	Область	Зависимость	Отсчёт	$U_{\text{ДАН}}$, мВ	СКО, мВ	SNR	Особые отметки
1	1	Зависимость сильно нарушена					
	2	Зависимость сильно нарушена					
	3	Зависимость сильно нарушена					
	4	Зависимость сильно нарушена					
2	1	Обратная	187	1.474	58.27	0.00174	
	2	Прямая	190	0.381	3.69	0.02801	
	3	Зависимость сильно нарушена					
	4	Обратная	214	0.077	1.83	0.00227	
3	1	Зависимость сильно нарушена					
	2	Прямая	240	0.134	3.02	0.00486	
	2	Прямая	241	0.163	3.44	0.00429	
	3	Зависимость сильно нарушена					
	4	Обратная	275	0.057	1.72	0.00104	
	4	Обратная	283	0.038	1.69	0.00189	*
4	1	Зависимость сильно нарушена					
	2	Прямая	302	0.049	1.68	0.00081	
	2	Прямая	309	0.048	1.78	0.00100	
	3	Зависимость сильно нарушена					
	4	Прямая	314	0.032	1.75	0.00118	
	4	Прямая	315	0.051	2.26	0.00108	
5	1	Зависимость сильно нарушена					
	2	Обратная	340	0.044	1.89	0.00030	
	2	Обратная	353	0.028	1.64	0.00048	
	3	Зависимость сильно нарушена					
	4	Прямая	369	0.045	2.03	0.00055	*
	4	Прямая	370	0.040	1.92	0.00062	*
6	Зависимость сильно нарушена						

Таблица Е.8 – Отсчёты с наилучшими для злоумышленника значениями $U_{\text{ДАН}}$, СКО и SNR в каждой из четырёх областей тактов, на которых существует зависимость энергопотребления от веса Хэмминга обрабатываемых данных, при выполнении операции копирования данных из одного рабочего регистра в другой, предварительно заполненный регистр

Такт	Область	Зависимость	Отсчёт	$U_{\text{ДАН}}$, мВ	СКО, мВ	SNR	Особые отметки
1	1	Обратная	137	2.071	39.96	0.00648	
	2	Прямая	140	0.52	1.96	0.17938	
	3	Обратная	163	0.148	8.19	0.00082	*
	4	Обратная	170	0.091	1.80	0.00479	
	4	Обратная	172	0.080	1.73	0.00497	
2	1	Прямая	189	0.350	11.49	0.00245	
	2	Прямая	190	0.531	3.65	0.05374	
	3	Обратная	213	0.137	9.94	0.00049	*
	4	Обратная	224	0.088	1.79	0.00895	
	4	Обратная	225	0.091	1.75	0.00859	
3	1	Обратная	239	0.118	10.46	0.00031	*
	2	Прямая	253	0.076	1.72	0.00509	
	2	Прямая	254	0.11	3.14	0.00244	
	3	Зависимость сильно нарушена					
	4	Прямая	266	0.084	2.66	0.00148	
	4	Обратная	284	0.066	1.67	0.00211	
4	1	Обратная	289	0.151	9.38	0.00056	
	2	Обратная	290	0.055	1.92	0.00213	
	3	Зависимость сильно нарушена					
	4	Прямая	322	0.054	1.79	0.00185	
5	1	Зависимость сильно нарушена					
	2	Обратная	356	0.055	2.64	0.00080	*
	2	Обратная	357	0.026	1.77	0.00094	*
	3	Зависимость сильно нарушена					
	4	Прямая	380	0.047	1.66	0.00098	
	4	Прямая	385	0.144	13.86	0.00024	
6	Зависимость сильно нарушена						

Таблица Е.9 – Отсчёты с наилучшими для злоумышленника значениями $U_{\text{ДАН}}$, СКО и SNR в каждой из четырёх областей тактов, на которых существует зависимость энергопотребления от веса Хэмминга первого слагаемого операции XOR

Такт	Область	Зависимость	Отсчёт	$U_{\text{ДАН}}$, мВ	СКО, мВ	SNR	Особые отметки
1	1	Обратная	137	2.283	35.55	0.01700	
	2	Прямая	140	0.687	2.43	0.21649	
	3	Зависимость сильно нарушена					
	4	Обратная	166	0.119	2.81	0.00158	
	4	Обратная	173	0.072	2.02	0.00707	*
2	1	Обратная	187	4.204	50.31	0.02470	
	1	Прямая	189	1.615	8.87	0.07123	
	2	Прямая	190	1.812	2.71	1.12450	
	3	Зависимость сильно нарушена					
	4	Обратная	222	0.320	2.11	0.05304	
3	1	Обратная	237	2.559	39.46	0.01640	
	1	Прямая	239	1.190	8.21	0.04112	
	2	Прямая	240	1.289	2.39	0.71992	
	3	Зависимость сильно нарушена					
	4	Обратная	278	0.281	2.00	0.04574	
	4	Обратная	279	0.285	1.98	0.04438	
4	1	Зависимость сильно нарушена					
	2	Прямая	308	0.217	1.98	0.02562	
	2	Прямая	310	0.234	14.62	0.00156	
	3	Зависимость сильно нарушена					
	4	Прямая	314	0.12	2.12	0.01126	*
5	1	Зависимость сильно нарушена					
	2	Обратная	340	0.124	2.12	0.01182	*
	3	Зависимость сильно нарушена					
	4	Прямая	374	0.120	1.99	0.00817	*
6	1	Зависимость сильно нарушена					
	2	Обратная	406	0.105	2.72	0.00563	
	3	Зависимость сильно нарушена					
	4	Прямая	431	0.076	1.96	0.00249	
	4	Прямая	434	0.069	1.96	0.00258	

Таблица Е.10 – Отсчёты с наилучшими для злоумышленника значениями $U_{\text{ДАН}}$, СКО и SNR в каждой из четырёх областей тактов, на которых существует зависимость энергопотребления от веса Хэмминга второго слагаемого операции XOR

Такт	Область	Зависимость	Отсчёт	$U_{\text{ДАН}}$, мВ	СКО, мВ	SNR	Особые отметки
1	1	Обратная	137	1.866	36.08	0.00684	
	1	Прямая	139	0.529	7.04	0.01464	
	2	Прямая	140	0.631	2.50	0.16584	
	3	Обратная	162	0.217	21.13	0.00038	*
	3	Обратная	163	0.116	6.69	0.00106	*
	4	Обратная	170	0.116	2.07	0.00596	*
	4	Обратная	172	0.103	2.04	0.00613	*
2	1	Обратная	187	1.686	51.49	0.00284	
	1	Прямая	189	0.559	9.22	0.00930	
	2	Прямая	190	0.654	3.83	0.07402	
	3	Зависимость сильно нарушена					
	4	Обратная	224	0.129	2.09	0.00954	
	4	Обратная	225	0.122	2.03	0.01006	
3	1	Обратная	238	0.152	18.89	0.00018	*
	2	Прямая	252	0.096	2.01	0.00625	
	2	Прямая	254	0.109	3.36	0.00254	
	3	Зависимость сильно нарушена					
	4	Обратная	283	0.072	1.99	0.00277	
4	1	Обратная	289	0.093	7.54	0.00065	*
	2	Обратная	290	0.064	2.07	0.00286	*
	3	Зависимость сильно нарушена					
	4	Прямая	315	0.085	2.49	0.00150	*
	4	Прямая	319	0.053	2.14	0.00218	*
5	1	Зависимость сильно нарушена					
	2	Обратная	353	0.046	1.95	0.00079	*
	2	Обратная	355	0.057	3.30	0.00059	*
	3	Зависимость сильно нарушена					
	4	Прямая	384	0.038	1.99	0.00046	*
6	1	Зависимость сильно нарушена					
	2	Прямая	391	0.027	2.68	0.00025	*
	2	Прямая	392	0.040	2.92	0.00023	*

Таблица Е.11 – Отсчёты с наилучшими для злоумышленника значениями $U_{\text{ДАН}}$, СКО и SNR в каждой из четырёх областей тактов, на которых существует зависимость энергопотребления от веса Хэмминга результата сложения по mod2 содержимого двух регистров

Такт	Область	Зависимость	Отсчёт	$U_{\text{ДАН}}$, мВ	СКО, мВ	SNR	Особые отметки
1	Зависимость сильно нарушена						
2	1	Прямая	189	0.146	9.14	0.00080	
	2	Прямая	190	0.179	3.95	0.00395	
	3	Зависимость сильно нарушена					
	4	Обратная	215	0.040	2.55	0.00054	*
	4	Обратная	216	0.049	2.91	0.00051	*

Приложение Ж. Реализация табличного преобразования на микроконтроллерах Atmel с использованием подхода «чтение со смещением»

Ниже будут рассмотрены принципы реализации табличного нелинейного преобразования на микроконтроллере Atmel на примере преобразования на S-box шифра «Магма». Восьми разрядная архитектура чипа предполагает, что в любых преобразованиях чип оперирует как минимум байтами (имея соответственно восьми разрядные шины, регистры, АЛУ). Поступающий для шифрования «Магмой» блок сообщения $m(i)$ разбивается на восьми битовые части и сохраняется в регистрах или ОЗУ (занимая восемь байт), секретный ключ логично изначально преобразовать в раундовые $k(r)$, где r – номер раунда, и хранить в ПЗУ (вместо того, чтобы для каждой итерации шифрования вырабатывать заново). Для того, чтобы сложить правую часть сообщения с раундовым ключом, байт сообщения и ключа необходимо считать в регистры, после чего выполнить операцию арифметического сложения этих двух регистров, при этом результат будет записан в один из них (перезаписывая одно из слагаемых). Затем результат может быть скопирован в ОЗУ, заменяя к примеру соответствующий байт сообщения $m(i)$. Эта операция повторяется для остальных трёх восьми битовых блоков сообщения и раундового ключа. Аналогично производится операция сложения по mod2 в шифре «Кузнечик».

Таким образом, операция сложения по $\mathrm{mod}2^{32}$: $m(i)+k(r)$ шифра «Магма» будет осуществляться побайтно – от младших байт к старшим (что позволит учесть перенос бит в соседний разряд). После этого полученные четыре восьми битовых блока должны быть преобразованы на восьми S-box. Очевидно, что преобразования на S-box эффективнее реализовывать побайтно, но вначале необходимо пояснить как эти преобразования реализуются на практике. Сначала опишем реализацию преобразований на S-box, предполагающую обработку каждого S-box «Магмы» в отдельности, для того, чтобы проиллюстрировать неэффективность четырёх битовых преобразований на восьми битовой архитектуре чипа. Любое табличное преобразование на типовом ассемблере реализуется с использованием подхода, называемого «чтение со смещением». При программировании чипа, в ячейки его ПЗУ (под ПЗУ понимается Flash или EEPROM память – их использование обусловлено тем, что записанные в них данные сохраняются даже после отключения питания от чипа) последовательно друг за другом записываются все выходные значения S-box начиная со значения, получаемого в случае прихода на вход комбинации 0: «0000», затем 1: «0001», и так далее, вплоть до выходного значения, получаемого по приходу комбинации 15: «1111», при этом адрес первой ячейки заносится в память программ, данная операция повторяется для всех восьми S-box. При этом каждое

выходное значение S-box записывается в виде байта, четыре старших бита которого будут нулевыми, значащими являются лишь четыре младших бита (рисунок Ж.1).

