

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
Политехнический институт
Факультет информационных технологий и электроники
Кафедра
«Информационная безопасность систем
и технологий»

Специальность 10.05.03 – Информационная безопасность
автоматизированных систем
Специализация - Защищенные автоматизированные
системы управления

ДИПЛОМНЫЙ ПРОЕКТ

на тему

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ВСТРАИВАЕМОГО УСТРОЙСТВА ПЕРЕДАЧИ ЗАШИФРОВАННОЙ ИНФОРМАЦИИ

Студент [подпись] Киреев Дмитрий Алексеевич

Руководитель

ведущий инженер НИО-25 АО «ПНИЭИ» [подпись] Дуплянский А.Л.

Нормоконтролёр [подпись] Иванов А.П.

Рецензент

зам. ген. директора АО «ПНИЭИ» [подпись] Сомкин С.А.

Работа допущена к защите (протокол заседания кафедры от 25.06.21 № 16)

Заведующий кафедрой [подпись] С.Л. Зефирова

Работа защищена с отметкой хорошо (протокол заседания ГЭК
от 28.06.21 № 4)

Секретарь ГЭК [подпись] А.П. Иванов

Подпись [подпись] уполномоченно

Ведущий инженер [подпись] И.Л. Орлова



Пенза 2021

Содержание

Введение.....	9
1 Анализ состояния технической проблемы.....	10
1.1 Актуальность разработки устройства передачи зашифрованной информации.....	10
1.2 Требования, предъявляемые к разрабатываемым средствам криптографической защиты информации.....	11
1.3 Анализ угроз информационной безопасности.....	14
2 Сбор и анализ исходных данных для проектирования.....	15
2.1 Описание устройства-прототипа.....	15
2.2 Описание схемы структурной и электрической принципиальной.....	15
2.3 Обоснование выбора среды разработки программного модуля.....	19
2.4 Анализ схемы устройства-прототипа с точки зрения вопросов информационной безопасности.....	21
2.5 Обзор угроз и методов защиты от аппаратных закладок.....	23
3 Разработка программного обеспечения.....	33
3.1 Описание функционирования программного обеспечения.....	33
3.2 Описание функционирования средства защиты от активации закладок в импортных интегральных микросхемах.....	34
3.4 Описание среды разработки STM32CubeIDE.....	35
3.5 Описание алгоритмов криптозащиты и имитозащиты.....	39
3.6 Разработка программного обеспечения.....	45
3.7 Тестирование и отладка программного обеспечения.....	50
4 Вредные психофизиологические факторы, влияющие на органы зрения специалиста по защите информации. Методы и средства физической культуры, снижающие их воздействие.....	54
Заключение.....	64
Список использованных источников	65

Приложение А (обязательное) Текст программы на исходном языке.....	67
Приложение Б (обязательное) Результаты тестирования программного обеспечения.....	78

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

УЛК 004.056.55

Факультет
информационных технологий и
электроники
Выпускающая кафедра:
Информационная безопасность систем
и технологий
Учебная группа 16ПИ1

УТВЕРЖДАЮ
Зав. кафедрой ИБСТ
~~к.т.н. доцент~~ С.Л. Зефиров

«26» 09 2021 г.

ТЕХНИЧЕСКОЕ ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
студента группы 16ПИ1 Киреева Дмитрия Алексеевича

Специальность 10.05.03 – Информационная безопасность
автоматизированных систем

Тема дипломного проекта:

Программное обеспечение встраиваемого устройства передачи зашифрованной
информации

Руководитель ВКР

ведущий инженер НИО-25 АО «ПНИЭИ»

 А.Л. Дуплянский

Нормоконтролер

к.т.н. доцент

 А.П. Иванов



1 Цели и задачи ВКР

Цель работы – разработка программного обеспечения реализующего алгоритм криптографического преобразования передаваемой информации и функции защиты от активации закладок в импортных интегральных микросхемах в соответствии с техническими требованиями.

2 Тактико-технические требования к выполнению ВКР

Технические требования:

- устройство-прототип должно быть дополнено программным модулем, реализующим алгоритм блочного шифрования с длиной блока 64 бита по ГОСТ Р 34.12-2018 в режимах выработки имитовставки и гаммирования по ГОСТ Р 34.13-2015;

- программный модуль должен иметь функцию разрушения последовательности на стыке цифрового процессора обработки сигналов и программируемой логической интегральной схемы для защиты от активации возможных закладок в интегральных микросхемах иностранного производства;

- программный модуль должен быть разработан в среде STM32Cube IDE (аппаратная платформа – процессор STM32F4150G);

- программный модуль должен обеспечить зашифрование/расшифрование информации со скоростью до 16 кбит/с;

- для реализации программного модуля вычислительный ресурс процессора не должен быть задействован более чем на 20%;

- в программный модуль в качестве первого параметра указывается адрес входного массива данных, в качестве второго – адрес выходного массива данных, в качестве третьего – вызов функции режима работы;

- объем программного модуля не должен превышать 16 Кбайт.

В ВКР должны быть рассмотрены следующие вопросы:

- описание устройства-прототипа
- сбор и анализ исходных данных для проектирования
- анализ состояния технической проблемы

- описание функционирования программного обеспечения
 - обоснование выбора среды разработки программного модуля;
 - разработка программного обеспечения
 - тестирование и отладка программного обеспечения
 - вредные психофизиологические факторы, влияющие на органы зрения
- специалиста по защите информации. Методы и средства физической культуры, снижающие их воздействие.

3 Требования к разрабатываемой документации

Отчет по ВКР выполняется в соответствии с требованиями документов, действующих в Пензенском государственном университете.

Перечень и содержание графической части ВКР:

- схема электрическая структурная встраиваемого устройства передачи информации (устройство-прототип) – 1 л., ф. А1;
- принцип защиты от активации закладок в импортных интегральных микро схемах – 1 л., ф. А1;
- алгоритм программы зашифрования и расшифрования в режиме гаммирования – 1 л., ф. А1 (плакат);
- алгоритм выработки имитовставки – 1 л., ф. А1 (плакат);
- алгоритм работы модуля криптографической защиты передаваемой информации в устройстве – 2 л., ф. А1 (плакат);
- результаты замеров производительности – 1 л., ф. А1 (плакат).

Электронная копия отчета и графической части ВКР должна быть представлена на магнитном или оптическом носителе.

4 Сроки выполнения ВКР

Тема утверждена приказом ректора ПГУ № 118 от «18» мая 2021 г.

Дата выдачи задания «17» апреля 2021 г.

Время дипломного проектирования с 12.01.2021 г. по 17.06.2021 г.

Задание к исполнению принял «16» 04 2021 г.

Исполнитель ВКР _____ Киреев Д.А.



Обозначения и сокращения

АЗ – аппаратная закладка;

ИВЭП — источник вторичного электропитания;

ИМС – интегральная микросхема;

КПД – коэффициент полезного действия;

МК – микроконтроллер;

ОЗУ – оперативное запоминающее устройство;

ООД – окончное оборудование данных;

ПК – персональный компьютер;

ПЛИС – программируемая логическая интегральная микросхема;

ПО – программное обеспечение;

ПСП – псевдослучайная последовательность;

ПЭМИН – побочные электромагнитные излучения и наводки;

СП – синхропосылка;

СКЗИ – средство криптографической защиты информации;

СнК – система на кристалле;

УПЗИ - устройство передачи зашифрованной информации;

ЭВМ – электронно вычислительная машина;

IDE – интегрированная среда разработки;

RTL – уровень регистровых передач.

Введение

Вопрос защиты информации за последние годы стал жизненно важным аспектом личной и финансовой безопасности. Без достаточного уровня квалификации применение современных стандартов и средств защиты информации невозможно. Криптографические методы – наиболее эффективные средства защиты информации в системах передачи информации.

Разработка радиоэлектронных устройств ведется постоянно, в следствии как растущих возможностей элементной базы, так и требований законодательства. В зависимости от реализации и предъявляемых к ней требований, в каждом конкретном случае осуществляется выбор между аппаратной, программной и аппаратно-программной воплощение средств защиты.

Устройство-прототип, для которого нужно разработать программный модуль криптозащиты – модемный модуль встраиваемый в радиостанции дециметрового диапазона. Он был создан для передачи информации ограниченного доступа, не содержащей государственную тайну.

В результате выполнения работы должно быть получено программное обеспечение, реализующее функции криптографического преобразования информации и защиты от активации закладок в импортных интегральных микросхемах для устройства передачи зашифрованной информации.

1 Анализ состояния рассматриваемой технической проблемы

1.1 Актуальность разработки средств криптографической защиты информации

Целью работы является разработка программного обеспечения, реализующего алгоритм криптографического преобразования передаваемой информации и функции защиты от активации закладок в импортных микросхемах для ранее разработанного устройства (или устройства-прототипа). Устройство-прототип – это малогабаритный модемный модуль, встраиваемый в радиостанции декаметрового диапазона. Предполагается, что программное обеспечение модемного модуля должно быть дополнено новым режимом работы для улучшения его качественных характеристик.

Потребность дополнения программного обеспечения устройства возникла в следствии разгоревшейся с новой силой в Российской Федерации волной импортозамещения и повышения производительности элементных баз. С каждым годом всё больше функций раньше существовавших как специализированное аппаратное решение интегрируется в вычислительное ядро. А тенденции к использованию планировщиков и автоматизации приводит к мысли о возможности несанкционированных действий со стороны программ, устройств или внешних факторов.

Особые требования предъявляются к средствам криптографической защиты информации, используемым в портативных средствах связи. Такие средства должны быть компактными, должны иметь средства защиты ключей зашифрования, иметь малые энергозатраты и обеспечивать достаточно высокую скорость обработки и засекречивания информации. Под СКЗИ понимаются, предназначен информации ограниченного доступа, не содержащей сведений, составляющих государственную тайну.

Кроме криптографических методов защиты информации, при разработке СКЗИ, рассматривается ещё целый комплекс мер защиты со стороны коммутационных элементов аппаратуры, а также системы электропитания и

заземления. Так в данной работе предусмотрена дополнительная защита от активации закладок в импортных интегральных микросхемах, которая заключается в дополнительной обработке канальной информации, для исключения возможных внешних воздействий.

1.2 Требования, предъявляемые к разрабатываемым средствам криптографической защиты информации

Для новых СКЗИ сформулированы следующие требования:

- зашифрованное сообщение может быть прочитано только при наличии ключа;
- знание алгоритма защиты не должно влиять на надёжность;
- всякий ключ из множества возможных обязан гарантировать защиту информации;
- алгоритм шифрования должен допускать как программную, так и аппаратную реализацию.

Программные реализации от криптографических алгоритмов отличаются значительно меньшим быстродействием, но не требуют особых издержек бюджета и могут использоваться на протяжении длительного времени на нескольких различных устройствах. Программное шифрование не имеет выделенного процессора для выполнения необходимых алгоритмов и не имеет выделенной памяти для размещения защищенного ключа. Также, нет системы изолированной от всего остального. Поэтому программное шифрование гораздо более уязвима для внешних атак и потребляет общие системные ресурсы для выполнения операций. На аппаратную разработку требуются значительные издержки бюджета. Так же реализация аппаратным методом отличается высокой скоростью обработки данных и обращения к памяти. Аппаратная реализация гарантирует целостность по причине неизменности схемы во время работы. С помощью аппаратных шифраторов можно реализовать системы разграничения доступа и защиты информации от несанкционированного

доступа. Выпускаются также и аппаратно-программные модули шифрования, их использование позволяет снизить требования к вычислительным возможностям и для них не придется закупка вспомогательные средства отладки и им не требуется покупка дополнительного оборудования на первоначальных этапах.

Основной целью информационной безопасности считается формирование условий, в которых будет обеспечена качественная и высокоэффективная защита информации от ненамеренного или специального вмешательства, потенциально способного привести к утрате, удалению, изменению, подлогу и иным видам воздействия на информацию.

Доктор Огюст Керкхоффс — языковед и криптограф, в своем труде «Военная криптография» ещё во второй половине девятнадцатого века сформулировал требования многие из которых соблюдают до сих пор. Правило Керкхоффса - это правило разработки криптографических систем, согласно которому секретность является основной причиной хрупкости, и поэтому она может привести к краху системы. Хотя открытость и обеспечивает гибкость системы. Принцип Керкгоффса направлен на то, чтобы сделать безопасность алгоритмов и протоколов независимой от их секретности. Наиболее часто используемые криптографические системы, соответствуют принципу Керкгоффса.

Атаки обычно классифицируются на основе действий, предпринятых злоумышленником. Атака, таким образом, может быть пассивной или активной. Основная цель пассивной атаки — получить несанкционированный доступ к информации. Активная атака включает в себя изменение информации каким-либо образом путем проведения некоторого процесса обработки информации. В зависимости от количества известной информации и её типов доступной злоумышленнику для анализа. Выделяются атаки на основе шифротекста, атаки на основе известного открытого текста, атаки на основе подобранного открытого текста, атаки на основе адаптивно подобранного открытого текста.

В наше время внимание акцентируется на разработке алгоритмов взаимодействия абонентов с применением криптографических средств защиты информации называемых протоколами. Протоколы основаны на наборе правил, регулирующих использование криптографических преобразований. Некоторые виды протоколов:

- протоколы конфиденциальной передачи сообщений;
- протоколы аутентификации и идентификации;
- протоколы распределения ключей;
- протоколы электронной цифровой подписи;
- протоколы обеспечения неотслеживаемости.

В России ГОСТ Р 3412-2018 был принят в качестве стандарта для алгоритмов блочного шифрования с закрытым ключом в 2019 году [1]. Данный стандарт был принят в связи с необходимостью расширить количество утвержденных шифров с различной длиной блоков. Незадолго до этого был введен ГОСТ Р 3413-2015 [2], который описывает режимы работы алгоритмов для n -разрядных блочных шифров. Эти ГОСТы пришли на замену ГОСТ 28147-89 [4]. Аналог алгоритма из ГОСТ 28147-89 [4] представлен в ГОСТ Р 3413-2015 [2], нововведением стало добавление алгоритма с длиной блока 128 бит. Также режимы из ГОСТ 28147-89 [4] перешли в ГОСТ Р 3413-2015 [2] с добавлением нового режима простой замены с зацеплением.

ГОСТ Р 3413-2015 предусматривает шесть режимов работы:

- зашифрование (расшифрование) в режиме простой замены;
- зашифрование (расшифрование) в режиме гаммирования;
- зашифрование (расшифрование) в режиме простой замены с зацеплением;
- зашифрование (расшифрование) в режиме гаммирования с обратной связью;

- зашифрование (расшифрование) в режиме гаммирования с обратной связью по шифротексту;
- формирование имитовставки.

В данной работе алгоритм криптографического преобразования с длиной блока 64 бита по ГОСТ Р 3412-2018 был задан для реализации программно в режиме гаммирования с выработкой имитовставки в рамках имеющихся ресурсов встраиваемого модуля.