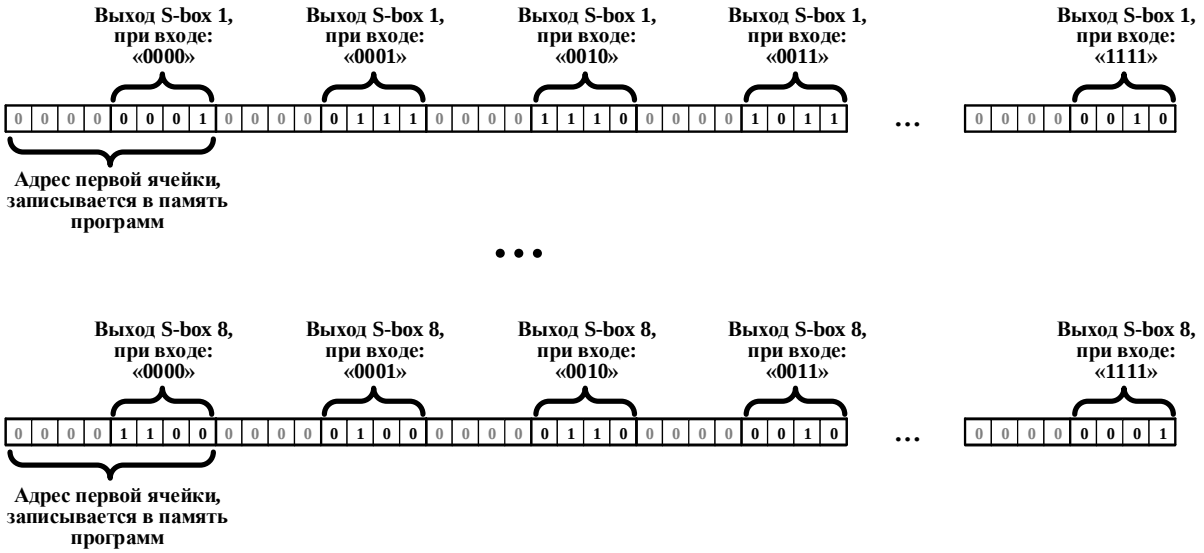


Рисунок Ж.1 – Процесс записи выходов S-box в ПЗУ

Затем четыре (вначале младших) бита 32-ух битовой последовательности $m(i)+k(r)$ необходимо сложить с *адресом* первой ячейки восьмого S-box, и по полученному адресу (то есть по адресу первой ячейки восьмого S-box со смещением на величину входного значения), произвести чтение из ячейки ПЗУ. Её содержимое очевидно окажется выходом восьмого S-box, соответствующим его входному значению. Приведём пример, положим, что адресом первой ячейки восьмого S-box (рисунок Ж.1) является: «0001 1111», тогда если четыре младших бита суммы $m(i)+k(r)$ будут равны: «0000», очевидно, что произойдёт чтение из первой ячейки и считается комбинация: «0000 1100». Если четыре младших бита суммы $m(i)+k(r)$ будут равны: «0001», то произойдёт чтение из следующей ячейки (с адресом: $0001\,1111+0001=0010\;0000$), будет считана комбинация: «0000 0100» и так далее. Описанную операцию необходимо повторить семь раз для оставшихся S-box. Из приведённого примера сразу видна слабость описанной реализации табличного преобразования – очевидно, что восьми разрядные шины, регистры и АЛУ чипа будут использоваться не эффективно (на половину пропускной способности). К тому же в большинстве ассемблеров возникнут сложности с вычленением четырёх бит из суммы $m(i)+k(r)$, записанной в четырёх регистрах (или ячейках ОЗУ) побайтно (в ассемблерах восьми разрядных чипов обычно отсутствуют команды, позволяющие оперировать четырьмя битами), в этом случае четыре бита суммы $m(i)+k(r)$ будет необходимо

вырезать побитово (побитовые операции как правило присутствуют), в некоторый промежуточный регистр, который затем складывать с адресом ячейки S-box. На выполнение этих операций уйдёт значительное время, что абсолютно неразумно (рисунок Ж.2). Аналогичные проблемы возникнут с выходами S-box, считанными из ПЗУ, четыре старших бита которых, не будут нести никакой информации – окажутся нулевыми. Конечно существуют и другие подходы к преобразованию на четырёх разрядных S-box, но их эффективность окажется не лучше.

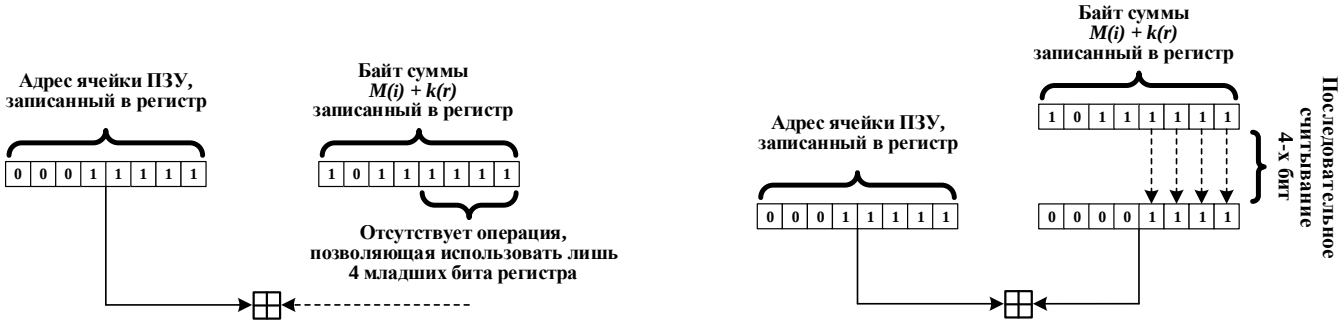


Рисунок Ж.2 – Процесс сложения адреса первой ячейки S-box с комбинацией, представляющей его вход

Учитывая описанные недостатки четырёх разрядных преобразований, реализацию S-box «Магмы» очевидно логичнее осуществлять побайтно с использованием сдвоенных S-box. Данная методика предполагает, что в ПЗУ будут записываться выходы пары соседних S-box: S-box 1-2, S-box 3-4, S-box 5-6, S-box 7-8 для каждого из 256-ти возможных вариантов входа (рисунок Ж.3).

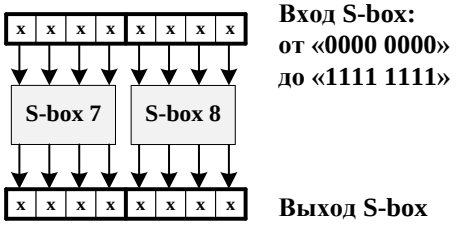


Рисунок Ж.3 – Демонстрация работы сдвоенных S-box «Магмы»

В итоге каждый, записанный в ПЗУ, байт будет использован полностью (рисунок Ж.4).

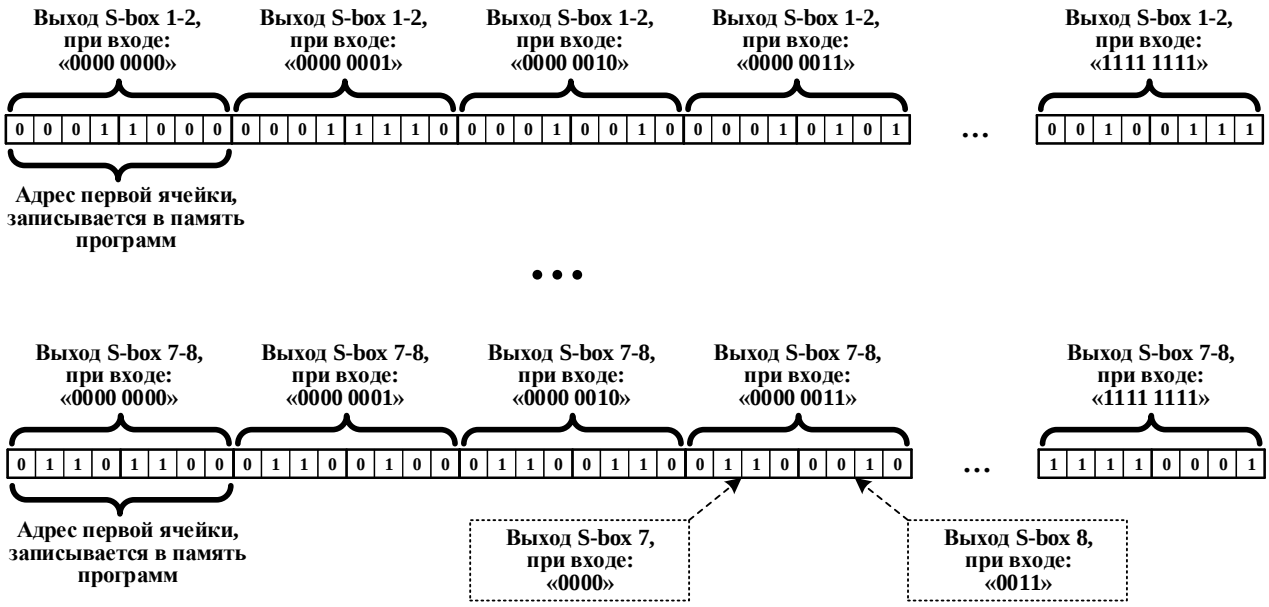


Рисунок Ж.4 – Процесс записи выходов сдвоенных S-box в ПЗУ

Следовательно, результат сложения $m(i)+k(r)$, может считываться побайтно одной командой, этот байт прибавляется к адресу первой ячейки соответствующего S-box, и по найденному адресу ПЗУ, считывается байт – выход соответствующей пары S-box (рисунок Ж.5). Данные операции будет необходимо повторить уже не восемь, а четыре раза. Этот подход увеличит скорость выполнения преобразования на S-box не менее чем в 5 раз, но для хранения таблиц преобразования четырёх сдвоенных S-box, вместо 128-ми байт ПЗУ потребуется 1024 байта, что, учитывая выигрыш в скорости преобразования и меньший объём кода, можно считать вполне оправданным.

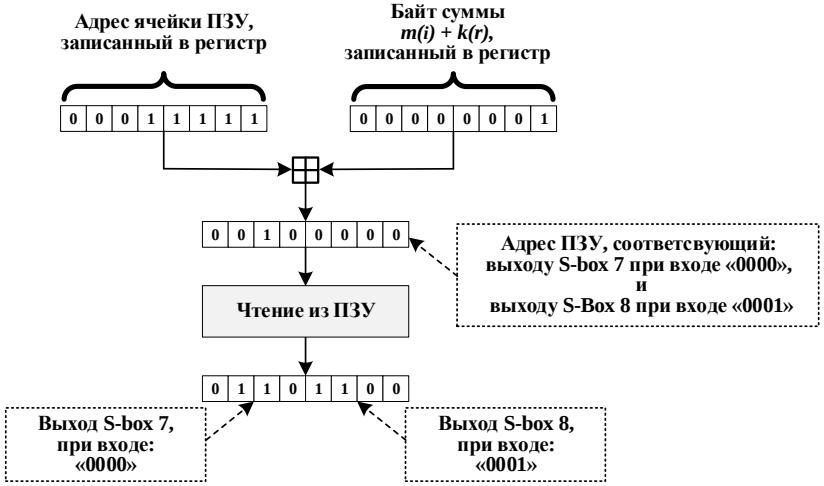


Рисунок Ж.5 – Преобразование на сдвоенном S-box

Приложение И. Графики, корреляционных векторов для рассматриваемых операций

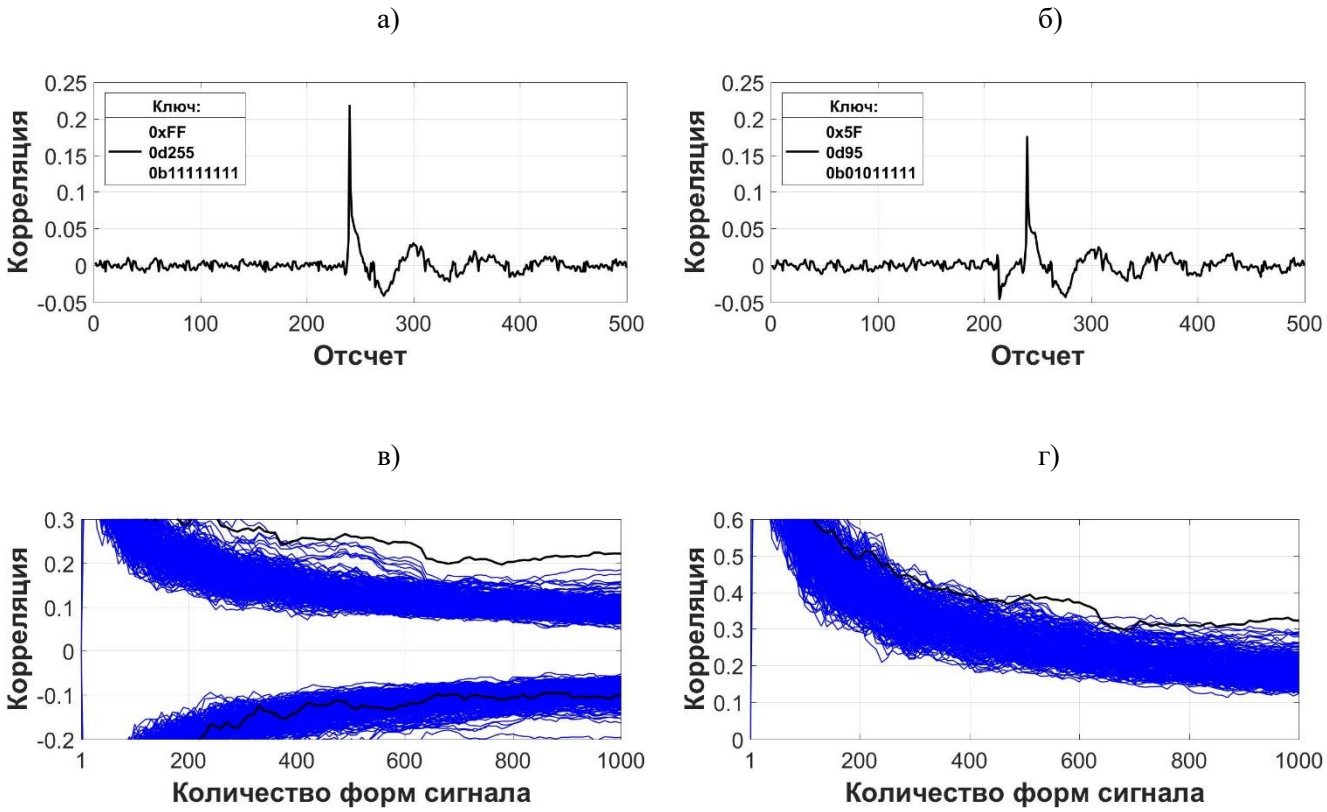


Рисунок И.1 – а), б) Корреляционные вектора, полученные при использовании в качестве рассчитываемого промежуточного значения, комбинации $S_{7-8}^{OUT}\left(i,\ k'_{7-8}\left(d\right)\right)$, считываемой из ячейки ОЗУ в чистую ячейку рабочего регистра, при обработке 50 000 форм сигнала, на соответственно истинном и наихудшем для злоумышленника ложном вариантах ключа; в) сводные графики корреляционных векторов; г) сводные графики корреляционных векторов по сумме максимумов и минимумов

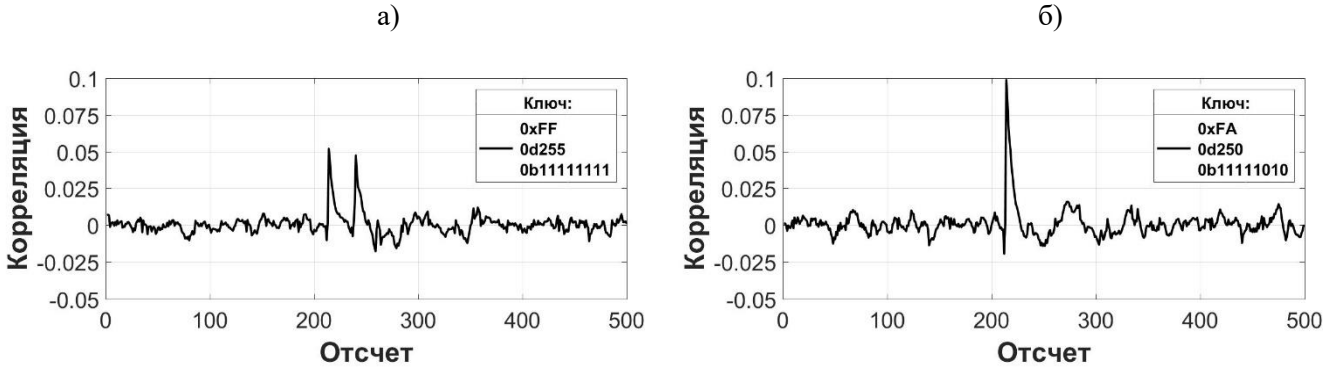
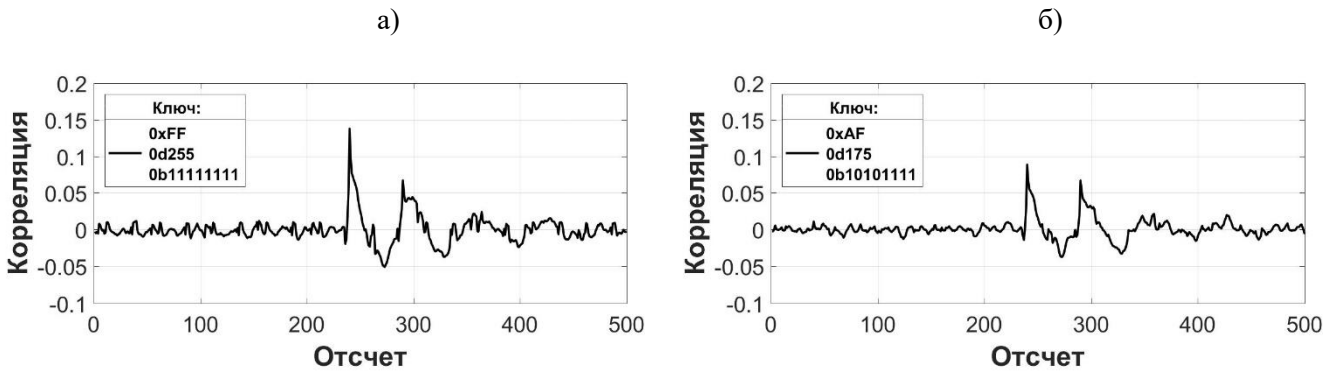


Рисунок И.2 – а), б) Корреляционные вектора, полученные при использовании в качестве рассчитываемого промежуточного значения, комбинации $S_{7-8}^{OUT}\left(i,\ k'_{7-8}(d)\right)$, считываемой из ячейки ОЗУ в предварительно заполненную ячейку рабочего регистра, при обработке 50 000 форм сигнала, на соответственно истинном и наихудшем для злоумышленника ложном вариантах ключа (с использованием модели веса Хэмминга)