1.3 Анализ угроз информационной безопасности

Подход к решению вопросов обеспечения информационной безопасности, в первую очередь, нужно принять во внимание нюансы, связанные с предполагаемыми видами угроз и их источников и ценностью передаваемой информации. Эти параметры [3] определяют выбор СКЗИ, а также стоимость разрабатываемого устройства. В нашем случае задаются следующие параметры:

- информация ограниченного доступа, но не составляет государственную тайну;
- актуальность информации не является долгосрочной;
- вероятность использования методов взятия и анализа ПЭМИН для перехвата информации очень мала;
- вероятность дешифровки информации нарушителем и попыток обнаружения побочных каналов информации связи высока;
- вероятность захвата оборудования и компрометации ключевой информации высока.

Алгоритм криптографического преобразования ГОСТ Р 34.12-18 [1], который был задан для реализации программным способом, является математически стойким криптоалгоритмом и в настоящее время обширно используется и рекомендуется к применению. Надёжность шифрования полностью отвечает принципу Керкхоффа, противник может знать алгоритм шифрования, но ключи держатся в засекреченном виде. При разработке

устройства передачи зашифрованной информации необходимо предусматривать возможность аварийного сброса ключевой информации, а при эксплуатации СКЗИ чаще менять ключи.

2 Сбор и анализ исходных данных для проектирования

2.1 Описание устройства-прототипа

Рассматриваемое устройство-прототип представляет собой малогабаритный встраиваемый модуль, представлен на рисунке 2.1, с разъемом для подключения к техническим средствам, конструктивно выполненный в виде платы с габаритными размерами 30 х 43 х 4 мм. Масса модуля — 10 г. Мощность потребления в рабочем режиме на частоте работы процессора 160 МГц составляет не более 3 Вт. Электропитание модуля должно осуществляться от источника плюс 5 В. Элементная база – импортного производства.

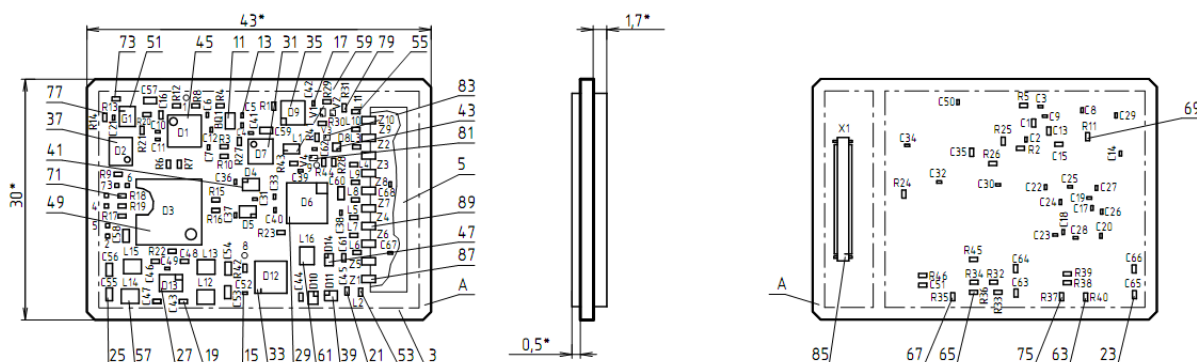


Рисунок 2.1 — Устройство - прототип

Исходное программное обеспечение модуля реализует модемные алгоритмы преобразования сигнала для обеспечения передачи и приёма данных со скоростями: 2, 4, 8 и 16 кбит/с.

Данное устройство было выбрано в качестве макета для реализации в нем алгоритма блочного шифрования с блоком 64 бита по ГОСТ 34.12-18 в целях последующей интеграции его с имеющимся рабочим ПО.

2.2 Описание структурной и электрической схемы

Структурная схема модуля приведена на рисунке 2.2 и в графической части дипломного проекта (ПГУ.100503.С32.ВКР.16ПИ106.001). Обозначение в левом верхнем углу каждого структурного блока, а также наименование элемента соответствует обозначению микросхемы на схеме электрической

принципиальной.

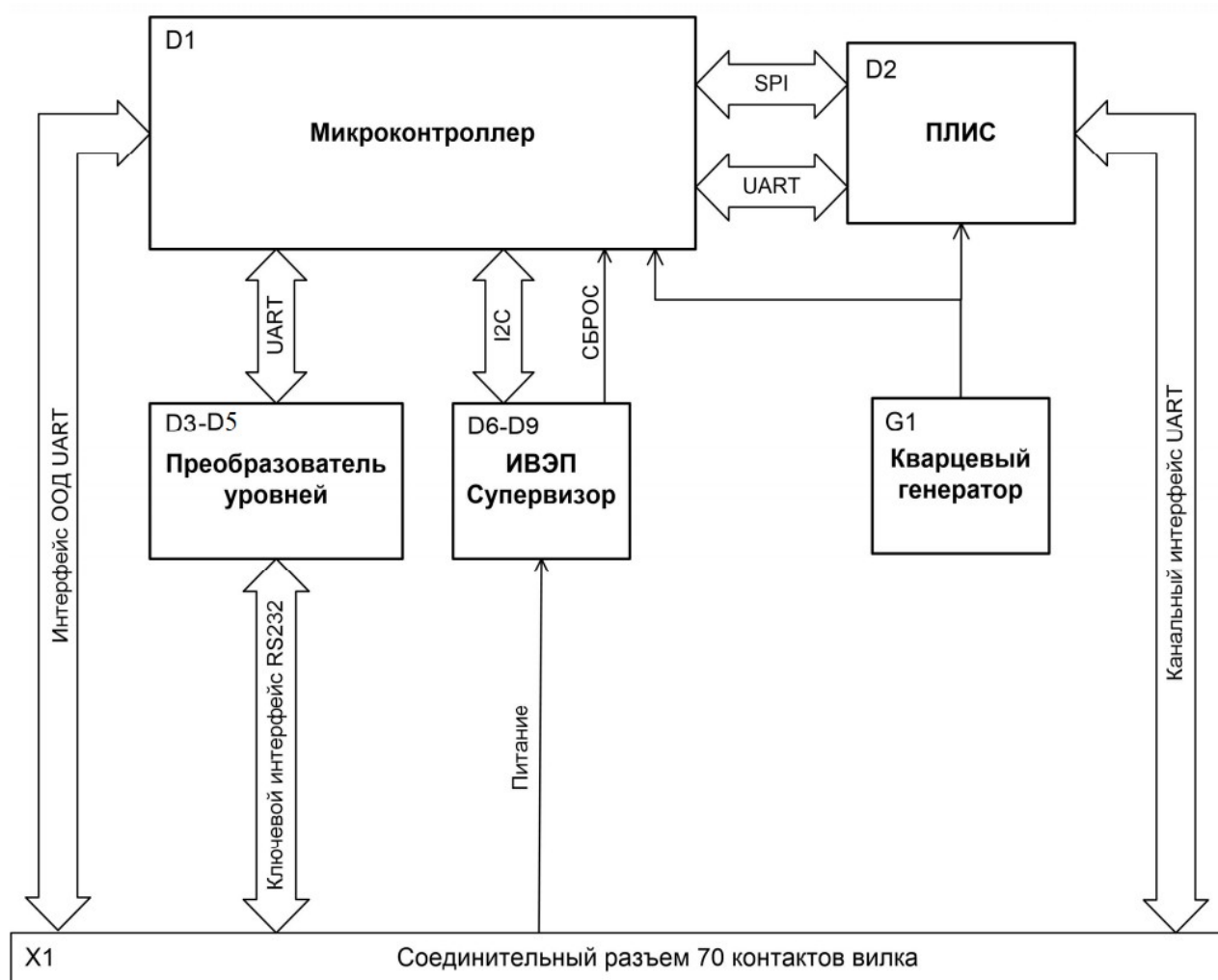


Рисунок 2.2 – Структурная схема модуля

Согласно заданию ВКР аппаратной платформой разработки является микроконтроллер семейства STM32F4 фирмы ST Microelectronics.

В качестве вычислителя используется высокопроизводительный 32-разрядный микроконтроллер STM32F415OG фирмы STMicroelectronics, обозначенный на структурной схеме как элемент D1, который относится к семейству микроконтроллеров STM32F на базе RISC-ядра Arm Cortex-M4.

Данный микроконтроллер имеет 192 Кбайт встроенного ОЗУ, 1 Мбайт памяти программ, а также богатый набор внутренней периферии:

- CAN - последовательная магистраль объединяющая устройств ввода/вывода, датчиков и исполнительных устройств;

- EBI для подключения к микроконтроллеру внешних устройств ввода-вывода;
- EMI для подключения к микроконтроллеру внешних запоминающих устройств, а также для доступа к ним в пространстве памяти данных;
- I²C сетевой последовательный интерфейс со скоростью от 100 до 400 кбит/с;
- IrDA позволяет соединяться с периферийным оборудованием без кабеля при помощи ИК-излучения с длиной волны 880nm;
- LIN - шина локальной соединительной сети;
- SPI 4-х проводный последовательный синхронный интерфейс передачи данных со скоростью до 42 Мбит/с;
- два универсальных асинхронных приемопередатчика (UART) со скоростью до 10,5 Мбит/с;
- четыре модуля последовательного ввода-вывода USART со скоростью до 5,25 Мбит/с, используемые для работы с внешними периферийными устройствами, такими как терминалы, модемы, микросхемы и т.д.;
- USB OTG;
- Brown-out Detect/Reset обеспечивающую снижение напряжения источника питания ниже уровня, необходимого для надежной работы, обнаружение низкого напряжения и сброс при снижении напряжения питания ;
- DMA для прямого доступа к памяти, минуя центральный процессор;
- I²S;
- LCD;
- POR схему, обеспечивающую начальный сброс микроконтроллера при включении питания;
- PWM устройство, для получения прямоугольных импульсов;
- семнадцать 16 и 32-битных таймеров;
- WDT схему контроля над зависанием системы;

- АЦП/ЦАП А/D 13х12бит, D/A 2х12бит;
- напряжение питания от 1,8 до 3,6 В;
- рабочая температура минус 40 до плюс 85С;
- аналоговый аппаратный генератор случайной последовательности (RNG);
- STM32F415 и STM32F417 несут на борту crypto/hash-процессор. Этот процессор обеспечивает аппаратное ускорение AES 128, 192, 256, Triple DES, HASH (MD5, SHA-1). Пропускная способность шифрования по AES-256 достигает 149,33 Мбайт/с.

Максимальная тактовая частота контроллера может достигать 168МГц, что обеспечивает 210 DMIPS/1.25 DMIPS/МГц (Dhrystone 2.1). Ядро Cortex-M4 оснащено блоком с плавающей запятой (FPU) и реализует полный набор DSP-инструкций. У данного процессора низкое энергопотребления (до 0,150 мВт), что делает данную линейку идеальной платформой для ресурсозатратных приложений к примеру обработки данных или голоса (телефоны, модемы, радиостанции и т.п.).

Программируемая логическая интегральная схема, элемент D2 на структурной схеме. Используется для защиты от навязывания средствами опасных сигналов или АЗ. Защита заключается в разрушении информационной последовательности полученной из канала путем подмешивания генерируемой в микроконтроллере ПСП которая передаётся в ПЛИС по интерфейсу SPI после чего происходит подмешивание полученной последовательности в канальную информацию и передача полученной последовательности по интерфейсу SPI в микроконтроллер где происходит снятие подмешенной последовательности вследствие чего любая полученная микроконтроллером информация не содержащая подмешанную последовательность считается ложно навязанной.

Преобразователей уровней IRU1010, элемент D3 на структурной схеме, для обеспечения совместимости логических уровней микроконтроллера и

приемопередатчика

Кварцевый генератор HCMOS/TTL, элемент G1 на структурной схеме, предназначен для синхронизации элементов устройства.

УПЗИ питается от источника электропитания напряжением от 3,2 до 5,5 В.

Узел ИВЭП построен по принципам защиты от ПЭМИН используя разные источники питания для элементов в которых обрабатывается канальная информация от элементов обрабатывающих открытую и ключевую информации. Микросхема D7 TPS62236DRY применена для питания микроконтроллера куда загружается ключевая информация и происходит обработка открытой информации. А микросхема D9 TPS62730DRYT для питания ПЛИС так как она подключена к канальному интерфейсу. Супервизор D8 KP1171СП64, предназначенный для обеспечения контроля над уровнем напряжения на элементы устройства, что требуется для контроля производительности элементов, или сброса параметров устройства при выхода питающих напряжений за допустимые границы.

Все цепи интерфейсов сопряжения, питания и управления выведены на соединительный разъем X1.

2.3 Обоснование выбора среды разработки программного модуля

STM32CubeIDE – первая интегрированная среда разработки от ST, это продвинутая платформа разработки на языке C/C++ с периферийной конфигурацией, генерацией кода, компиляцией кода. Также STM32CubeIDE обладает продвинутым набором инструментов для отладки. Имеет обратную совместимость для поддержки продукта на протяжении долгого времени. Это бесплатное средства разработки из за чего его часто используют в коммерческих организациях. Она основана на платформе Eclipse/CDT и цепочке инструментов GCC для разработки и GDB для отладки. Она позволяет интегрировать сотни существующих плагинов, которые дополняют функции

IDE Eclipse. STM32CubeIDE интегрирует функции настройки STM32 и создания проектов из STM32CubeMX, чтобы предложить универсальный инструмент и сберечь время на установку и разработку. После выбора микроконтроллера или процессора, для выбранного продукта формируется проект и генерируется код инициализации. В любое время во время разработки пользователь имеет возможность вернуться к этапу инициализации или настройки периферийных устройств или же промежуточного программного обеспечения и вернуть код инициализации без какого-либо влияния или изменения собственного ранее написанного кода, что очень помогает при долгой поддержке разработанного продукта или выпуске продукта на другие устройства.

Среда разработки представлена на рисунке 2.3.