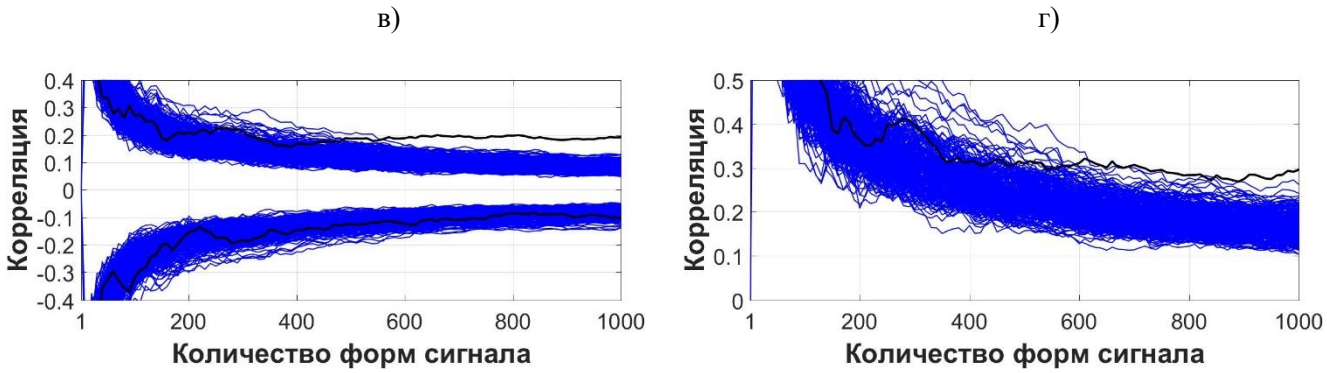
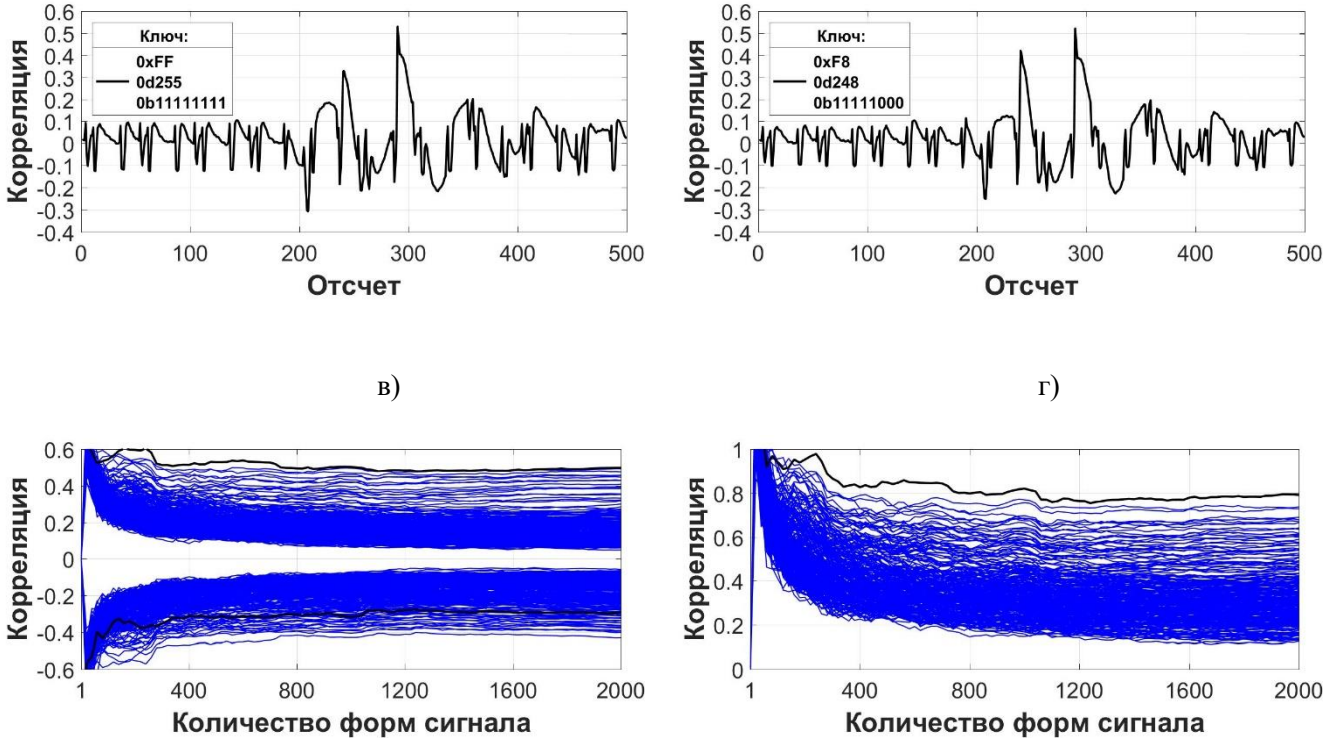


Рисунок И.3 – а), б) Корреляционные вектора, полученные при использовании в качестве рассчитываемого промежуточного значения, комбинации $S_{7-8}^{OUT}\left(i,\ k'_{7-8}(d)\right)$, считываемой из ячейки ОЗУ в предварительно заполненную ячейку рабочего регистра, при обработке 50 000 форм сигнала, на соответственно истинном и наихудшем для злоумышленника ложном вариантах ключа (с использованием модели расстояния Хэмминга); в) сводные графики корреляционных векторов; г) сводные графики корреляционных векторов по сумме максимумов и минимумов

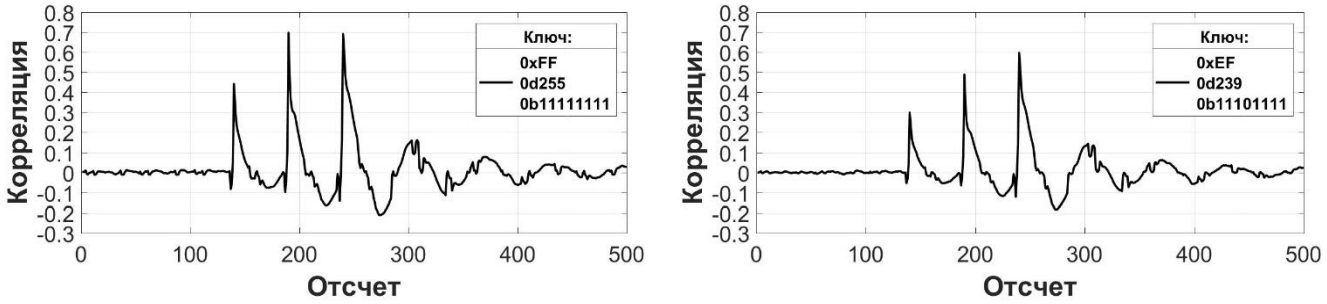
а) б)



в) г)

Рисунок И.4 – а), б) Корреляционные вектора, полученные при использовании в качестве рассчитываемого промежуточного значения, комбинации $S_{7-8}^{OUT}\left(i,\ k'_{7-8}\left(d\right)\right)$ считываемой из ячейки ПЗУ в чистую ячейку рабочего регистра, при обработке 50 000 форм сигнала, на соответственно истинном и наихудшем для злоумышленника ложном вариантах ключа; в) сводные графики корреляционных векторов; г) сводные графики корреляционных векторов по сумме максимумов и минимумов

а) б)



в) г)

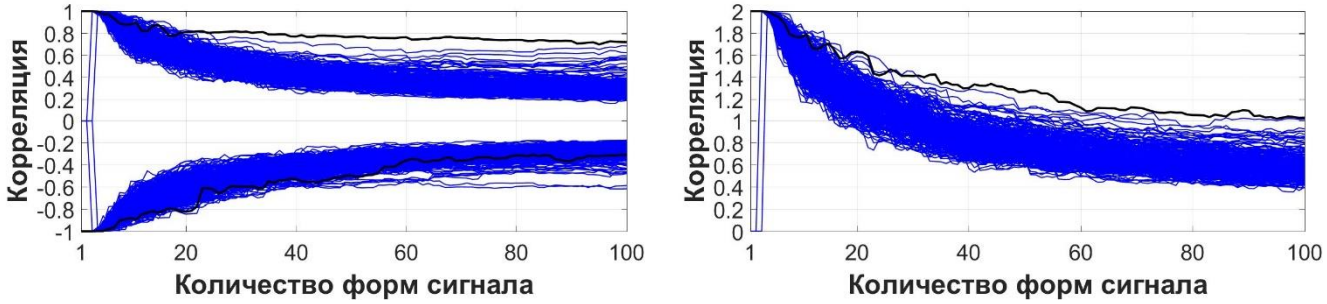
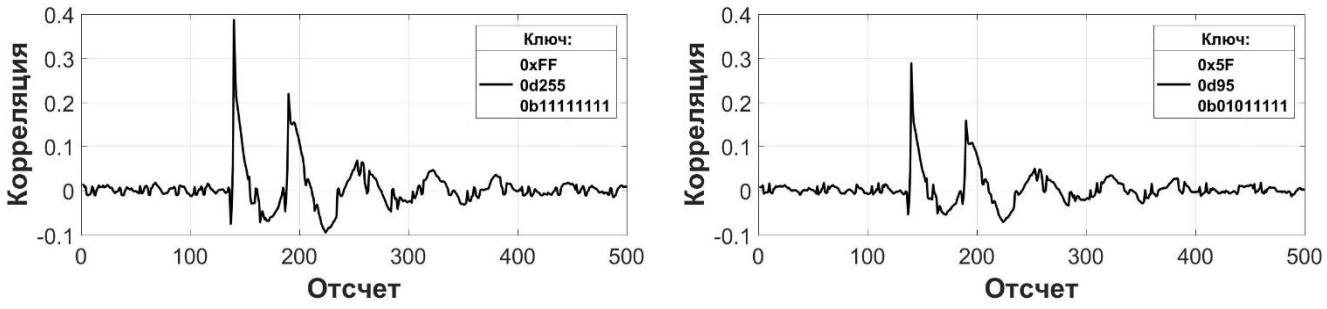


Рисунок И.5 – а), б) Корреляционные вектора, полученные при использовании в качестве рассчитываемого промежуточного значения, комбинации $S_{7-8}^{OUT}\left(i,\ k'_{7-8}\left(d\right)\right)$ копируемой из ячейки одного рабочего регистра в чистую ячейку другого рабочего регистра, при обработке 50 000 форм, сигнала на соответственно истинном и наихудшем для злоумышленника ложном вариантах ключа; в) сводные графики корреляционных векторов; г) сводные графики корреляционных векторов по сумме максимумов и минимумов

а) б)



в) г)

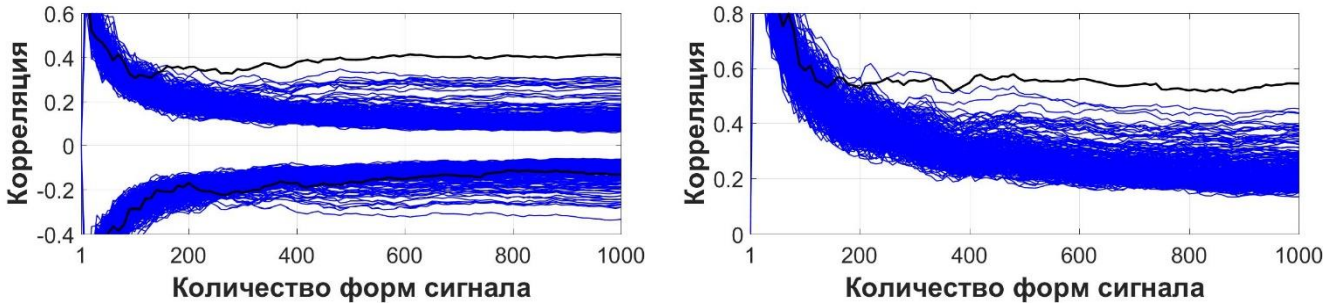


Рисунок И.6 – а), б) Корреляционные вектора, полученные при использовании, в качестве рассчитываемого промежуточного значения, комбинации $S_{7-8}^{OUT}\left(i,\ k'_{7-8}\left(d\right)\right)$, копируемой из ячейки одного рабочего регистра, в предварительно заполненную ячейку другого рабочего регистра, при обработке 50 000 форм сигнала, на соответственно истинном и «наихудшем» для злоумышленника ложном вариантах ключа (с использованием модели веса Хэмминга); в) сводные графики корреляционных векторов; г) сводные графики корреляционных векторов по сумме максимумов и минимумов

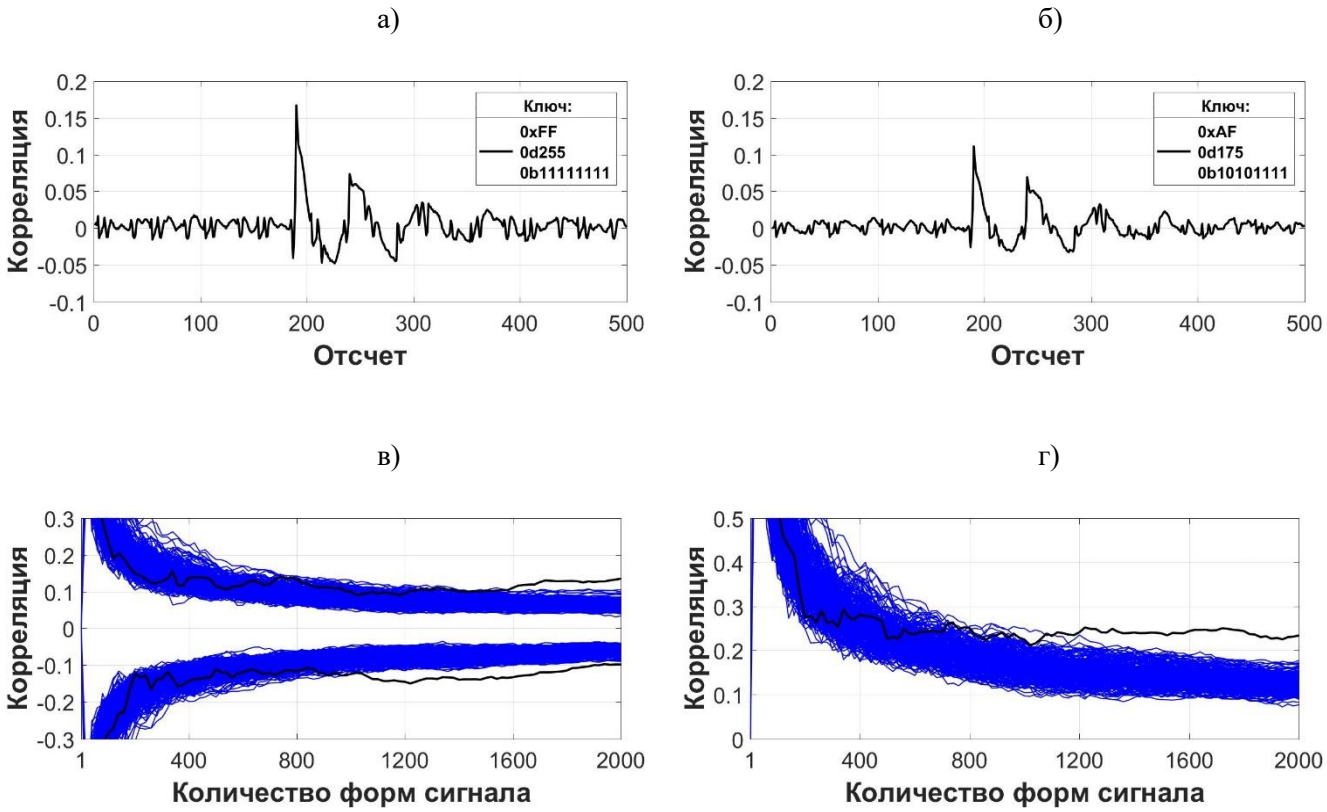
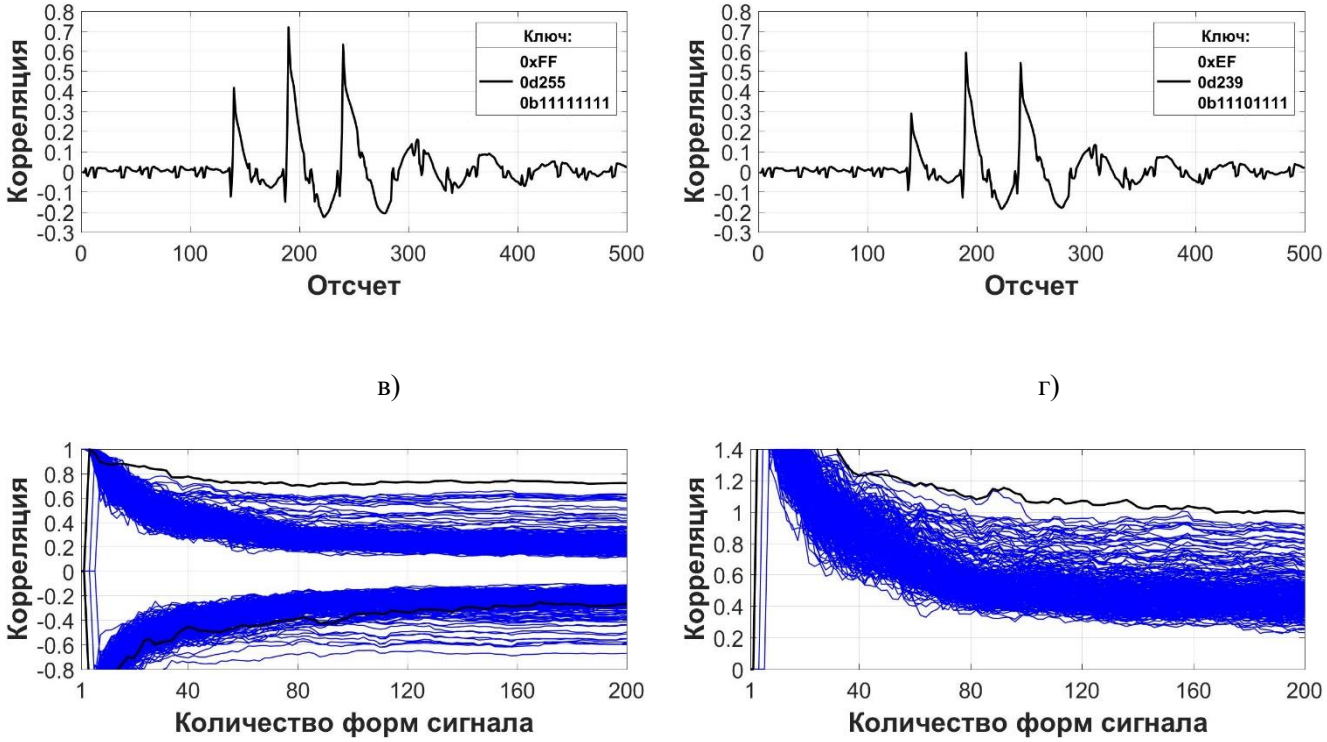


Рисунок И.7 – а), б) Корреляционные вектора, полученные при использовании в качестве рассчитываемого промежуточного значения, комбинации $S_{7-8}^{OUT}\left(i,\ k'_{7-8}\left(d\right)\right)$, копируемой из ячейки одного рабочего регистра, в предварительно заполненную ячейку другого рабочего регистра, при обработке 50 000 форм сигнала, на соответственно истинном и наихудшем для злоумышленника ложном вариантах ключа (с использованием модели расстояния Хэмминга); в) сводные графики корреляционных векторов; г) сводные графики корреляционных векторов по сумме максимумов и минимумов

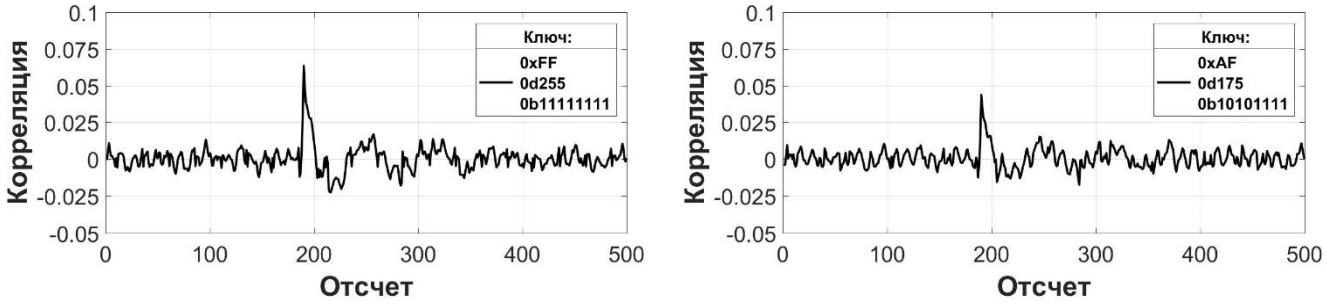
а) б)



в) г)

Рисунок И.8 – а), б) Корреляционные вектора, полученные при использовании в качестве рассчитываемого промежуточного значения, комбинации $S_{7-8}^{OUT}\left(i,\ k'_{7-8}\left(d\right)\right)$, складываемой по mod2 с известной случайной комбинацией, при обработке 50 000 форм сигнала, на соответственно истинном и наихудшем для злоумышленника ложном вариантах ключа; в) сводные графики корреляционных векторов; г) сводные графики корреляционных векторов по сумме максимумов и минимумов

а) б)



в) г)

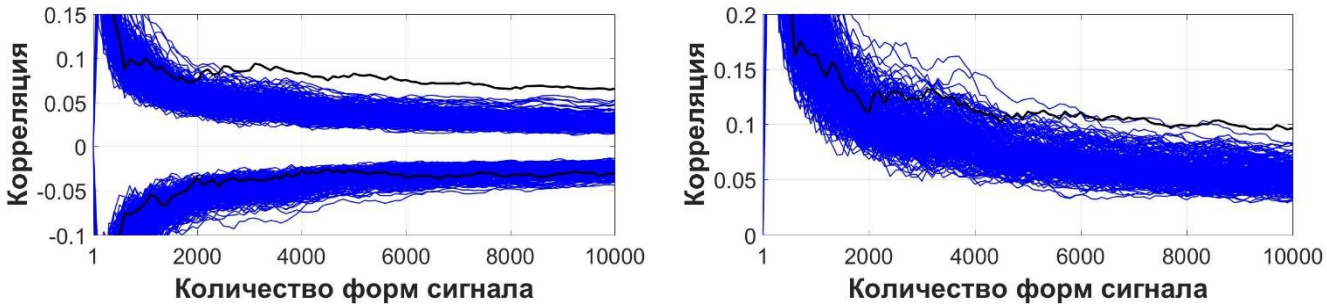


Рисунок И.9 – а), б) Корреляционные вектора, полученные при использовании в качестве рассчитываемого промежуточного значения, результата сложения по mod2 комбинации $S_{7-8}^{OUT}\left(i,\ k'_{7-8}\left(d\right)\right)$ с известной случайной комбинацией, при обработке 50 000 форм сигнала, на соответственно истинном и наихудшем для злоумышленника ложном вариантах ключа; в) сводные графики корреляционных векторов; г) сводные графики корреляционных векторов по сумме максимумов и минимумов

Приложение К. Реализация УМЗ на основе балансировки на микроконтроллерах фирм Atmel и Microchip

Предлагаемая структура узлов предобработки/постобработки универсального метода защиты предполагает сбалансированность обрабатываемых данных. Ниже будут показаны принципы реализации сбалансированных операций сложения по mod2 и преобразования на S-box на ассемблере восьмиразрядных микроконтроллеров фирмы Atmel (семейство ATmega) и Microchip (семейство PIC). Основная информация о строении таких чипов была дана в разделе 1.3.