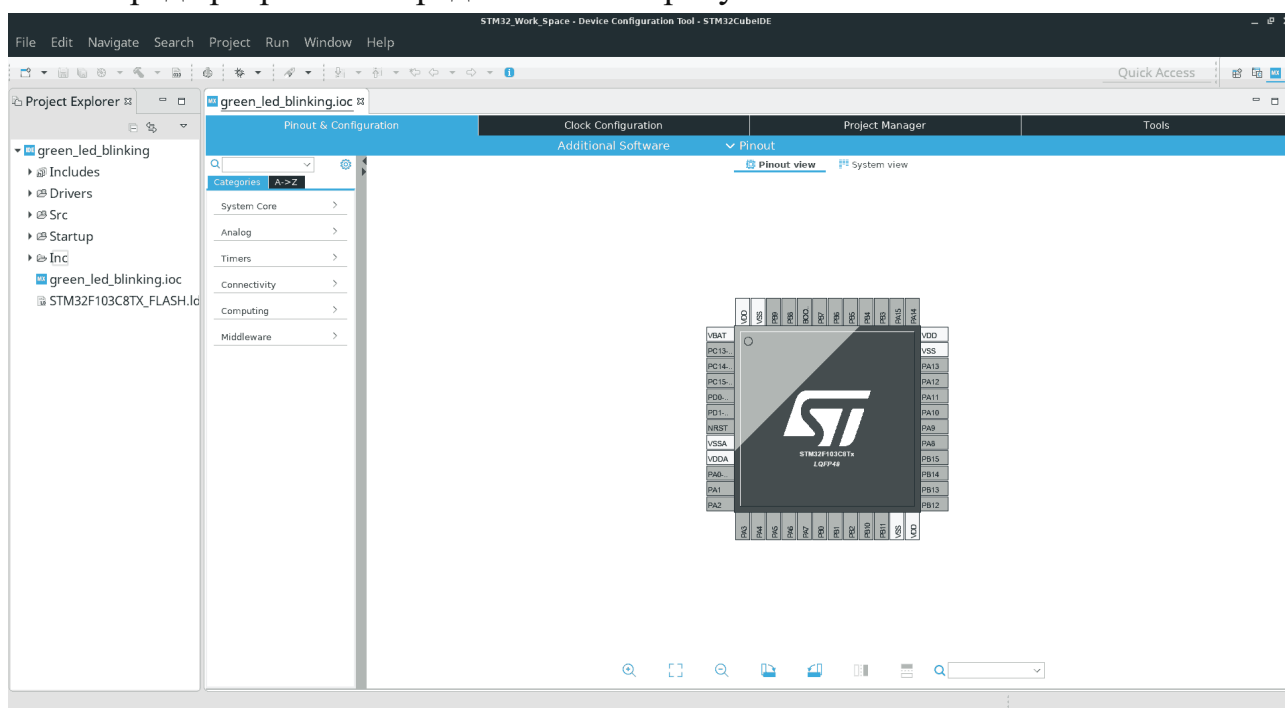


Рисунок 2.3 - STM32CubeIDE

STM32CubeIDE включает в себя анализаторы сборки и стека, которые дают пользователю требуемую информацию о состоянии проекта и используемой памяти. STM32CubeIDE также включает в себя стандартные и расширенные функции отладки, охватывая просмотр регистров ядра процессора, памяти и периферийных регистров, а также просмотр переменных

в реальном времени.

2.4 Анализ устройства-прототипа с точки зрения информационной безопасности

В процессе функционирования любое электронное устройство подвержено утечке информации через различные демаскирующие и побочные излучения.

Использование в системе стойких алгоритмов шифрования нередко не оставляет шансов дешифровать перехваченное сообщение. По причине чего поиск и перехват из каналов ПЭМИН может быть единственной методикой получения информации или же её части до того, как она будет зашифрована.

Нужно отметить, что описываемое устройство-прототип, изображенное на рисунке 2.1, является законченным вычислителем и его назначение определяется прошиваемым программным обеспечением. Далее рассматривается приведенная на рисунке 2.4 структурная и принципиальная схема встраиваемого модуля с точки зрения вопросов утечки передаваемой конфиденциальной информации из-за возможных побочных электромагнитных излучений и наводок.

Устройство является малопотребляющим, что, соответственно, делает его малоизлучающим.

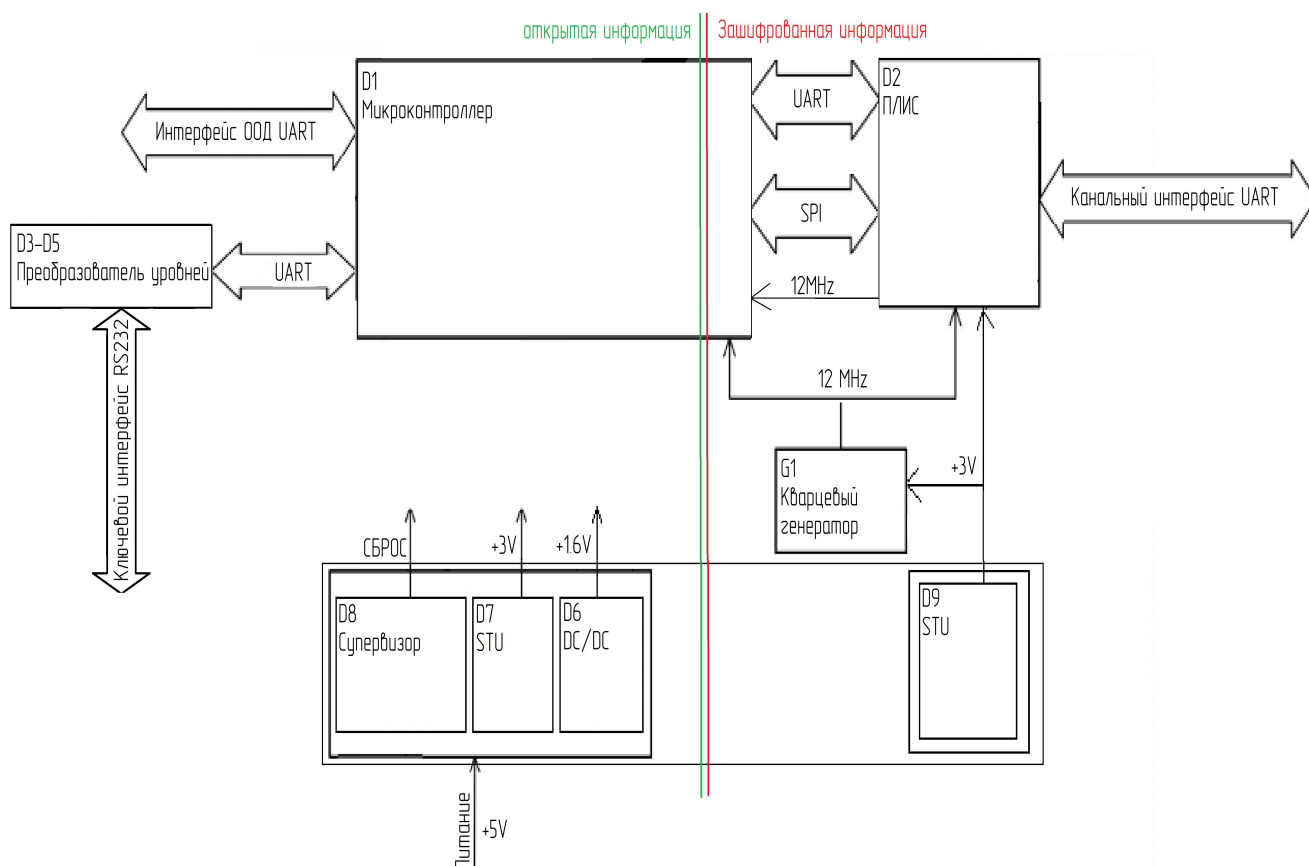


Рисунок 2.4 – Структурная схема модуля с точки зрения ПЭМИН

Структурную схему устройства символически можно поделить на две части – открытую, где циркулирует открытая информация, и закрытую, где циркулирует зашифрованная информация. Открытая информация не должна обнаруживаться в канальных цепях. Для этого в схеме используется раздельное питание элементов. Так ПЛИС, т.е. канальная часть схемы, где циркулирует зашифрованная информация, «запитывается» от одного стабилизатора напряжения (D8), а остальная, «открытая» часть схемы – от другого (D7). Такое разделение, вкупе с элементами фильтрации цепей питания, позволяет исключить возможность воздействия опасных сигналов, несущих открытую и ключевую информацию в цепи первичного электропитания и канальные интерфейсы.

Малогабаритность устройства-прототипа позволяет встраивать его в другие устройства с последующим полным экранированием модуля.

Кроме побочного излучения в результате работы в устройстве

иностранного производства может происходить утечка информации из-за наличия аппаратных закладок.

2.5 Обзор угроз и защиты от аппаратных закладок

Аппаратная закладка — элемент, микросхема или устройство скрытно внедряемое в атакуемую систему или в её инфраструктуру.

АЗ могут запускать серьезные атаки, такие как отключение или изменение функциональности ИМС или утечка конфиденциальной информации.

До сих пор количество зарегистрированных АЗ намного меньше, чем количество программных троянов. Однако беспокойство по поводу АЗ обостряется, существует мнение, что в обозримом будущем будут открыты более сложные АЗ и большее количество функций для них [5].

Чтобы быть скрытыми в ИМС, АЗ обычно спроектированы так, чтобы большую часть времени не работать, а их триггеры связаны с редкими сигналами или событиями [6]. Когда появляется указанный сигнал или событие, цепь нагрузки активируется и реализует вредоносные функции. При грамотном проектировании маловероятно, что редкие сигналы или события возникнут во время моделирования или тестирования конструкции, но могут произойти в течение длительного периода полевой эксплуатации.

Как правило, АЗ можно разделить на два типа: комбинированные и последовательные.

Комбинированные, зависят от одновременного появления набора редких сигналов вызвать неисправность.

Последовательные, зависят от череды редких событий, где используются специализированные закладки запускаемые разным набором редких сигналов и используют узконаправленные вредоносные функции такие как кража ключевой информации или реализация вредоносного кода.

В качестве альтернативы существует систематика АЗ с точки зрения условия и функциональных возможностей.

По условию срабатывания аппаратные закладки можно разделить на аналоговые и цифровые. Первые активируются аналоговыми условиями, такими как температура, задержка или старение устройства, тогда как цифровые запускаются логическими функциями.

С точки зрения функциональных возможностей АЗ, могут вызывать нарушение конфиденциальности через переданный радиосигнал, интерфейс серийного порта или сети международного обмена, может быть атакой по побочному каналу, где утечка информации происходит по цепи питания. Другим типом функциональных возможностей может быть несанкционированное изменение функциональности схемы или атака типа «отказ в обслуживании». Функции аппаратных закладок во многом зависят от того, на каком уровне проектирования они были внедрены.

Обычно в разработке, производстве и применении ИМС участвуют пять сторон. Роль каждой стороны описана ниже:

- литейные предприятия: производители полупроводников (например, TSMC, UMC, IBM), которые заключают контракт с разработчиками на изготовление микросхем;
- инженеры систем на кристалле (далее СнК): проектировать и создавать коммерческие продукты;
- поставщики ИМС: разработка ядер интеллектуальной собственности (например, блоки памяти, ядра DSP) для разработчиков СнК;
- поставщики инструментов EDA: предоставляют инструменты EDA разработчикам СнК и поставщикам, чтобы облегчить проектирование крупномасштабных интегрированных схем, например, Synopsys, Cadence, Xilinx и Altera;
- конечные пользователи: компании или частные лица покупают коммерческие продукты у разработчиков СнК.

Описание угроз между уровнями:

- угроза АЗ между разработчиками СнК и производителями.

В процессе изготовления нет гарантии, что литейные предприятия не вставят в стружку определенный тип АЗ. Чип, изготовленный на литейном производстве, может подвергнуться угрозе со стороны ненадежного персонала или третьей стороны, которым доступен процесс изготовления. Например, троян может быть имплантирован в ИС путем преднамеренного или непреднамеренного изменения уровня примеси или компоновки маски во время отбора образцов или массового производства. Кроме того, у литейных заводов есть свои конфиденциальные инструменты для манипулирования производством микросхем в некоторых злонамеренных целях, и они могут передать генерацию маски третьей стороне, которая имеет возможность злонамеренно включить макросы маски в GDSII;

- угроза АЗ между конечными пользователями и разработчиками СнК.

Конечные пользователи, у которых нет команды разработчиков на уровне микросхем, обеспокоены атаками АЗ на продукты, приобретенные у разработчиков СнК. АЗ могут быть встроены в чипы на этапе проектирования СнК, чтобы обойти средства защиты программного обеспечения и шпионить за пользователями. Конечные пользователи практически не имеют возможности обнаруживать подобные угрозы безопасности на уровне оборудования, что заставляет их больше не доверять традиционно надежным микросхемам. Доказательства наличия АЗ или намеренных дефектов и уязвимостей, скрытых в системах контроля над вооружениями, атомных электростанциях и системах общественного транспорта, были представлены в;

- угроза АЗ между разработчиками СнК и поставщиками блоков микросхем.

При взаимодействии между разработчиками систем на кристалле и поставщиками блоков микросхем первым необходимо обеспечить, чтобы приобретенные функционально законченными блоками микросхем не скрывали вредоносные функциональные блоки, которые впоследствии будет чрезвычайно

трудно обнаружить. Различные типы троянских программ могут быть созданы противником на этапе, предшествующем кремнию. Насколько нам известно, подавляющее большинство АЗ предположительно добавлены в дизайн на уровне RTL, до синтеза, размещения и маршрутизации. Не доверенный инсайдер у поставщиков может легко манипулировать RTL, вставлять вредоносные коды, изменять макросы во время синтеза проекта и даже изменять размещение и маршрут, чтобы освободить место для АЗ. Более того, ненадежный подрядчик, который разрабатывает части спецификации для поставщиков и разработчиков СнК, также имеет возможность включать вредоносные элементы;

- угроза АЗ между поставщиками блоков микросхем (или разработчиками СнК) и поставщиками EDA.

Инструменты EDA широко используются на многих критических этапах проектирования. Программные инструменты, разработанные поставщиками EDA, могут содержать вредоносные коды, которые собирают ценные данные в блоках микросхем / СнК. В исследовании инструментов проектирования EDA проанализировали уязвимости безопасности и было доказано, что логические реализации, выведенные с помощью инструментов EDA, могут делать больше, чем требуется, независимо от надежности команды разработчиков, источника инструментов EDA и поставщиков. Эти нарушения могут быть использованы злоумышленниками для проведения атак.

Мы можем обнаружить, что каждая пара сторон имеет соответствующие методы для борьбы с угрозами АЗ и несколько защитных приемов должны работать вместе, чтобы обеспечить надежность.

В основном, контрмеры против АЗ-атак можно разделить на три категории:

- подходы к обнаружению АЗ;
- подходы к диагностике АЗ;

- подходы к предотвращению АЗ.

Обнаружение АЗ - это процесс, который определяет, существуют ли какие-либо АЗ в цепи. Подходы к раннему обнаружению АЗ обычно основываются на золотой модели, которая представляет ожидаемое поведение системы без возможных закладок [7]. Если подозреваемая система в достаточной степени отклоняется от золотой модели, она будет считаться зараженной. Однако такую модель трудно получить из-за возрастающей сложности элементных баз.

Важно отметить, что все существующие способы обнаружения аппаратных закладок имеют множество особенностей и столько же изъянов. Не существует метода, который мог бы точно обнаружить АЗ. Единственный способ повысить точность - использовать комбинацию различных методов обнаружения.

Общую классификацию методов обнаружения аппаратных закладок можно увидеть на рисунке 2.5 Их можно разделить на деструктивные и недеструктивные.



Рисунок 2.5 Общая классификация методов обнаружения аппаратных закладок

Деструктивные методы представляют собой послойное сканирование чипа после его деметеллизации. Сканирование проводят электронным микроскопом и проверяют слои на наличие изменений в маршрутизации в

схеме или появления сторонних элементов. Для сравнения используется маска чипа предоставленная производителем или золотая модель полученная при прошлом сканировании подобного чипа. Данный метод становится всё дороже и трудозатратнее с каждым годом из-за использования всё большего количества транзисторов, но при этом уменьшении общего размера микросхемы. В итоге анализ одного чипа занимает месяцы. Кроме того после деметаллизирования дальнейшее использование чипа невозможно, а также нельзя быть уверенным что вся партия не имеет внутри встроенных элементов. Таким образом сканированию и анализу снимков придется подвергнуть каждый чип. Этот метод также может быть использован для изучения технологий конкурентов или для проверки функциональности чипа. Именно таким деструктивным способом определяется золотая модель для сравнения.