Реализация УМЗ на основе балансировки, начинается с балансировки байт сообщения, на ассемблере микроконтроллера ATmega эта процедура описывается весьма просто – пятью элементарными командами. Положим, что в регистр $R1$ был помещён восьмибитовый блок сообщения, а в регистры $R20$ и $R21$ константы: «0000 1111» и «1111 0000» соответственно. Тогда операции по балансировке байта сообщения можно изобразить рисунком К.1:

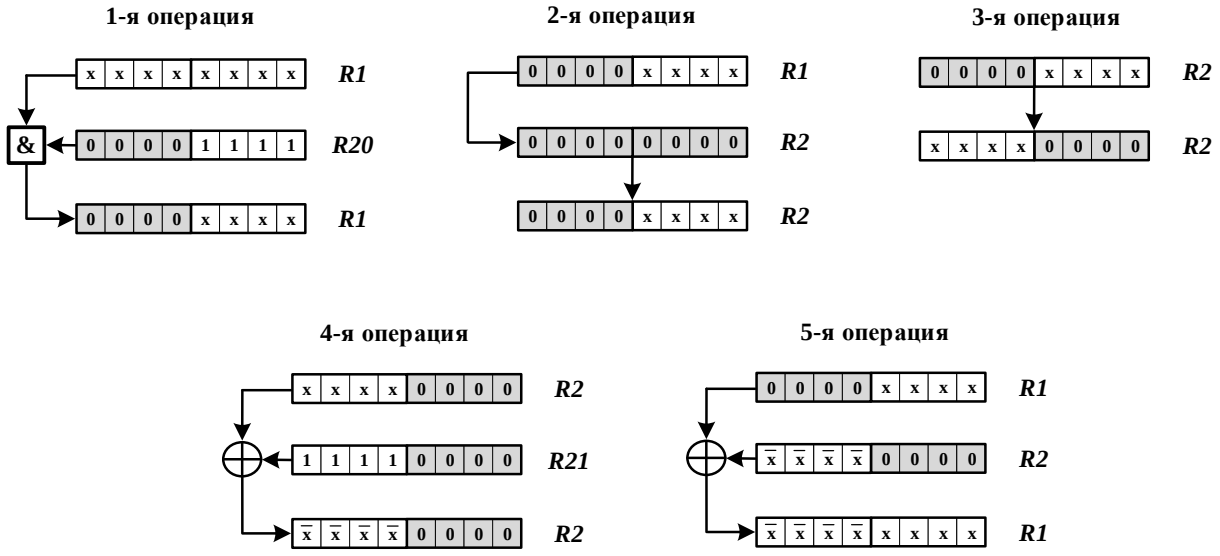


Рисунок К.1 – Этапы балансировки байта сообщения

Первой командой логического «И» («AND $R1$, $R20$») в байте сообщения обнуляется старший ниббл, а младший сохраняется неизменным.

Второй командой («MOV $R2$, $R1$»), содержимое регистра $R1$ копируется в другой «временный» регистр $R2$.

Третьей командой («SWAP $R2$») осуществляется перестановка ниббов регистра $R2$.

Четвёртой командой («EOR $R2$, $R21$») инвертируется старший ниббл регистра $R2$, а младший ниббл остаётся неизменным (равным «0000»), такой результат достигается обычным сложением по mod2 содержимого регистров $R2$ и $R21$, в который предварительно была записана комбинация «1111 0000».

Пятой командой («EOR $R1$, $R2$») регистры $R1$ и $R2$ складываются по mod2 – образуя сбалансированный байт.

Реализация защищённой операции сложения по mod2 на ассемблере также выполняется весьма просто – обычной командой сложения по mod2: EOR $R2, R1$, при этом в регистре $R1$ хранится сбалансированный байт сообщения, а в регистре $R2$ байт ключа, младший и старший ниббл которого повторяют друг друга (ключ предварительно загружается в регистр $R2$ из Flash или EEPROM памяти), результат операции помещается в регистр $R2$, перезаписывая хранящиеся в нём данные. При этом, согласно формуле (4.8), сбалансированным окажется не только вес Хэмминга результата сложения по mod2, но и расстояние Хэмминга между результатом сложения и содержимым регистра $R2$, которое тот перезаписывает.

Необходимо обратить внимание, что в микроконтроллерах фирмы Atmel реализован так называемый регистр статуса (регистр SREG) это дополнительный восьмиразрядный регистр, в который записывается информация о результате выполнения последней логической или арифметической команды. Схематично регистр статуса изображён на рисунке К.2, после выполнения защищённой операции сложения по mod2 в этом регистре будут изменяться два бита: S и N – они оба будут устанавливаться в единицу в том случае, если старший бит результата сложения по mod2 окажется равен единице. Это событие очевидно будет происходить ровно для половины обрабатываемых сообщений, для каких именно – зависит от выбранного ключа. Проведённое моделирование показало, что использование этой утечки для успешной реализации DPA оказывается незначительно сложнее классической реализации атаки.

Рисунок К.2 – Регистр статуса микроконтроллера Atmel

Принципиальное устранение описанной утечки было бы возможно включением операции сложения по mod2 в таблицу преобразований S-box. Данный метод предполагает использование 16-ти разных S-box (по количеству возможных вариантов ключа), таблицы преобразования которых модифицированы согласно формуле (К.1). При этом каждый раунд предобработки/постобработки должен состоять лишь из преобразования на ключевых S-box. Из

формулы (К.1) видно, что результат такого преобразования будет полностью соответствовать результату преобразования суммы подблоков сообщения и ключа на обычном S-box.

$$S_k\left(m(i)\right) = S\left(m(i) \oplus k\right), \qquad \text{(К.1)}$$

где $S_k(\cdot)$ – ключевой S-box; $S(\cdot)$ – обычный S-box; $m(i)$ – подблок сообщения; k – подблок ключа.

Однако данный подход оказывается неприменим на практике – реализация преобразования на S-box с применением так называемого подхода «чтение со смещением» (приложение Ж) включает в себя операцию арифметического сложения или сложения по mod2, поэтому применением табличного преобразования, исключить описанную выше утечку окажется невозможно.

В данном случае можно предложить лишь способ снижения утечки информации, в частности обеспечить защиту лишь от атаки с использованием модели расстояния Хэмминга. Ассемблер микроконтроллеров ATmega включает команду установки/сброса любого бита регистра статуса, поэтому можно предложить заранее, перед выполнением команды сложения по mod2, выполнить команду установки в единицу бита S или N (далее примем, что устанавливается бит S). В дальнейшем в результате выполнения операции сложения по mod2 могут наступить два события (рисунок К.3):

- Старший бит результата сложения окажется равным нулю – в этом случае биты S и N регистра статуса будут установлены в ноль. Следовательно, будет затрачена энергия на изменение значения бита S (переход $1 \to 0$).
- Старший бит результата сложения окажется равен единице – в этом случае биты S и N регистра статуса будут установлены в единицу. Следовательно, будет затрачена энергия на изменение значения бита N (переход $0 \to 1$).

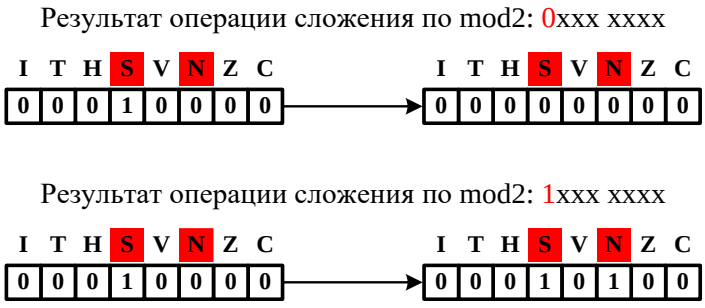


Рисунок К.3 – Изменения регистра статуса, в зависимости от результата операции

Отсюда понятно, что в обоих случаях расстояние Хэмминга между содержимым регистра статуса до выполнения операции сложения по mod2 и после, будет одинаковым (равным

единице). Поэтому, если положить, что на переходы $0 \to 1$ и $1 \to 0$ чипом затрачивается одинаковая энергия, то можно сделать вывод, что энергопотребление чипа будет одинаковым.

Однако очевидно, что вес Хэмминга комбинации, пересылаемой от АЛУ в регистр статуса, в первом случае будет равен нулю, а во втором случае – двум. Результаты проведённого ранее эксперимента показали, что б**о**льшая часть энергопотребления затрачивается именно на пересылку комбинации по шине данных, а не на перезапись хранящихся в ячейке памяти данных. Отсюда можно сделать вывод, что, по крайней мере в отношении исследованных на практике чипов фирмы Atmel, описанный метод защиты в указанном месте окажется не эффективным. Поэтому было принято решение исследовать возможность реализации УМЗ на основе балансировки на микроконтроллере семейства PIC фирмы Microchip (в частности PIC16F877A).

Балансировка подблоков сообщения на микроконтроллерах PIC принципиально реализуется тем же образом, что и на микроконтроллерах ATmega (рисунок К.1). Сложение по mod2 сбалансированных подблоков сообщения с ключом также реализуется одной командой, а результат перезаписывает ячейку в которой содержалось значение ключа. Однако, в отличие от микроконтроллеров ATmega, на этой операции не будет происходить изменение бит в регистре статуса (на микроконтроллерах PIC регистр статуса является значительно менее функциональным, чем на ATmega). Поэтому реализация операции сложения по mod2 на микроконтроллерах PIC окажется защищённой.

Далее рассмотрим реализацию защищённых S-box – согласно предлагаемой структуре узлов предобработки/постобработки на вход S'-box поступает сбалансированный байт $S_{l,\mathrm{j}}^{IN}(i) = \overline{t_{l,\mathrm{j}}(i)} \| t_{l,\mathrm{j}}(i)$ (где l - номер обрабатываемого подблока сообщения, j – номер раунда предобработки/постобработки), он состоит из младшего (информационного) ниббла $t_{l,\mathrm{j}}(i)$ и старшего (балансирующего) ниббла, являющегося инверсией информационного: $\overline{t_{l,\mathrm{j}}(i)}$ (рисунок 4.5), выход S'-box представляет собой также сбалансированную комбинацию $S_{l,\mathrm{j}}^{OUT}(i) = \overline{S\left(t_{l,\mathrm{j}}(i)\right)} \| S\left(t_{l,\mathrm{j}}(i)\right)$, где $S(\cdot)$ – функция, описывающая табличное нелинейное преобразование. Отсюда, а также учитывая, что преобразование на S'-box оптимальнее реализовывать как нахождение обратного элемента в конечном поле $GF\left(2^4\right)$, таблицу преобразования можно описать таблицей К.1.