Для сканирования чипа с возможностью дальнейшего использования используется рентген. Но рентгеновское излучение лежит на энергетической шкале между ультрафиолетом и гамма-излучением в следствии чего транзисторы и иные металлические части микросхемы накапливают заряд, что всё ещё может привести к выходу устройства.

Недеструктивные разделены на:

- мониторинг в процессе работы;
- тестирование до начала работы.

Тестирование до начала работы также можно разделить на:

- функциональное тестирование;
- анализ сторонних каналов.

Мониторинг во время работы как и тестирование до начала работы выполняется для обнаружения неверных, непреднамеренных, неправильных, пороговых, нестандартных, критических значений параметров системы, которые могут означать активацию АЗ. Например, закладка, используемая для сбора информации по каналам ПЭМИН, может вызвать большие скачки напряжения во время простоя, в следствии анализа поведения во время работы

системы можно выявить подозрительное поведение или элемент не использующийся при стандартной работе устройства. Данные мониторы могут использоваться как для обнаружения АЗ в схеме или для остановки системы при работе с системой.

Функциональное тестирование включает в себя написание огромного количества тестов для попытки активации вредоносного элемента системы с целью изучения негативных последствий. Недостатком данного метода является то, что нельзя статистически просчитать все варианты которыми может активироваться АЗ. Вторым недостатком является то, что условия запуска должны быть соблюдены, что существенно ограничивает эффективность применения.

Анализ сторонних каналов включает в себя мониторинг таких каналов как: время задержки, напряжение, ток, энергопотребление, выработку тепла или электромагнитное излучение [8]. Преимущество данного метода то, что даже в случае если АЗ не вызывает изменения в цепи, её присутствие может быть отображено характеристиками, а значит, для работы активация АЗ не потребуется. Недостатком данного метода тестирования является высокая вероятность ошибок из-за погрешности, большого разброса значений, неисправности измерительных приборов, воздействия окружающей среды, тем самым заставляя делать некорректные выводы о наличии АЗ. Таким образом, при проверке необходим статистический подход, чтобы понизить вероятность неверного обнаружения. При данном методе подразумевается, что АЗ обязательно должна выдавать себя через параметр. Также можно отметить, что данный метод тестирования, наиболее эффективен для обнаружения АЗ, которые не приводят к сбою схемы, но передают нарушителю конкретную информацию.

Диагностика АЗ заключается в ориентации АЗ в ИМС с точки зрения их типов, расположения и входных контактов [8], чтобы можно было удалить или замаскировать АЗ из схемы.

Подходы к обнаружению и диагностике АЗ, хотя и являются многообещающими, все же сталкиваются с некоторыми проблемами, такими как идентификация редких узлов, вариации процесса и отклонения в измерениях. Для того чтобы эти подходы были эффективными, ИМС должны разрабатываться с учетом самозащиты. В настоящее время запутывание, заполнение макетов, вставка фиктивных схем и изготовление разделенных схем являются основными методами предотвращения атак АЗ для поставщиков, разработчиков и производителей.

Логическая запутывание - это вероятностное преобразование O , которое преобразует схему источника F в новую схему $O(F)$, которая имеет ту же функциональность, что и F , но в некотором смысле менее понятна. После такого рода трансформации структуру схемы труднее понять. Эта функция поможет сохранить оригинальность схемы и защиту функциональности, который известен только поставщикам блоков микросхем или разработчикам СнК. Но запутывание схем имеет строгие требования.

Заполнение макета облегчает обнаружение и снижает вероятность вставки закладок путем заполнения пустого пространства макета что усложняет или предотвращает изменения положений на изначальной схеме для размещения АЗ.

Повышение вероятности перехода подразумевает уменьшение количества узлов с редким переходом или добавления фиктивных операций, что в свою очередь повысит эффективность обнаружения внедренных закладок или снизит вероятность внедрения из-за последующих частых ложных срабатываний закладки.

Раздельное производство слоёв также может повысить безопасность по причине того что злоумышленник не может определить место внедрения закладки так как ему не известны положения соединений на другом слое. Но в таком случае требуется усложнить расположения контактов так как злоумышленники могут найти информацию о соединении целевыми

контактами и внедрить закладку на основе мест с наименьшей задержкой где чаще всего и будут располагаться элементы связанные с целевым контактом элементы на ином слое.

Другой вид контрмер, называемый предотвращением АЗ, применяется на этапе проектирования современных ИС для увеличения трудностей вставки АЗ, чтобы увеличить эффективность методов обнаружения и диагностики АЗ, или для предотвращения успешной вставки АЗ во время развития ИС. Подходы к обнаружению и диагностике АЗ, хотя и являются многообещающими, все же сталкиваются с некоторыми проблемами, такими как идентификация редких узлов, вариации процесса и отклонения в измерениях. Для того чтобы эти подходы были эффективными, ИМС должны разрабатываться с учетом самозащиты. В настоящее время запутывание, заполнение макетов, вставка фиктивных схем и изготовление разделенных схем являются основными методами предотвращения атак АЗ для поставщиков, разработчиков и производителей.

Также предотвращение можно применять на стороне конечного пользователя с использованием спец средств мониторинга или разрушения информационной последовательности подмешиванием или шифрованием.

Мониторы могут использоваться при тестировании микросхемы и во время работы у конечного пользователя. Как только происходит ненормальная работа схемы, механизм предупреждения может отключить функцию схемы и вызвать другие меры безопасности для предотвращения дальнейших последствий, вызванных аппаратными закладками. Но монитора способного обнаружить любой класс вредоносных модификаций с высокой степенью достоверности, не существует. Также мониторы не эффективны при открытии злоумышленником нового типа закладок.

Принцип функционирования средств разрушения последовательности основан на наличие системы защиты на стыке ЦПОС – ПЛИС, которая заключается в формировании «собственной ПСП», передаваемой в ПЛИС и

изменением последовательности с использованием полученной ПСП с последующим её «снятием» (в ЦПОС).

Для исключения в принимаемом сигнале навязанной помехи к нему также должна добавляться псевдослучайная последовательность, которая формируется в ЦПОС.

3 Разработка программного обеспечения

3.1 Описание функционирования программного модуля

Программный модуль реализует алгоритм блочного шифрования с длиной блока 64 бита по ГОСТ Р 34.12-2018 [1] в режиме гаммирования без обратной связи и имитовставки по ГОСТ Р 34.13-2015 [2] на языке С для аппаратной платформы встраиваемого модуля.

Программный модуль представляет собой функцию, в которой в качестве первого параметра указывается адрес входного массива данных, в качестве второго – адрес выходного массива данных, а в качестве третьего – режим работы: шифрование или дешифрование входных данных.

Передача пакетов осуществляется без квитирования. Процедуры синхронизации информационных потоков выполняются модемным ПО. Достоверность передачи также обеспечивает ПО верхнего уровня. ПО шифратора реализует только симметричный блочный алгоритм шифрования ГОСТ Р 34.12-2018 [1] в режиме гаммирования без обратной связи.

В криптографических протоколах ПО верхнего уровня используется группа имитозащиты. Это связано с тем, что для данных, в отличие от речи, опасны любые искажения. Если требуется наличие имитовставки, то она будет входить в группу имитозащиты. В программном модуле имитовставка реализована как отдельная функция, которая может быть вызвана из ПО верхнего уровня.

Шифрование информации начинается с формирования 64-битной синхропосылки (СП), предваряющей зашифрованные данные, передаваемые по каналу связи. Синхропосылка не является секретной и может свободно передаваться по каналам связи в открытом виде. Для выработки СП используется аналоговый аппаратный генератор случайной последовательности (RNG). Этот источник тока генерирует непрерывный аналоговый шум, который охвачен обратной связью. Регистр сдвига (LFSR) используют

для получения 32-битного случайного числа. Аналоговый источник состоит из нескольких кольцевых генераторов, выходы которых используют операцию XOR.

3.2 Описание функционирования средства защиты от активации закладок

Из возможных программных реализаций защиты от активации закладок для конечного пользователя была выбрана реализация функции разрушения последовательности предотвращающего активацию закладок в следствии большей эффективности в сравнении с мониторами требующих активацию закладки и отключающими функцию схемы при срабатывании так как монитора способного обнаружить любой класс вредоносных модификаций с высокой степенью достоверности, не существует.

На рисунке 3.1 и в графической части дипломного проекта (ПГУ.100503.С32.ВКР.16ПИ106.002) приведена схема, поясняющая принцип функционирования средства защиты.

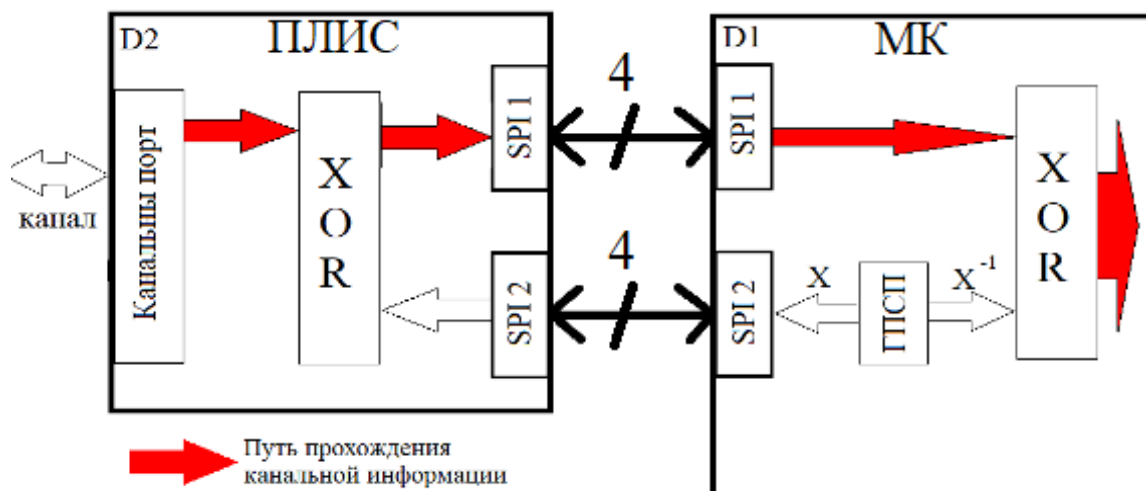


Рисунок 3.1 – Принцип работы средства защиты от активации закладок

На стыке элементов МК и ПЛИС реализуется защита от активации закладок в импортных ИМС. Она заключается в формировании процессором «собственной ПСП», передаваемой в ПЛИС. Эта ПСП добавляется к принимаемому канальному сигналу, а затем снимается в МК.

Прием канальной информации показан на рисунке красной линией. Через

стык SPI1 проходит информация из канала в процессор. Для исключения в принимаемом сигнале навязанной помехи к нему добавляется псевдослучайная последовательность, которая формируется в МК.

Как видно из рисунка, в ПЛИС добавлен сумматор по модулю 2, складывающий принимаемую информацию с ПСП, таким образом, предотвращая активацию закладки или разрушение информации подвергнутой утечке. В процессоре принимаемая информация вновь побайтово складывается с (задержанным байтом x^{-1}) ПСП по модулю 2, что приводит к получению исходной канальной информации.

3.3 Обоснование выбора аппаратной платформы

STM32-это платформа на базе микроконтроллеров STMicroelectronics на базе процессора ARM, различных модулей и периферийных устройств, а также программных решений (IDE) для работы с оборудованием. Решения на основе STM активно используются благодаря производительности микроконтроллера, его успешной архитектуре, низкому энергопотреблению и низкой цене. В настоящее время STM32 уже состоит из нескольких линий для различных целей.

Основные преимущества:

- низкая стоимость;
- большой процент доли на рынке;
- простота использования;
- большой выбор сред разработки;
- наличие большого выбора плат;
- микросхемы взаимозаменяемы — если ресурсов одного микроконтроллера недостаточно, его можно заменить на более мощный, не меняя схему и саму плату;
- высоко оптимизированные среды разработки;

- высокая производительность в сравнении с конкурентами;
- удобная отладка микроконтроллера.

3.4 Описание среды разработки STM32CubeIDE

Эта платформа, созданная на основе Atollic TrueSTUDIO для STM32 выкупленной компанией ST. Таким образом STM32CubeIDE, является слиянием STM32CubeMX и Atollic TrueSTUDIO, сочетая в себе их сильные стороны, сочетая и удобные и практичные первого и функциональность второго.

Встроенный STM32CubeMX позволяет:

- осуществить выбор микроконтроллера STM32;
- настроить тактирование, периферию или дополнительное программное обеспечение;
- создать проект и сгенерировать код инициализации;
- провести компиляцию и отладку кода.

Дополнительные возможности для отладки включают:

- просмотр регистров периферии, ядра и памяти;
- системный анализ и отслеживание в реальном времени (SWV);
- инструменты, позволяющие проводить анализ ошибок процессора.

Осуществляется поддержка ST-LINK и J-Link

При создании проекта можно выбрать микроконтроллер увидеть его краткое описание, некоторые параметры производительности, список внутренних модулей и скачать документацию. Представлено на рисунке 3.2.

STM32CubeIDE используется для разработки программного обеспечения ко всем процессорам, выпускаемым компанией ST Microelectronics.

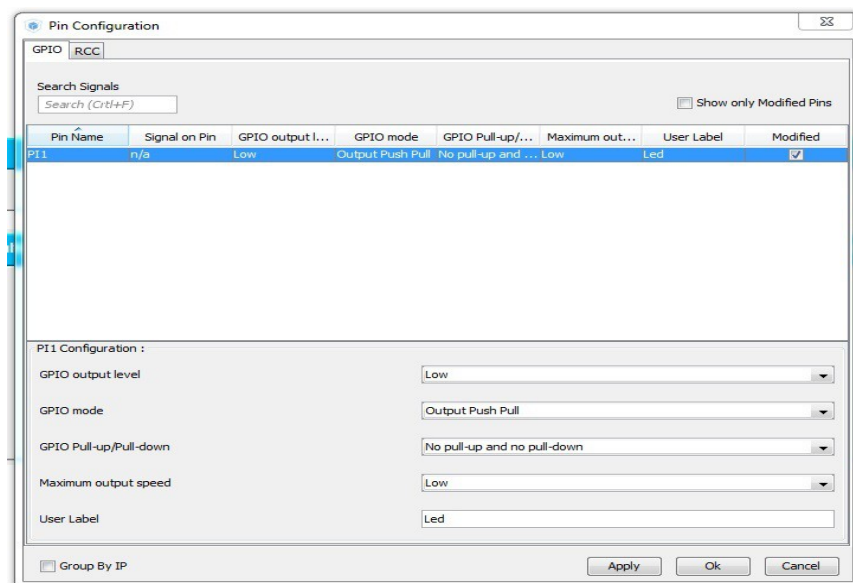


Рисунок 3.4 — Настройка выхода микроконтроля

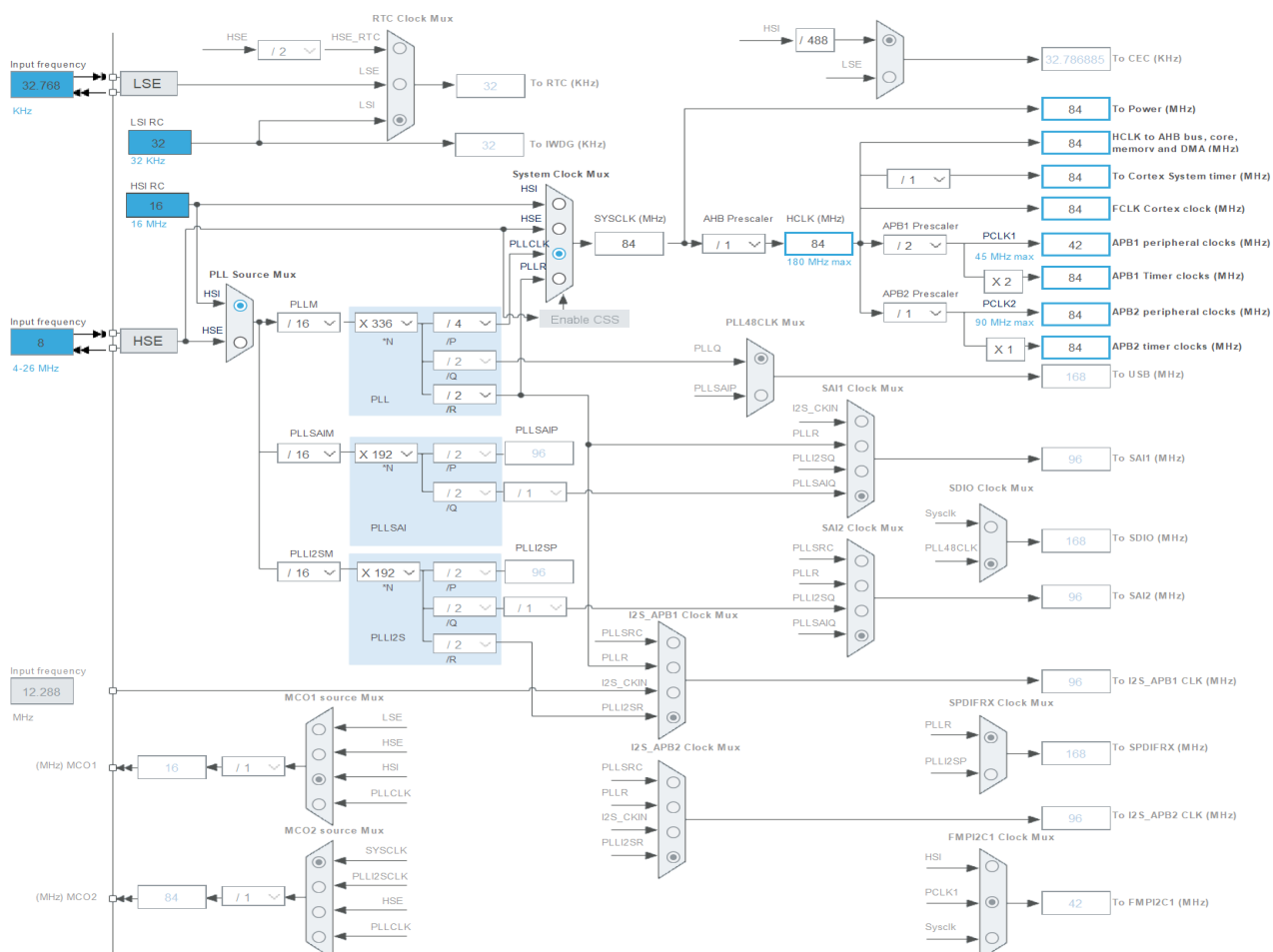


Рисунок 3.5 - Настройка частоты микроконтроллера

На рисунке 3.6 показана среда разработки во время отладки ПО

шифратора.

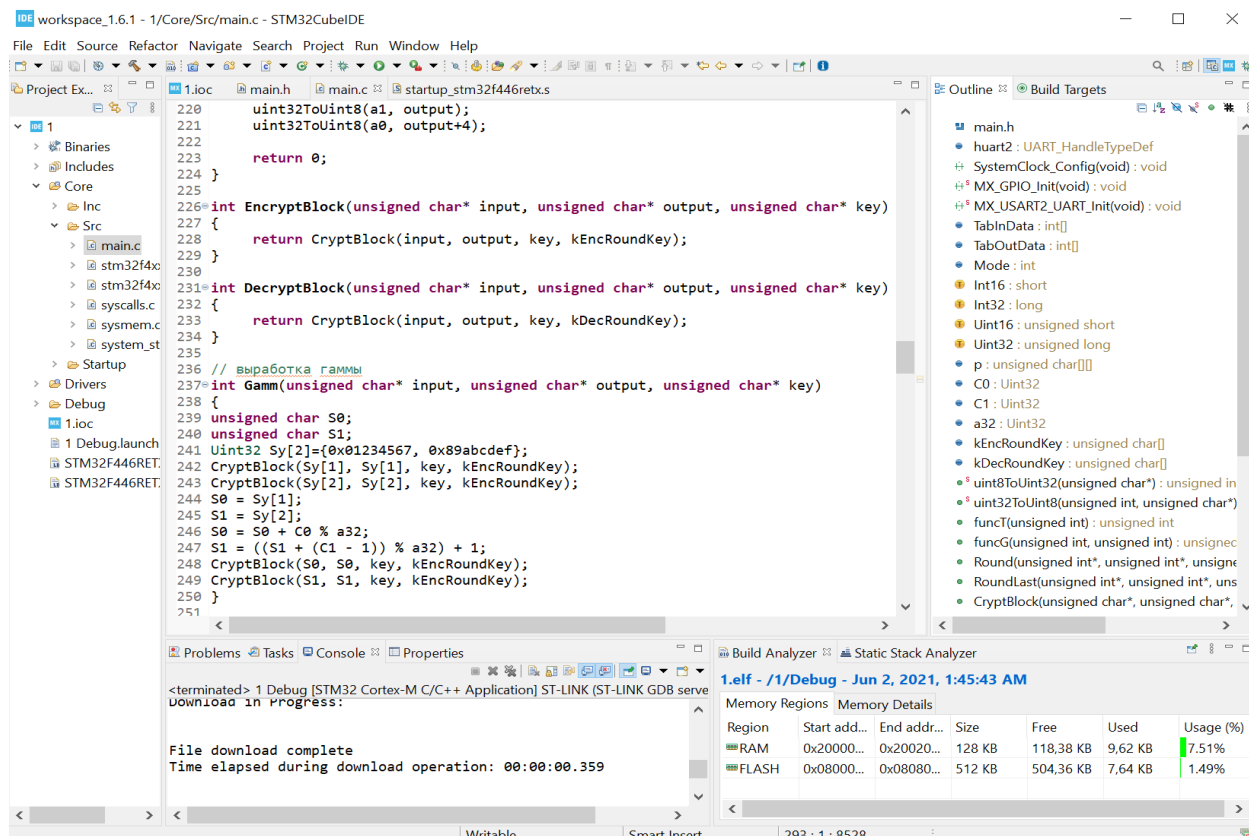


Рисунок 3.6 - Среда разработки

Ее достоинствами являются высокоэффективные компиляторы и удобный пользовательский графический интерфейс ко всем приложениям и пакетам различных производителей.

3.5 Описание алгоритмов криптозащиты и имитозащиты

ГОСТ Р 3412-2018 [1] определяет алгоритм и ГОСТ Р 3413-2015 [2] определяет режимы блочного шифра, которые применяются в криптографических методах обработки и защиты информации, в том числе для обеспечения конфиденциальности, аутентичности и целостности информации при ее передаче, обработке и хранении в автоматизированных системах. Является примером криптосистем, созданных по итерационной схеме Фейстеля.

В данном пункте приняты следующие обозначения:

- A_0 – массив открытых данных;
- A_{III} – массив открытых данных;

- n – число блоков по 64 бита в массиве данных;
- A – блок данных (64 бита);
- A_1 – младшая часть блока данных N ;
- A_0 – старшая часть блока данных N (N состоит из N_1, N_2);
- K_i – элемент ключа (32 бита);
- C – массив из 8 блоков по 4 бита;
- $\ll 11$ – функция циклического сдвига на 11 бит влево;
- $\oplus$ – побитовое сложение;
- T – таблица замен (матрица $8 \cdot 16$);
- S – синхропосылка;
- Z – цикл зашифрования/расшифрования ;
- K – ключ (матрица $8 \cdot 32$).

Алгоритм работает по следующему принципу. На вход подаётся блок открытого текста равный 64 бит и ключ, после чего осуществляется преобразование, называемое шифром. 64-битный блок разделяется на две равные части, над которыми производятся операции раунда, в которую входят: сложение с ключом раунда, подстановка, циклический сдвиг на 15, сложение с половиной блока. Схематический алгоритм работы представлен на рисунке 3.6.

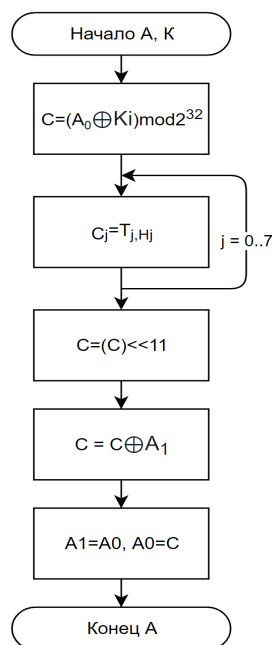


Рисунок 3.6 – Основной шаг криптографического преобразования

Подстановка производится по таблице id-tc26-gost-28147-param-Z представленной в таблице 1.

Таблица 1 - id-tc26-gost-28147-param-Z

Значение	Номер блока							
	8	7	6	5	4	3	2	1
0	1	8	5	7	c	b	6	c
1	7	e	d	f	8	3	8	4
2	e	2	d	5	2	5	2	6
3	d	5	6	a	1	8	3	2
4	0	6	9	8	d	2	9	a
5	5	9	2	1	4	f	a	5
6	8	1	c	6	f	a	5	b
7	3	c	a	d	6	d	c	9
8	4	f	b	0	7	e	1	e
9	f	4	7	9	0	1	e	8
a	a	b	8	3	a	7	4	d
b	6	0	1	e	5	4	7	7
c	9	d	4	b	3	c	b	0
d	c	a	3	4	e	9	d	3
e	b	3	e	2	9	6	0	f
f	2	7	0	c	b	0	f	1

Раундовых ключей в алгоритме – 32. Каждый имеет разрядность 32 бита. Эти ключи формируются из ключа шифрования длиной 256 бита при помощи алгоритма развёртывания ключа. 256 бита – это 8 раундовых ключей. Используемые подряд, но с 25-ого раунда включительно ключи используются в обратном порядке. Развертывание раундовых ключей показано на рисунке 3.7.

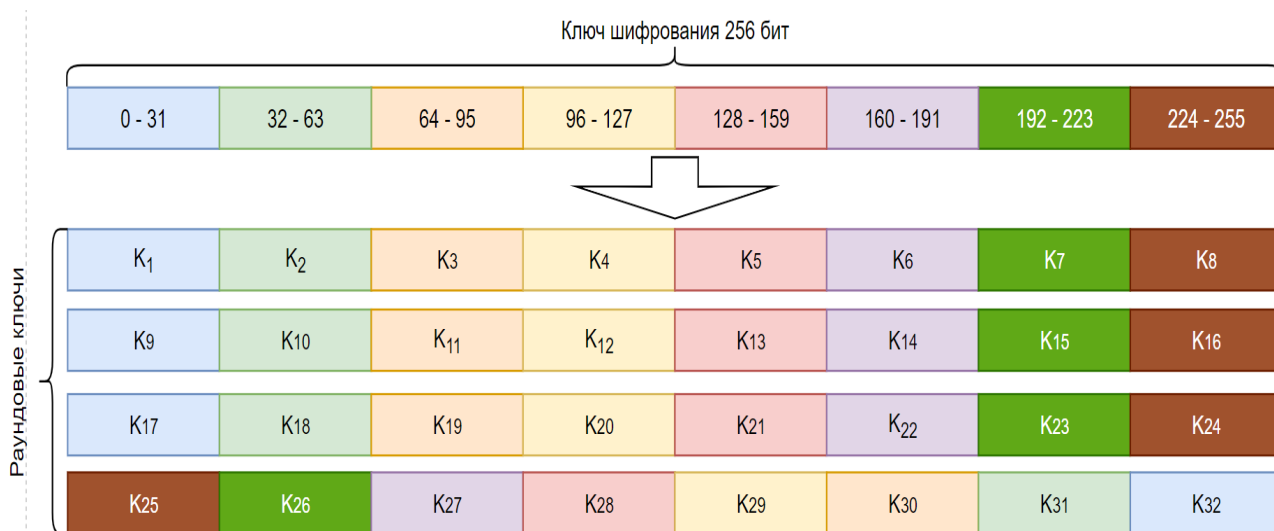


Рисунок 3.7 – Развертывание раундовых ключей

Режим простой замены представляет собой 32 раунда преобразования где на последнем раунде полученные младшая и старшая части блока не меняются местами. Режим простой замены имеет существенные недостатки:

- одна ошибка искажает все биты блока;
- для одинаковых блоков открытого текста блоки шифротекста также одинаковые, что даст информацию нарушителю при криптоанализе.

Таким образом, режим простой замены принято использовать только для преобразования ключевой информации.

Гаммирование – это наложение (снятие) на открытые (зашифрованные) данные криптографической гаммы, то есть последовательности элементов данных, вырабатываемых с помощью некоторого криптографического алгоритма, для получения зашифрованных (открытых) данных. Алгоритм представлен на рисунке 3.8.

Гамма образуется из синхропосылки S — псевдослучайной последовательности, которая изменяется для каждого блока открытого текста с использованием преобразования в режиме простой замены. Полученная гамма складывается по модулю с открытым текстом. Алгоритм режима гаммирования:

- 1) допустим, у нас есть канал синхронизации S — 64-битная псевдослучайная последовательность;

2) S проходит через режим простой замены, при этом разделяясь при этом на два подблока — S_0 и S_1 ;

3) подблоки складываются с фиксированными константами $C_0=1010104$ $C_1=ffffff$ по формулам:

$$S_0 = (S_0 + C_0) \bmod 2^{32}$$

$$S_1 = (S_1 + C_1 - 1) \bmod (2^{32} - 1) + 1$$

и далее сращиваются;

4) Полученная последовательность снова проходит через 32 раунда простой замены полученные блоки сращиваются и полученное будет является шифрующей гаммой для данного блока;

5) открытый текст складывается по модулю 2 с полученной гаммой.