Таблица К.1 – Таблица преобразования S'-box

Десятичный вид			**Двоичный вид (сбалансированный)**	
Вход S-box	**Выход S-box**		**Вход S'-box**	**Выход S'-box**
0	0		1111 0000	1111 0000
1	1		1110 0001	1110 0001
2	9		1101 0010	0110 1001
3	14		1100 0011	0001 1110
4	13		1011 0100	0010 1101
5	11		1010 0101	0100 1011
6	7		1001 0110	1000 0111
7	6		1000 0111	1001 0110
8	15		0111 1000	0000 1111
9	2		0110 1001	1101 0010
10	12		0101 1010	0011 1100
11	5		0100 1011	1010 0101
12	10		0011 1100	0101 1010
13	4		0010 1101	1011 0100
14	3		0001 1110	1100 0011
15	8		0000 1111	0111 1000

Табличное преобразование наиболее оптимально реализуется на ассемблере с использованием так называемого подхода «чтение со смещением». Данный подход предполагает особый алгоритм записи таблицы преобразования в ячейки ПЗУ чипа и соответствующий ему алгоритм чтения, принципы работы такого алгоритма были рассмотрены в приложении Ж.

В классической (незащищённой) реализации табличного преобразования на S-box, при программировании чипа, в ячейки ПЗУ, последовательно записываются выходные значения S-box: $S_out(y)$, $y = 0, ..., 255$ получаемые в случае прихода на вход комбинации 0: «0000 0000», затем 1: «0000 0001», и так далее вплоть до выходного значения, получаемого по приходу комбинации 255: «1111 1111», при этом адрес первой ячейки: S_addr заносится в память программ. Таким образом, в ячейке ПЗУ с адресом, к примеру: S_addr будет хранится выход S-box при входе «0000 0000», в ячейке с адресом $S_addr + 1$, будет хранится выход S-box при входе «0000 0001» и так далее (рисунок К.4). Отсюда понятно, что выход S-box можно найти смещением от адреса S_addr на количество ячеек, равное значению входа S-box.

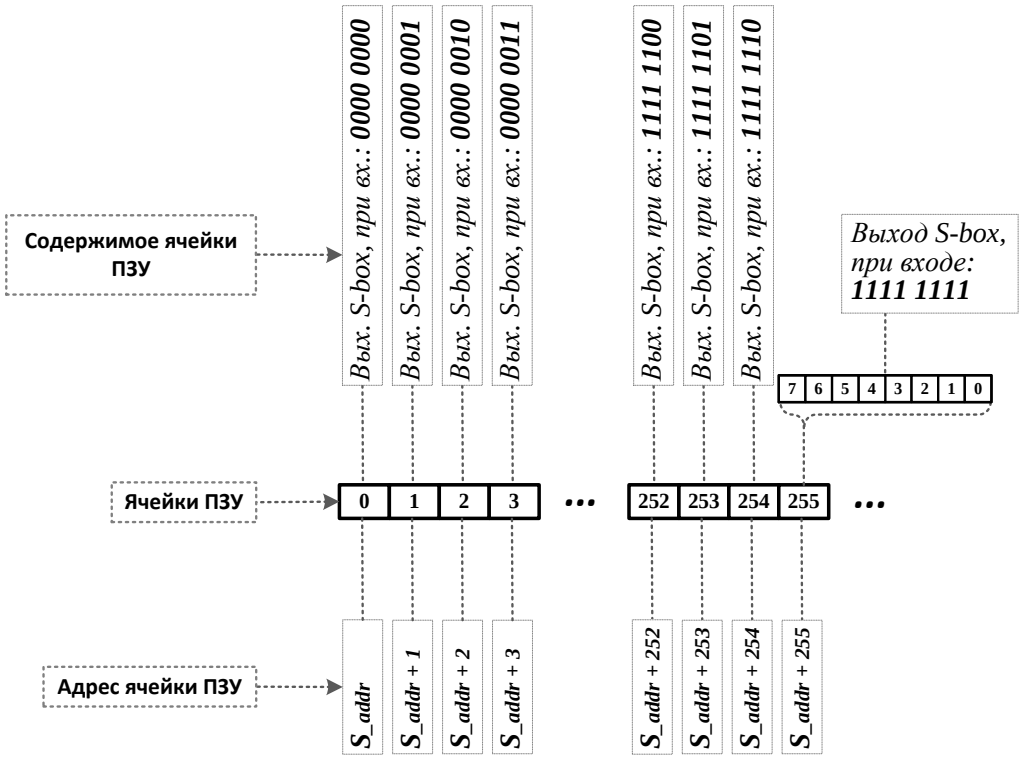


Рисунок К.4 – Демонстрация подхода к реализации S-box «чтение со смещением»

При выполнении чипом программы, преобразование на S-box реализуется последовательным выполнением двух операций:

- Вычисление адреса ячейки ПЗУ, в которой хранится комбинация выхода S-box, соответствующая его входу.
- Чтение содержимого ячейки ПЗУ.

Первая операция реализуется командой сложения байта входа S-box: S_in с адресом стартовой ячейки S-box: S_addr: $S_in_addr = S_addr + S_in$. Вторая операция реализуется командой чтения из ПЗУ: $S_out = READ_EEPROM(S_in_addr)$, где $READ_EEPROM(x)$ – условная команда загрузки из EEPROM памяти содержимого ячейки (байта данных) с адресом x (на практике, на ассемблере микроконтроллера PIC чтение из ячейки EEPROM реализуется последовательным выполнением комбинации команд, в частности: загрузка комбинации S_in_addr в «EEADR» – регистр адреса EEPROM; выставление двух бит в регистре «EECON1»; непосредственное считывание содержимого ячейки EEPROM из регистра «EEDATA»).

Для исключения возможности успешной реализации DPA с использованием промежуточных данных преобразования на S'-box, утечка информации по цепи электропитания от обеих операций должна быть устранена. В первую очередь необходимо отметить, что при классической реализации первой операции, производится арифметическое сложение адреса

EEPROM: S_addr с входом S'-box: S_in. Негативной особенностью операции арифметического сложения двух переменных является возможность возникновения переносов бит в соседние разряды. Их не будет возникать лишь в том случае, если значение S_addr окажется равным «0000 0000». (Выбрав любое иное значение S_addr, и пробегая все возможные значения S_in из таблицы К.1, несложно проверить, что при сложении $S_addr + S_in$, перенос будет возникать хотя бы при одном значении S_in.) На практике представляется крайне сложно использовать описанную утечку для успешной реализации DPA, однако для защиты от потенциальных атак будет разумно её устранить. Поэтому операцию арифметического сложения можно рекомендовать заменить сложением по mod2, преобразование на S'-box в этом случае будет описываться формулой (К.2):

$$S_in_xor = S_addr \oplus S_in,$$
$$S_out = READ_EEPROM(S_in_xor) \quad \text{(К.2)}$$

Необходимо отметить, что комбинация S_in (также как и S_out) является сбалансированной, комбинация S_addr является константой и не зависит ни от S_in ни от и иных секретных данных шифра, единственным источником утечки информации остаётся результат сложения S_in_xor (указывающий на адрес ячейки ПЗУ, в которой записан выход S'-box). Если злоумышленнику будут известны особенности реализации метода защиты на анализируемом чипе, то не составит труда сопоставить значения S_in_xor, со значениями выхода S'-box, хранящимися в соответствующих ячейках ПЗУ, поэтому комбинация S_in_xor должна быть сбалансированной. Ранее формулами (4.4)-(4.5) описывался простой способ балансировки результата сложения по mod2 – младший и старший нибблы комбинации S_addr должны быть одинаковыми. Поэтому, адрес S_addr может иметь одно из 16-ти значений: «0000 0000», «0001 0001», «0010 0010», …, «1111 1111» (далее положим, что $S_addr = 0001\ 0001$). При программировании чипа, с учётом выбранного значения S_addr, по формуле (К.2) вычисляются адреса 16-ти ячеек ПЗУ (по одной для каждого варианта входа S'-box). Затем в каждую ячейку ПЗУ записывается соответствующее значение S_out (таблица К.2). Подставляя значения S_addr и S_in из таблицы К.2 в формулу (К.2) несложно проверить что в результате получается корректное преобразование на S'-box, более того, все промежуточные данные обеих операций, составляющих преобразование на S'-box, окажутся защищены от DPA.

Таблица К.2 – Распределение таблицы преобразования S'-box по ячейкам ПЗУ

Адрес стартовой ячейки ПЗУ	**Вход S'-box**	**Адрес ячейки ПЗУ, для записи выхода S'-box**	**Адрес ячейки ПЗУ, для записи выхода S'-box, десятичный вид**	**Записываемый выход S'-box**
S _ addr	*S _ in*	*S _ in _ xor*	*S _ in _ xor*	*S _ out*
0001 0001	1111 0000	1110 0001	225	1111 0000
	1110 0001	1111 0000	240	1110 0001
	1101 0010	1100 0011	195	0110 1001
	1100 0011	1101 0010	210	0001 1110
	1011 0100	1010 0101	165	0010 1101
	1010 0101	1011 0100	180	0100 1011
	1001 0110	1000 0111	135	1000 0111
	1000 0111	1001 0110	150	1001 0110
	0111 1000	0110 1001	105	0000 1111
	0110 1001	0111 1000	120	1101 0010
	0101 1010	0100 1011	75	0011 1100
	0100 1011	0101 1010	90	1010 0101
	0011 1100	0010 1101	45	0101 1010
	0010 1101	0011 1100	60	1011 0100
	0001 1110	0000 1111	15	1100 0011
	0000 1111	0001 1110	30	0111 1000

Из таблицы К.2 видно, что при реализации S'-box запись таблицы преобразования в ячейки ПЗУ производится не последовательно и с интервалами (в каждую 15 ячейку). Поэтому необходимо следить, чтобы другие компоненты, выполняемой чипом программы, не перезаписывали эти данные.

Комбинация с выходов S'-box последнего раунда узлов предобработки/постобработки должна быть избавлена от балансирующей части, проще всего это осуществляется выполнением дополнительной операции логического «И» выхода S'-box и соответствующей «маски». Реализация этой операции в точности аналогична операции 1 из рисунка К.1.

После обработки соседнего подблока сообщения по тому же алгоритму, производится перестановка нибблов, после этого оба четырёх-битовых подблока «склеиваются» в байт операцией сложения по mod2 (по аналогии с операцией 5 из рисунка К.1).

Оценка требуемых дополнительных вычислительных ресурсов чипа для реализации УМЗ, не может являться строгой и зависит от особенностей конкретной ассемблерной реализации защищаемого шифра и метода защиты. В качестве примера можно привести L-преобразование

шифра «Кузнечик», на каждом из дести раундов которого, 208 раз применяется операция умножения в конечном поле – она может быть реализована минимум двумя способами:

- С использованием большого количества машинных циклов (то есть большого времени), но низкими требованиями к объёму ПЗУ.
- С использованием небольшого количества машинных циклов (то есть значительно быстрее), но с чрезвычайно большими требованиями к объёму ПЗУ.