Шаги 3–5 повторяются для каждого блока. Расшифрование выполняется аналогично, вместо блока открытого текста подается блок шифротекста.

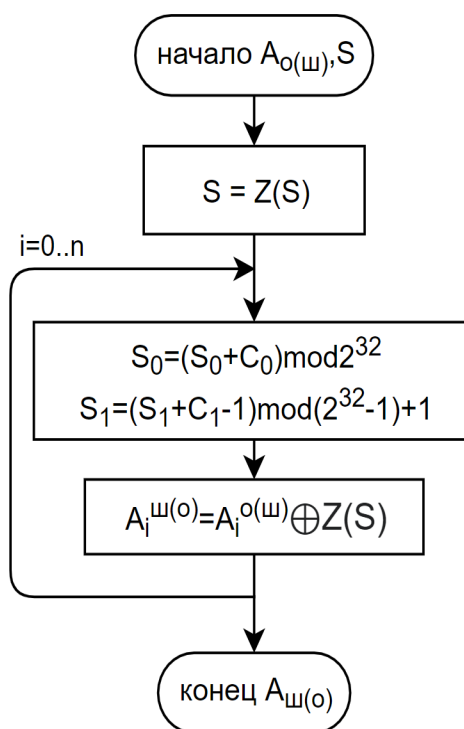


Рисунок 3.8 – Алгоритм шифрования (расшифрования) в режиме гаммирования

Имитовставка — дополнительный блок фиксированной длины, зависящий от исходного текста и ключей. Такой небольшой блок нужен для подтверждения того, что в шифротекст случайно или преднамеренно не были внесены искажения, — то есть для проверки целостности. Этот режим работает следующим образом:

Алгоритм выработки имитовставки представлен на рисунке 3.9.

Рисунок 3.9 – Алгоритм имитовставки

Основной шаг криптопреобразования реализован в коде программы как функция Round.

Базовый цикл шифрования реализован как функция EncryptBlock, а расшифрования DecryptBlock.

3.6 Разработка программного обеспечения

Разработка проводилась на языке Си в среде STM32CubeIDE в соответствии с алгоритмами, описанными в подразделе 3.5.

Полный текст программы представлен в приложении А.

В данной реализации основное преобразование разделено на несколько функций для большей наглядности. На вход функции основного шага подается младшая и старшая часть блока и раундовый ключ. Данная функция производит вызов преобразования G и смену блоков местами. Текст функции:

```
void Round(unsigned int* a1, unsigned int* a0,
unsigned int k)
{
    unsigned int av;
    unsigned int tmp;

    av = *a0;
    tmp = funcG(*a0, k);

    *a0 = *a1 ^ tmp;
    *a1 = av;
}
```

Функция реализующая преобразование G принимает параметрами старшую часть блока и раундовый ключ. Функция проводит сложение части блока с ключом и вызывает функцию нелинейного биективного преобразования T для замены по таблице, после чего проводит сдвиг полученного результата замены на одиннадцать разрядов. Текст функции:

```

unsigned int funcG(unsigned int a, unsigned int k)
{
    unsigned int c = a + k;
    unsigned int tmp = funcT(c);
    unsigned int r = (tmp << 15) | (tmp >> 21);
    return r;
}

```

Функция реализующая преобразование Т принимает параметром часть блока до этого сложенную с раундовым ключом. Функция проводит замену по таблице id-tc26-gost-28147-param-Z представленной в таблице 1. Текст функции:

```

unsigned int funcT(unsigned int a)
{
    unsigned int res = 0;

    res ^= p[0][a & 0x0000000f]; //p – таблица
замены
    res ^= (p[1][((a & 0x000000f0) >> 4)] << 4);
    res ^= (p[2][((a & 0x00000f00) >> 8)] << 8);
    res ^= (p[3][((a & 0x0000f000) >> 12)] << 12);
    res ^= (p[4][((a & 0x000f0000) >> 16)] << 16);
    res ^= (p[5][((a & 0x00f00000) >> 20)] << 20);
    res ^= (p[6][((a & 0x0f000000) >> 24)] << 24);
    res ^= (p[7][((a & 0xf0000000) >> 28)] << 28);

    return res;
}

```

Функция простой замены принимает на вход блок открытого текста и ключ. В теле функции производится разделение блока на две части и основное преобразование. Последний раунд преобразования вызывается отдельно так

как он не требует смену блоков местами. Текст функции:

```
/*
    @brief шифрование простой заменой одного блока (64
бита)
*/
int CryptBlock(unsigned char* input, unsigned char*
output, unsigned char* key, unsigned char* keyIndex)
{
    unsigned int a1 = uint8ToUint32(input);
    unsigned int a0 = uint8ToUint32(input + 4);
    int i;

    for (i = 0; i < 31; ++i)
    {
        Round(&a1, &a0, uint8ToUint32(key +
keyIndex[i]));
    }

    RoundLast(&a1, &a0, uint8ToUint32(key +
keyIndex[31]));

    uint32ToUint8(a1, output);
    uint32ToUint8(a0, output + 4);

    return 0;
}
/*
    @brief последний раунд преобразования без сложения
частей блоков
*/
void RoundLast(unsigned int* a1, unsigned int* a0,
unsigned int k)
{
    unsigned int tmp;

    tmp = funcG(*a0, k);
    *a1 ^= tmp;
}
```

Функция имитовставки принимает на вход блок открытого текста, ключ и индекс раундового ключа. После чего преобразует блок открытого текста в старшую и младшую часть и вызывает функцию основного преобразования.

Текст функции:

```
int ImitPack(unsigned char* input, unsigned char*
output, unsigned char* key, unsigned char* keyIndex)
{
    unsigned int a1 = uint8ToUint32(input);
    unsigned int a0 = uint8ToUint32(input + 4);
    int i;
    for (i = 0; i < 16; ++i)
    {
        Round(&a1, &a0, uint8ToUint32(key +
keyIndex[i]));
    }
    uint32ToUint8(a1, output);
    uint32ToUint8(a0, output + 4);
    return 0;
}
```

Для гаммирования из синхропосылки формируются подблоки для вычислений результирующей гаммы. После чего вызывается функция выработки гаммы производит сложение подблоков с фиксированными константами и вызывает функцию простой замены для получения результирующей гаммы.

```
Uint32 C0 = 0x1010104;    // C0 и C1 - известные из
ГОСТА константы
Uint32 C1 = 0x1010101;
Uint32 a32 = 0xFFFFFFFF; // число 2^32 (для сложения
по модулю)
CryptBlock(Sy[1], Sy[1], key, kEncRoundKey);
CryptBlock(Sy[2], Sy[2], key, kEncRoundKey);
unsigned char S0;
```

```

unsigned char S1;
S0 = Sy[1];
S1 = Sy[2];
// выработка гаммы
int Gamm(unsigned char* input, unsigned char* output,
unsigned char* key)
{
    S0 = S0 + C0 % a32;
    S1 = ((S1 + (C1 - 1)) % a32) + 1;
    CryptBlock(S0, S0, key, kEncRoundKey);
    CryptBlock(S1, S1, key, kEncRoundKey);
}

```

Генератор ПСП для разрушения информационной последовательности реализован вызовом аппаратного генератора случайной последовательности присутствующего в процессоре. В настройках проекта произведена автоматическая регенерация Clock Configuration. В Configuration в DAC выключен аппаратный шум (триггер). Обработчик прерывания по окончании счёта таймера:

```

void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef
*htim6)
{
}

```

Функция генератора случайных чисел:

```

void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef
*htim6)
{
    HAL_DAC_SetValue(&hdac, DAC_CHANNEL_1, DAC_ALIGN_12B_R, (HAL_
RNG_GetRandomNumber(&hrng) & 0x00000FFF));
}

```

Для отладки ПО применялся встроенный отладчик, который был выбран для конкретного типа процессора.

3.7 Тестирование и отладка программного обеспечения

После разработки программного модуля необходимо удостовериться в том, что алгоритм криптографического преобразования ГОСТ Р 3412-18 реализован верно.

Проверка правильности реализации алгоритма криптографического преобразования осуществляется путём тестирования с использованием контрольных примеров.

Набор тестов используется для того, чтобы определить, корректно ли работает программное обеспечение. Тестирование было проведено для проверки правильности шифрования и расшифрования данных.

Программный модуль, разработанный в рамках данного дипломного проекта, был подвергнут тестовой проверке по алгоритму, изложенному выше. Результаты тестовой проверки приведены в Приложении Б.

Правильность реализации алгоритма криптографического преобразования ГОСТ Р 34.12-18 считается доказанным при успешном прохождении всех тестовых проверок.

По результатам тестовой проверки, полученные зашифрованные данные равны контрольным зашифрованным данным, что свидетельствует об успешном прохождении, данным криптографическим обеспечением, тестовой проверки, а также о правильности реализации алгоритма криптографического преобразования.

Для встраиваемых систем критически важно контролировать время выполнения определенных участков кода, чтобы оно не превышало заданную величину. Время обработки может иметь решающее значение для приложения, и скорость вычислений может быть повышена за счет использования новых алгоритмов или применения специальных оптимизаций для особо критичных участков кода. На скорость выполнения также влияют опции компилятора. Однако перед тем, как попытаться улучшить производительность кода, может

потребуется измерить реальную скорость выполнения определенных участков кода.

Известна популярная техника измерения времени выполнения с помощью дергания ножкой GPIO и осциллографа, и это конечно достоверный и реалистичный метод. Однако можно измерить время выполнения кода с помощью других достаточно простых способов, основанных на функционале MCU и средств разработки.

Современные микроконтроллеры содержат в себе встроенный аппаратный регистр, который показывает количество прошедших тактов ядра. Ядро микроконтроллера ARM Cortex-M4 предоставляет для этого регистр с именем CYCLECOUNTER (счетчик циклов).

Следует взять значение CYCLECOUNTER в двух местах кода, время выполнения которого нужно измерить. Затем, зная значение тактовой частоты CPU, можно измерить реальное время выполнения кода. Например, если CPU работает на частоте 100 МГц, то можно полученное значение прошедших тактов поделить на 100000000, и в результате получится измеренное время в секундах.

Особенности метода:

- не требуется модификация кода;
- точность измерения времени составляет 1 такт CPU;
- поддержка метода зависит от применяемого MCU.

Использование CYCLECOUNTER делается следующим образом:

В отладочной сессии открывают окно просмотра регистров CPU Registers через меню View.

В открывшемся окне, среди других регистров микроконтроллера, будет показано значение CYCLECOUNTER, которое равно количеству тактов, прошедших с момента запуска программы.

Для измерения времени выполнения функции, сделайте на ней шаг

отладчика без захода в тело функции (Step Over). Значение CCSTEP покажет разницу между началом выполнения шага и его завершением.

Например: 424 (текущий CYCLECOUNTER) минус 416 (предыдущий CYCLECOUNTER) = 8 (текущий CCSTEP).

CYCLECOUNTER прост в использовании, и дает точное измерение времени. Если нужно узнать время в секундах, то нужно разделить количество прошедших тактов на тактовую частоту CPU.

Производительность процессора составляет 210 MIPS, что соответствует 210 миллионам операций в секунду. В соответствии с техническим заданием, работа криптографического модуля должна занимать не более 20% от этой величины, т.е. не более 42 MIPS. В процессе проверки соответствия ПО требованиям ТЗ было выяснено, что шифрование/расшифрование объема данных 16 кбит осуществляется за 25 MIPS, что обеспечивает требуемое быстродействие ПО шифратора.

Результаты замеров свидетельствуют о том, что требования ТЗ по объему разрабатываемого ПО и быстродействию выполнены полностью.

Объем программного обеспечения модуля составил 8 Кбайт, что соответствует поставленным требованиям. Замеры памяти представлены на рисунке 3.6.

Memory Regions		Memory Details				
Region	Start add...	End addr...	Size	Free	Used	Usage (%)
 RAM	0x20000...	0x20020...	128 KB	118,38 KB	9,62 KB	 7.51%
 FLASH	0x08000...	0x08080...	512 KB	504,36 KB	7,64 KB	 1.49%

Рисунок 3.6 — Память используемая программным модулем

На рисунке 3.7 приведены характеристики разработанного ПО шифратора.

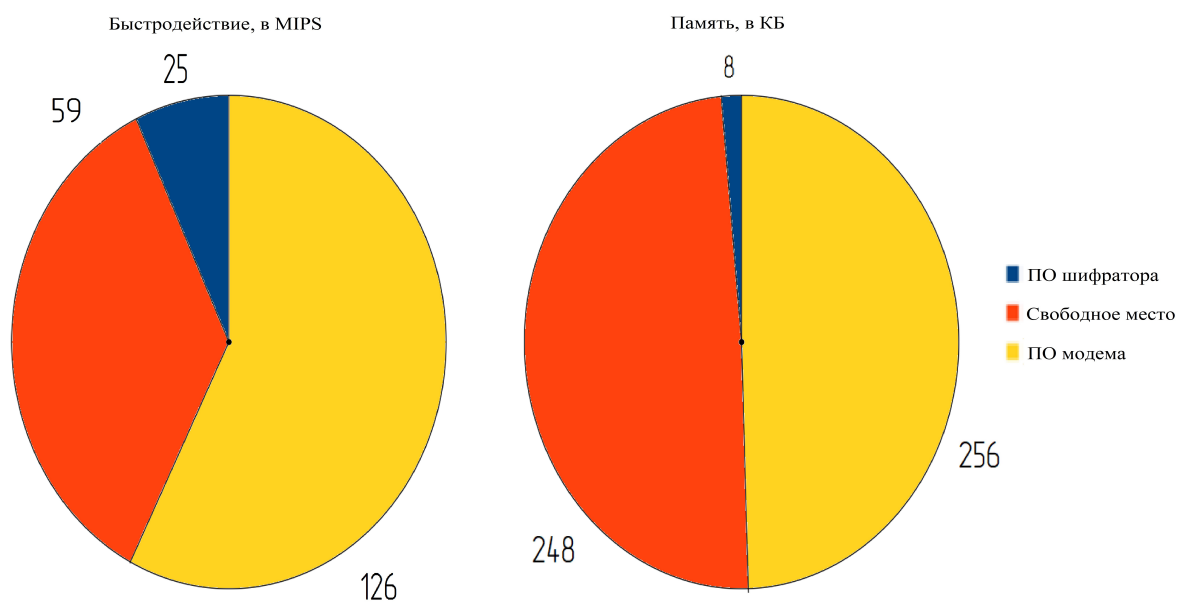


Рисунок 3.7 – Объем внесенных изменений

Результаты замеров свидетельствуют о том, что требования ТЗ по объему разрабатываемого ПО и быстродействию выполнены полностью.

4 Вредные психофизиологические факторы, влияющие на органы зрения специалиста по защите информации. Методы и средства физической культуры, снижающие их воздействие.

Специалисты по защите информации во время выполнения своих профессиональных обязанностей подвержены воздействию опасных и вредных производственных факторов. Каждый работник должен быть проинструктирован о возможном негативном воздействии тех или иных производственных факторов.

Опасные и вредные производственные факторы делятся на физические, химические, биологические и психофизиологические. Последние в свою очередь делятся на физические и нервно-психические перегрузки.

На основе выявленных опасных факторов осуществляется выбор методов и средств обеспечения безопасности, негативно влияющих на органы зрения и присущих тому или иному производственному оборудованию или технологическому процессу.

На рабочем месте специалиста по защите информации обеспечиваются меры защиты от возможного воздействия опасных и вредных факторов. Уровни этих факторов не превышают предельных значений, оговоренных правовыми, техническими и санитарно-эпидемиологическими правилами, указанными в документе СанПин 2.2.2/2.4.1340-03 от 3 июня 2003 года «Гигиенические требования к персональным электронно-вычислительным машинам и организации работы» [9]. Эти санитарные требования направлены на предотвращение неблагоприятного влияния вредных факторов среды и трудового процесса при работе с ПЭВМ на здоровье человека.

Рабочее место разработчика автоматизированной системы включает в себя:

- стол;
- стул;

- персональный компьютер;
- периферийные устройства к ПК.

В процессе работы с компьютером важно соблюдение правильного режима труда и отдыха. Длительное зрительное утомление способно вызвать головные боли, раздражительность, нарушение сна, усталость и ухудшение зрения. Для обеспечения отдыха делают перерывы определенной частоты и длительности.

Первыми симптомами, которые свидетельствуют о зрительном утомлении, являются покраснение и жжение, чувство «песка» в глазах, болезненность в области глаз и лба. Возможно появление конъюнктивита, ощущения инородного тела в глазу, замедление перемещения взгляда с ближних объектов на дальние, затуманивание зрения.

Продолжительность работы с ПК при вводе данных, редактировании программ, чтении информации с экрана не должна превышать 4 ч при 8-часовом рабочем дне. Через каждый час работы необходимо делать перерыв на промежутки времени от 5 до 10 мин, а через 2 ч — на 15 мин.

К монитору должны быть предъявлены следующие требования:

- яркость экрана должна быть не менее 100 кд/м²;
- освещенность поверхности экрана не должна быть более 300 лк;
- яркость бликов на экране — не более 40 кд/м²;
- размер светящейся точки — не более 0,56 (для цветного дисплея);
- контраст изображения знака — не менее 0,8 [10].

Должна предусматриваться возможность регулирования экрана:

- по высоте плюс 3 см;
- по наклону от минус 10° до плюс 20° относительно вертикали;
- в левом и правом направлениях.

Расположение экрана монитора должно обеспечивать удобство зрительного наблюдения в вертикальной плоскости под углом плюс-минус 30° от линии зрения пользователя. Наилучшие зрительные условия и возможность распознавания знаков достигается такой геометрией размещения, когда верхний

край монитора находится на высоте глаз, а взгляд направлен вниз, на центр экрана. Поскольку наиболее благоприятным считается наклон головы вперед приблизительно на 20° от вертикали (при таком положении головы мышцы шеи расслабляются), то экран также должен быть наклоненным назад на 20° от вертикали.

Экран монитора должен располагаться на оптимальном расстоянии от глаз пользователя, но не ближе 600 мм, с учетом размера буквенно-цифровых знаков и символов. Если расстояние от глаз программиста до экрана дисплея составляет от 60 до 80 см, то высота знака должна быть не менее 3 мм, оптимальное соотношение ширины и высоты знака составляет 3 к 4, а расстояние между знаками – от 15% до 20% их высоты. Соотношение яркости фона экрана и символов должно быть от 1 к 2 до 1 к 15. [11]

Некоторые эргономические характеристики рабочего места с ПК приведены на рисунке 4.1.

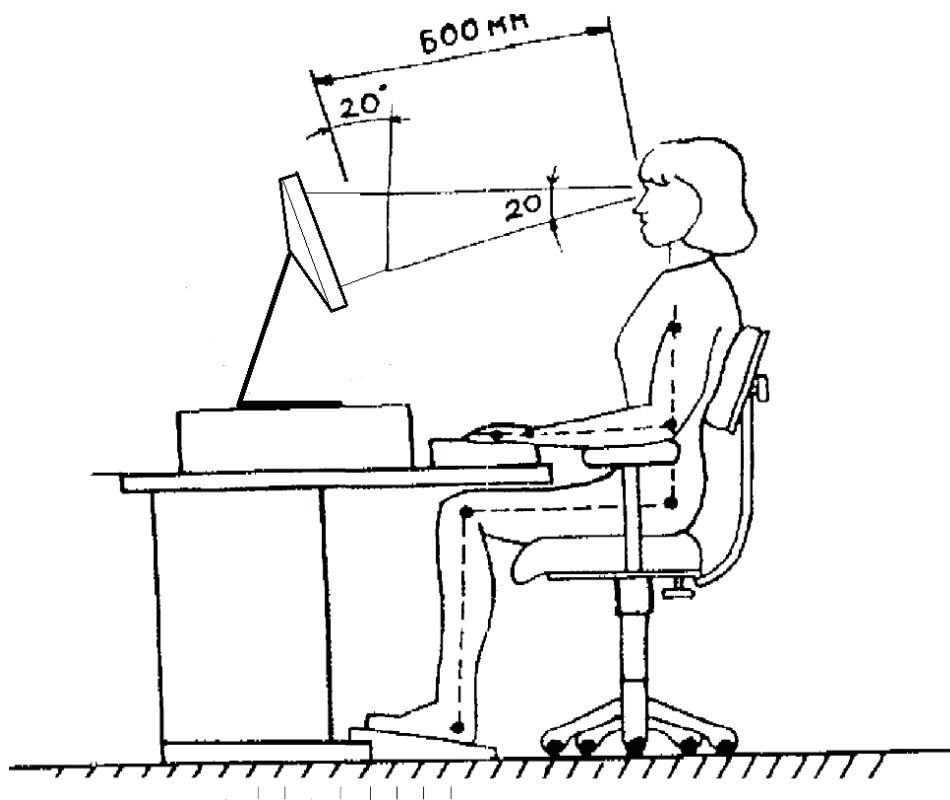


Рисунок 4.1 – Некоторые эргономические характеристики рабочего места с ПК

Немаловажную роль играет освещение рабочего места. Хорошее освещение улучшает условия зрительной работы, ослабляет зрительное и нервное утомление, способствует повышению внимания и улучшению координационной деятельности.

Освещение, отвечающее техническим и санитарно-гигиеническим нормам, называется рациональным. Создание такого освещения на рабочем месте программиста является важной и актуальной задачей.

Рациональное освещение рабочего места является одним из важнейших факторов, влияющих на эффективность трудовой деятельности человека, предупреждающих травматизм и профессиональные заболевания. Правильно организованное освещение создает благоприятные условия труда, повышает работоспособность и производительность труда. Освещение на рабочем месте программиста должно быть таким, чтобы работник мог без напряжения зрения выполнять свою работу [12].

В помещении используется естественное и искусственное освещение. Естественное освещение предполагает проникновение внутрь здания солнечного света через окна и различного типа светопроемы (верхние световые фонари). Естественное освещение часто меняется и зависит от времени года и суток, а также от атмосферных явлений. На освещение влияют местонахождение и устройство здания, величина застекленной поверхности, форма и расположение окон и др.

Естественное освещение — наиболее благоприятное для человека, однако оно не может в полной мере обеспечить необходимую освещенность производственных помещений. Поэтому в практической деятельности широко используют искусственное освещение.

Искусственное освещение выполняется посредством электрических источников света двух видов: ламп накаливания и люминесцентных ламп. На рабочем месте разработчика автоматизированной системы будут использованы люминесцентные лампы, которые по сравнению с лампами накаливания имеют

существенные преимущества:

- по спектральному составу света они близки к дневному, естественному освещению;
- обладают более высоким КПД (в 1,5-2 раза выше, чем КПД ламп накаливания);
- обладают повышенной светоотдачей (в 3-4 раза выше, чем у ламп накаливания);
- имеют более длительный срок службы.

Рациональное искусственное освещение предусматривает равномерную освещенность, без резких изменений и пульсаций, благоприятный спектральный состав света и достаточную яркость. Поэтому для рационального освещения помещения необходимо создать общее и местное освещение, которые в сочетании образуют комбинированное освещение.

Необходимо строго следить за защитой глаз от слепящего действия источников света, не допускать снятия с осветительных приборов защитных стекол и рефлекторов, уменьшения высоты подвеса светильников [13].

Оптимальные условия зрительной работы также зависят от цветового решения производственного помещения. При светлой окраске помещения и той же мощности источников света освещенность повышается на 20-60%, уменьшается резкость теней, световой поток по помещению распределяется более равномерно, снижается яркостный контраст между светильниками и освещаемыми поверхностями.

Поддержание рациональной цветовой гаммы в производственных помещениях достигается правильным выбором осветительных установок, обеспечивающих необходимый световой спектр. В процессе эксплуатации осветительных установок необходимо предусматривать регулярную очистку от загрязнений светильников и остекленных проемов, своевременную замену отработавшей свой срок службы лампы, контроль напряжений питания осветительной сети, регулярную и рациональную окраску стен, потолка,

оборудования [10].

Темная окраска потолка, стен и оборудования способствует неравномерному распределению яркостей в поле зрения работающего. Перевод взгляда с ярко освещенной на слабо освещенную поверхность вынуждает глаз переадаптироваться, что ведет к утомлению зрения и соответственно к снижению производительности труда.

Регулярное зрительное утомление может снизить остроту зрения. Для профилактики заболеваний глаз хорошо зарекомендовала себя специальная гимнастика, предназначенная для снижения усталости органов зрения. Комплекс упражнений не требует специальных условий для выполнения, поэтому может выполняться в рабочем помещении.

Предлагается следующий комплекс упражнений:

- плотно закрытые глаза сильно зажмурить, находиться в таком положении в течение нескольких секунд; открыть глаза и удерживаться от моргания 5-10 секунд;

- перемещать взгляд горизонтально влево и вправо;

- перемещать взгляд вертикально вверх-вниз;

- перемещать взгляд по кругу – по ходу часовой стрелки и в обратную сторону.

- перемещать взгляд по диагонали, скосить глаза в нижний левый угол, а затем по воображаемой прямой линии перевести взгляд в верхний правый угол (по аналогии выполнить упражнение в другую сторону);

- свести глаза к носу, для чего медленно поднести палец к переносице и следить за его кончиком (ещё один вариант этого упражнения — вытянутую руку с карандашом медленно приблизить к носу и следить за движением карандаша);

- смотреть вдаль перед собой в течение 2-3 секунд, затем — на палец или карандаш, удалённый от глаз на расстояние 25-30 см; снова посмотреть вдаль и снова на карандаш, как показано на рисунке 4.2. (повторять упражнение 10-

12 раз);

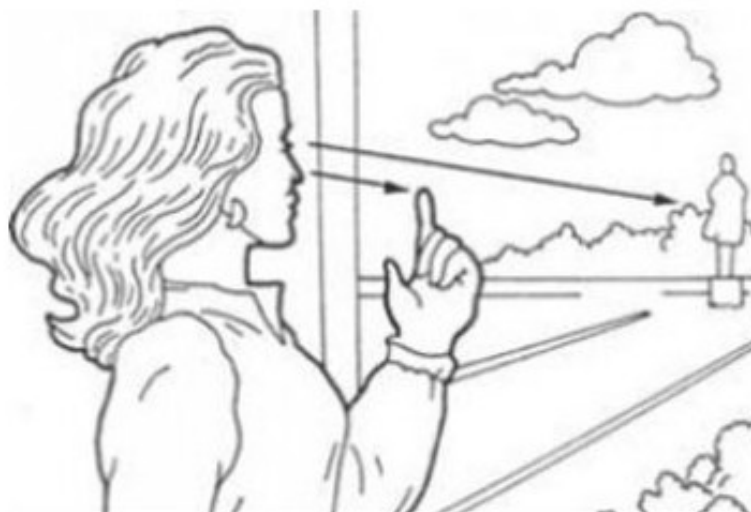


Рисунок 4.2 – Упражнение

- закрепить на оконном стекле метку от 3 до 5 см в диаметре, на высоте по уровню глаз; попеременно смотреть на далёкие объекты за окном и на метку (выполнять 10 или 12 раз);

- медленно и плавно рисовать открытыми глазами горизонтальную восьмёрку от 5 до 7 раз;

- большой палец правой руки вытянуть вперед на высоте, соответствующей уровню глаз и сконцентрировать на нём взгляд на 5 или 6 секунд, плавно отводя руку влево; следить за пальцем, стараясь не поворачивать головы (повторить от 5 до 7 раз, повторить левой рукой);

- часто моргать несколько секунд;

- закрыть глаза и круговыми движениями массировать надбровные дуги и нижнюю половину глазниц; движение выполняется от носа к вискам;

- медленно двигать глазными яблоками закрытых глаз налево, затем направо (повторить 10 раз).

Упражнения могут выполняться в указанном порядке, либо группами [10].

Возможно выполнение и других групп упражнений, одна из которых наглядно представлена на рисунках 4.3 и 4.4.

В данном разделе были описаны мероприятия по снижению зрительного

утомления специалиста по защите информации, работающего за ПК. Необходимо, чтобы рабочее место специалиста полностью соответствовало СанПин 2.2.2/2.4.1340-03 от 3 июня 2003 года «Гигиенические требования к персональным электронно-вычислительным машинам и организации работы» [9].

В помещении, где расположено рабочее место специалиста по защите информации, обеспечено достаточное освещение рабочего места и соблюдены требования к ЭВМ.