Помимо операции умножения, существует целый ряд операций, при реализации которых также необходимо находить баланс между требованиями к аппаратным ресурсам чипа и скоростью шифрования. Созданный автором ассемблерный код для микроконтроллера PIC требует приблизительно 70 тысяч машинных циклов и 2.5 тысячи слов команд (порядка 4 300 байт памяти программ) для реализации 10-ти раундов шифра «Кузнечик». В тоже время, на реализацию операций предобработки и постобработки потребуется дополнительно 30 тысяч машинных циклов и 500 слов команд. Отсюда скорость шифрования, в представленной реализации, снизится не более чем на 50%, причём потребуется пренебрежимо малый объём памяти для хранения дополнительного кода программы (включая дополнительный ключ и таблицу преобразования S'-box). В итоге можно заключить, что предлагаемый метод защиты потребует весьма небольших дополнительных аппаратных ресурсов чипа, и может быть принципиально реализован даже на бытовых микроконтроллерах, из нижнего и среднего ценовых сегментов.

При подготовке данного приложения использовалось описание строения и принципов работы микроконтроллеров, представленные в следующих источниках: [1; 3; 7; 18]. Основное содержание приложения и полученных результатов является оригинальным и изложено в работах автора [33; 34].

Приложение Л. Раунд шифра «Кузнечик»

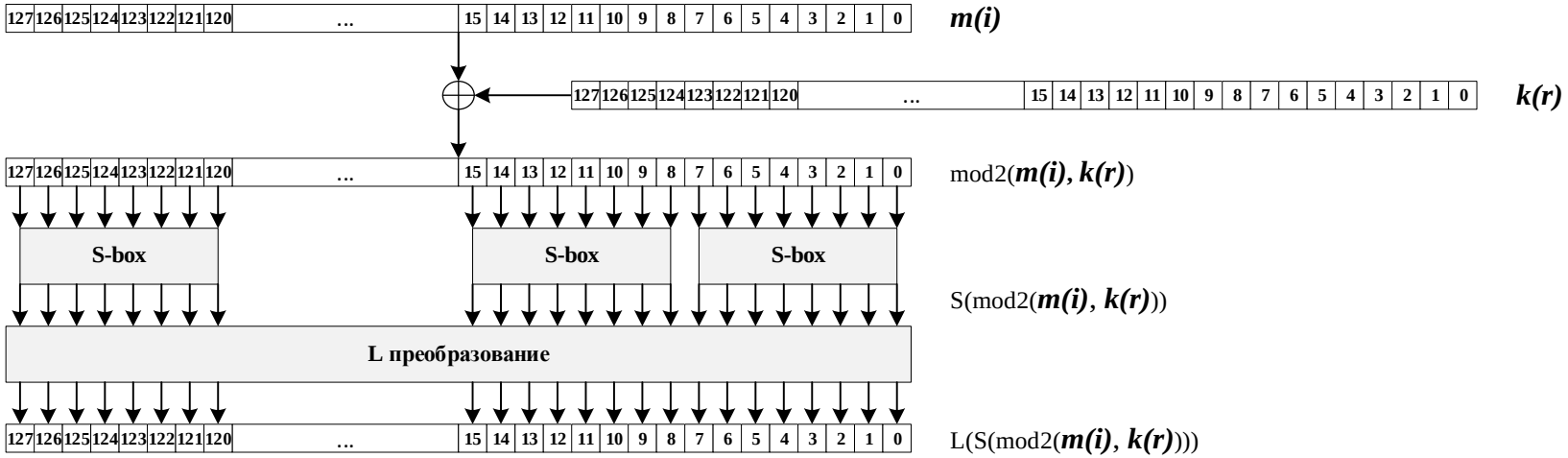


Рисунок Л.1 – Схема одного раунда шифра «Кузнечик»

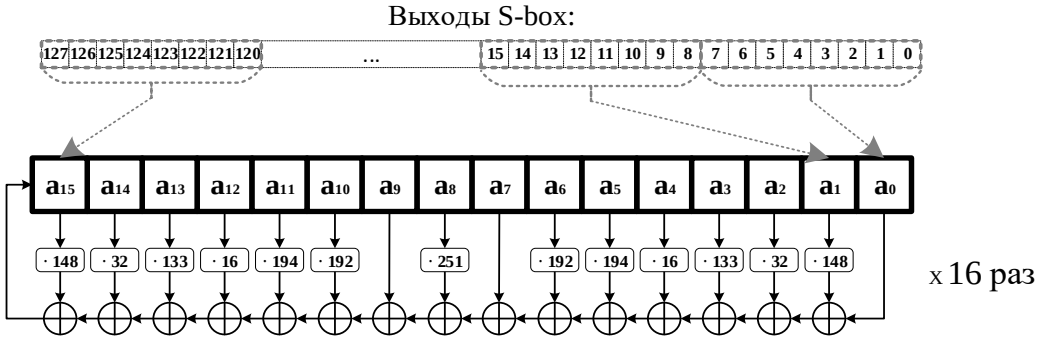


Рисунок Л.2 – Схема L преобразования шифра «Кузнечик»

Приложение М. Предлагаемая структура УМЗ

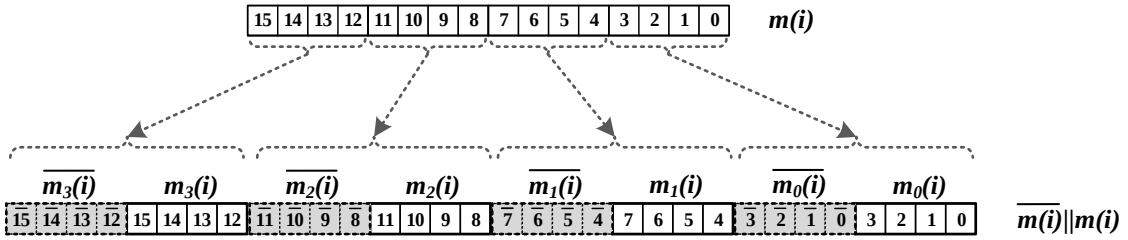


Рисунок М.1 – Разбиение 16-ти битового сообщения на блоки по четыре бита и балансировка каждого блока

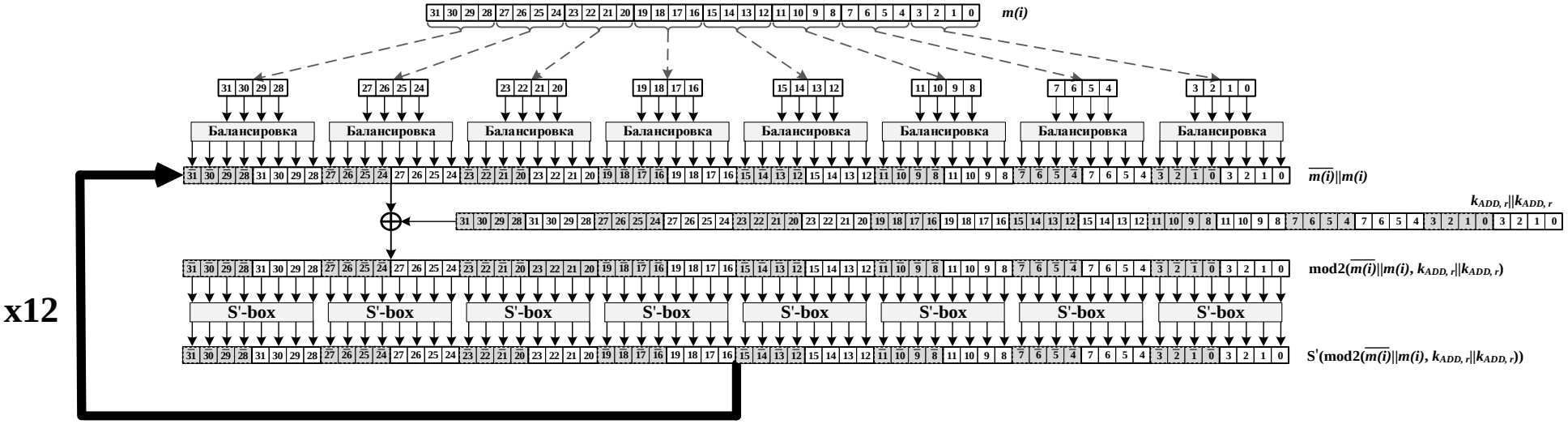


Рисунок М.2 – Предобработка 32-х битового блока входа шифра

Приложение Н. Акт о внедрении научных результатов

«Утверждаю»

Генеральный директор АО «Научные приборы»

г. Санкт-Петербург

В.А. Елохин

06 / 09 / 2017 г.

АО «НАУЧНЫЕ ПРИБОРЫ»
ИНН 7826012838 / 783901001
190103 Г С. ПЕТЕРБУРГ Рижский. 26
ТЕЛ:251-28-50 ФАКС: 251-73-53
ОГРН 1027810338343
06.09.2017г №100-374/Е

АКТ

о внедрении научных результатов,

полученных Тихоновым Сергеем Владимировичем

Комиссия в составе:

- Позднышева Михаила М., руководителя отдела информационных технологий;
- Бородинова Андрея Геннадьевича, кандидата физико-математических наук, начальника сектора информационных проектов;
- Изотова Бориса Викторовича, кандидата технических наук, заместитель начальника конструкторско-технологического отдела по информационной безопасности

составила настоящий акт о том, что научные результаты, полученные Тихоновым Сергеем Владимировичем, а именно:

1) оптимальная конструкция измерительной установки, предназначенной для сбора данных об энергопотреблении микрочипов, такого качества, которое позволяет использовать их при реализации побочных атак анализа энергопотребления;
2) алгоритм предварительной обработки данных об энергопотреблении микрочипов, с возможностью автоматизации его выполнения;
3) результаты исследования величины утечек информации по цепи питания микроконтроллеров фирмы Atmel, при выполнении ими различных команд использованы в отчёте о научно-исследовательской работе:

«Исследование основных направлений создания средств и систем защиты электронных документов нового поколения»: отчет о НИР: АО «Научные приборы»; рук.: Позднышев М.М.; исполн.: Бородинов А.Г. [и др.]; – СПб, 2014 г. – 323 с. – Библиогр.: с. 249-280. – Инв. №231-АО от 16.11.2015 г.

Применение указанных результатов также позволило:

1) уточнить требования к разрабатываемым электронным идентификационным документам;
2) повысить надёжность функционирования производимого оборудования со встроенными блоками электронной идентификации.

Руководитель отдела информационных технологий — М.М. Позднышев

Начальник сектора информационных проектов, канд. физ.-мат. наук — А.Г. Бородинов

Заместитель начальника конструкторско-технологического отдела по информационной безопасности, канд. техн. наук — Б.В. Изотов