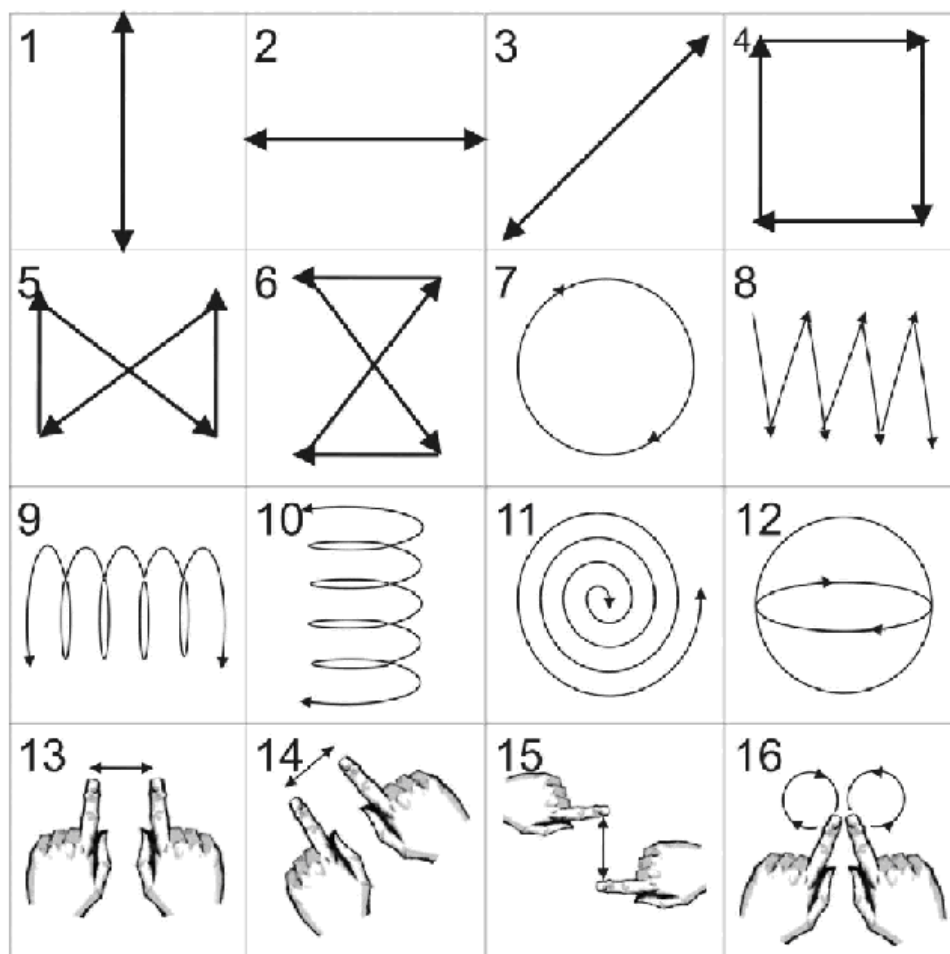


Рисунок 4.3 – Группа упражнений для глаз

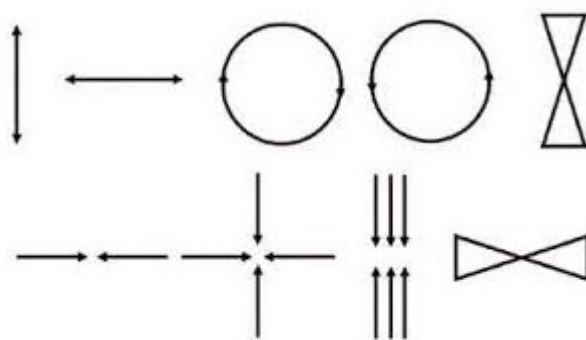


Рисунок 4.4 – Гимнастика для глаз

Предложены мероприятия по снижению зрительного утомления специалиста по защите информации:

- обеспечение продолжительности работы с ПК не более 4 часов при 8- часовом рабочем дне;
- организация перерывов во время работы – каждый час на промежуток времени от 5 до 10 мин, каждые 2 часа – на 15 минут;
- обеспечение соответствия параметров, характеристик и расположения монитора требованиям;
- организация рационального освещения и рациональной цветовой гаммы на рабочем месте;
- информирование сотрудника, работающего за ПК, о правилах проведения гимнастики для поддержания здоровья зрения;
- проверка соответствия рабочего места требованиям СанПин 2.2.2/2.4.1340-03 от 3 июня 2003 года [9].

Заключение

В процессе дипломного проектирования была произведена разработка программного модуля, реализующего алгоритм криптографического преобразования передаваемой информации и функции защиты от активации закладок в импортных интегральных микросхемах.

В процессе работы был проведен анализ структурной схемы устройства-прототипа на предмет возможности реализации заданного алгоритма шифрования и средства защиты от активации закладок в импортных интегральных микросхемах.

Устройство было дополнено программным модулем, реализующим алгоритм криптозащиты по ГОСТ Р 3412-2018 [1], в режимах гаммирования и имитозащиты по ГОСТ Р 3413-2015 [2] – режим гаммирования и выработки имитовставки. Помимо этого был разработан механизм защиты от активации закладок.

Был разработан перечень мероприятий по снижению вредных факторов влияющих на органы зрения специалиста по защите информации.

Результатом работы является программное обеспечение, реализующее криптографическое преобразование и функцию защиты от активации закладок. Оно готово к объединению с имеющимся программным обеспечением устройства.

Таким образом задание на дипломное проектирование выполнено в полном объеме.

Список использованных источников

1. ГОСТ Р 34.13-2018. Информационная технология. Криптографическая защита информации. Блочные шифры. - Москва: Стандартинформ, 2018.-18 с.
2. ГОСТ Р 34.13-2018. Информационная технология. Криптографическая защита информации. Режимы работы блочных шифров. - Москва: Стандартинформ, 2016. - 29 с.
3. Приказ ФСБ РФ "Об утверждении Положения о разработке, производстве, реализации и эксплуатации шифровальных (криптографических) средств защиты информации (Положение ПКЗ-2005)" от 09.02.2005 № 66 // ФСБ РФ. с изм. и допол. в ред. от 12.04.2010
4. ГОСТ 28147-89. Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования. – Москва: ИПК Издательство стандартов, 1996. – 26 с.
5. Bhasin S., Regazzoni F., A survey on hardware trojan detection techniques, in: IEEE International Symposium on Circuits and Systems (ISCAS), 2015, pp. 2021-2024.
6. Bhunia S., Hsiao M.S., Banga M., Narasimhan S., Hardware Trojan Attacks: Threat Analysis and Countermeasures, Proceedings of the IEEE 102 (8) (2014) 1229-1247.
7. Narasimhan S., Bhunia S., Hardware Trojan Detection, Introduction to Hardware Security and Trust (2012) 51-57.
8. J. He, Y. Zhao, X. Guo, Y. Jin. Hardware Trojan Detection Through Chip Free Electromagnetic Side Channel Statistical Analysis. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2017.
9. Постановление Главного государственного санитарного врача Российской Федерации от 13 июня 2003 г., № 158, г. Москва. О введении в действие санитарно-эпидемиологических правил и нормативов САНПИН 2.2.2/2.4.1340-03. - URL: <http://www.rg.ru/2003/06/21/134.html>. Дата обращения:

25.05.2021

10. Безопасность жизнедеятельности: Учебник / Под ред. проф. Э. А. Арустамова. 10-е изд., перераб. и доп. — М.: Издательско-торговая корпорация «Дашков и К*», 2006. — 476 с.

11. Зарядка для глаз после работы за компьютером - URL: http://ps-teach.ru/post_1385219890.html. Дата обращения: 25.05.2021.

12. Дементий Л.В., Юсина А.Л. Охрана труда в автоматизированном производстве. Обеспечение безопасности труда – Краматорск: ДГМА, 2007. – 300 с.

13. Самгин Э.Б. Освещение рабочих мест. - М.: МИРЭА, 1989. - 27 с

Приложение А

(обязательное)

Текст программы на исходном языке

```
int TabInData[1024];      // Таблица входных данных
int TabOutData[1024];     // Таблица выходных данных
int Mode;                 // Режим работы: 0 - шифрование; 1 -
расшифрование

    //знаковые типы
    typedef short          Int16;          // от -32768          до
+32767          = int
    typedef long           Int32;          // от -2147483648 до
+2147483647
    //беззнаковые типы
    typedef unsigned short  Uint16;        // от 0 до +65535
          = unsigned int
    typedef unsigned long   Uint32;        // от 0 до
+2147483647

    /*
@brief Функция формирования СП
*/
void GOST_GenerateSP (void)
{
    HAL_TIM_Base_Start_IT(&htim6);
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim6)
{
}
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim6)
{
    HAL_DAC_SetValue(&hdac, DAC_CHANNEL_1, DAC_ALIGN_12B_R,
(HAL_RNG_GetRandomNumber(&hrng) & 0x00000FFF);
}
}

    // Тестовые Открытые данные

    // @brief Таблица подстановки id-tc26-gost-28147-param-Z
unsigned char p[8][16] =
{
    {0xc, 0x4, 0x6, 0x2, 0xa, 0x5, 0xb, 0x9, 0xe, 0x8, 0xd,
0x7, 0x0, 0x3, 0xf, 0x1},
    {0x6, 0x8, 0x2, 0x3, 0x9, 0xa, 0x5, 0xc, 0x1, 0xe, 0x4,
0x7, 0xb, 0xd, 0x0, 0xf},
    {0xb, 0x3, 0x5, 0x8, 0x2, 0xf, 0xa, 0xd, 0xe, 0x1, 0x7,
0x4, 0xc, 0x9, 0x6, 0x0},
    {0xc, 0x8, 0x2, 0x1, 0xd, 0x4, 0xf, 0x6, 0x7, 0x0, 0xa,
```

```

0x5, 0x3, 0xe, 0x9, 0xb},
    {0x7, 0xf, 0x5, 0xa, 0x8, 0x1, 0x6, 0xd, 0x0, 0x9, 0x3,
0xe, 0xb, 0x4, 0x2, 0xc},
    {0x5, 0xd, 0xf, 0x6, 0x9, 0x2, 0xc, 0xa, 0xb, 0x7, 0x8,
0x1, 0x4, 0x3, 0xe, 0x0},
    {0x8, 0xe, 0x2, 0x5, 0x6, 0x9, 0x1, 0xc, 0xf, 0x4, 0xb,
0x0, 0xd, 0xa, 0x3, 0x7},
    {0x1, 0x7, 0xe, 0xd, 0x0, 0x5, 0x8, 0x3, 0x4, 0xf, 0xa,
0x6, 0x9, 0xc, 0xb, 0x2}
};

    Uint32 C0 = 0x1010104;    // C0 и C1 - известные из ГОСТа
константы
    Uint32 C1 = 0x1010101;
    Uint32 a32 = 0xFFFFFFFF; // число 2^32 (для сложения по
модулю)

    //Uint32 Sy[2] = { 0x01234567, 0x89abcdef }; //тестовая
синхропосылка
    CryptBlock(Sy[1], Sy[1], key, kEncRoundKey);
    CryptBlock(Sy[2], Sy[2], key, kEncRoundKey);
    unsigned char S0;
    unsigned char S1;
    S0 = Sy[1];
    S1 = Sy[2];

    /* Uint8 a_0[4]; // Правая половина блока
    Uint8 a_1[4]; // Левая половина блока
    Uint8 G[4];

    int i;
    // Делим 64-битный исходный блок на две части
    for (i = 0; i < 4; i++)
    {
        a_0[i] = a[4 + i];
        a_1[i] = a[i];
    } */

    //используемый байт ключа при шифровании
    unsigned char kEncRoundKey[32] =
    {
        0, 4, 8, 12, 16, 20, 24, 28, 0, 4, 8, 12, 16, 20, 24, 28,
        0, 4, 8, 12, 16, 20, 24, 28, 28, 24, 20, 16, 12, 8, 4, 0
    };

    //используемый байт ключа при расшифровании
    unsigned char kDecRoundKey[32] =
    {
        0, 4, 8, 12, 16, 20, 24, 28, 28, 24, 20, 16, 12, 8, 4, 0,
        28, 24, 20, 16, 12, 8, 4, 0, 28, 24, 20, 16, 12, 8, 4, 0
    };

```

```

/*
@brief Конвертирует массив байт в int32

@param input массив из 4 байт
@return int32 число
*/
static unsigned int uint8ToUint32(unsigned char* input)
{
    unsigned int r = ((input[3]) | (input[2] << 8) | (input[1]
<< 16) | (input[0] << 24));
    return r;
}

/*
@brief Конвертирует int32 в массив байт

@param input int32 число
@param output массив из 4 байт
*/
static void uint32ToUint8(unsigned int input, unsigned char*
output)
{
    int i;
    for (i = 0; i < 4; ++i)
    {
        output[3 - i] = ((input >> (8 * i)) & 0x000000ff);
    }
}

/*
@brief Нелинейное биективное преобразование (преобразование T)
*/
unsigned int funcT(unsigned int a)
{
    unsigned int res = 0;

    res ^= p[0][a & 0x0000000f];
    res ^= (p[1][((a & 0x000000f0) >> 4)] << 4);
    res ^= (p[2][((a & 0x00000f00) >> 8)] << 8);
    res ^= (p[3][((a & 0x0000f000) >> 12)] << 12);
    res ^= (p[4][((a & 0x000f0000) >> 16)] << 16);
    res ^= (p[5][((a & 0x00f00000) >> 20)] << 20);
    res ^= (p[6][((a & 0x0f000000) >> 24)] << 24);
    res ^= (p[7][((a & 0xf0000000) >> 28)] << 28);

    return res;
}

/*
@brief преобразование включает в себя:

```

32

- 1) сложение правой части блока с итерационным ключом по модулю
- 2) нелинейное биективное преобразование
- 3) сдвиг влево на одиннадцать разрядов

```

@param a правая часть блока 32бита
@param k ключ раунда 32бита
@param c промежуточная величина
@param r результат преобразования
*/
unsigned int funcG(unsigned int a, unsigned int k)
{
    unsigned int c = a + k;
    unsigned int tmp = funcT(c);
    unsigned int r = (tmp << 15) | (tmp >> 21);

    return r;
}

/*
@brief раунды преобразования 0-30

@param a1 правая часть блока 32бита
@param a2 левая 32бита
@param k ключ раунда 32бита
*/
void Round(unsigned int* a1, unsigned int* a0, unsigned int k)
{
    unsigned int av;
    unsigned int tmp;

    av = *a0;
    tmp = funcG(*a0, k);

    *a0 = *a1 ^ tmp;
    *a1 = av;
}

/*
@brief последний раунд преобразования без сложения частей
блоков
*/
void RoundLast(unsigned int* a1, unsigned int* a0, unsigned
int k)
{
    unsigned int tmp;

    tmp = funcG(*a0, k);
    *a1 ^= tmp;
}

/*

```

```

@brief шифрование простой заменой одного блока (64 бита)
*/
int CryptBlock(unsigned char* input, unsigned char* output,
unsigned char* key, unsigned char* keyIndex)
{
    unsigned int a1 = uint8ToUint32(input);
    unsigned int a0 = uint8ToUint32(input + 4);
    int i;

    for (i = 0; i < 31; ++i)
    {
        Round(&a1, &a0, uint8ToUint32(key + keyIndex[i]));
    }

    RoundLast(&a1, &a0, uint8ToUint32(key + keyIndex[31]));

    uint32ToUint8(a1, output);
    uint32ToUint8(a0, output + 4);

    return 0;
}
//зашифрование блока
int EncryptBlock(unsigned char* input, unsigned char* output,
unsigned char* key)
{
    return CryptBlock(input, output, key, kEncRoundKey);
}
//расшифрование блока
int DecryptBlock(unsigned char* input, unsigned char* output,
unsigned char* key)
{
    return CryptBlock(input, output, key, kDecRoundKey);
}

// выработка гаммы
int Gamm(unsigned char* input, unsigned char* output, unsigned
char* key)
{
    S0 = S0 + C0 % a32;
    S1 = ((S1 + (C1 - 1)) % a32) + 1;
    CryptBlock(S0, S0, key, kEncRoundKey);
    CryptBlock(S1, S1, key, kEncRoundKey);
}

//гаммирование summ
int GOST_test1(Uint32 Sy, Uint32 T)
{
    //Uint32 Sy[2] = { 0x01234567, 0x89abcdef };
    //Uint32 T[2] = { 0xfa14fc9d, 0xa39ac2da };
    for (int i = 0; i < 2; i++)
    {

```

```

        T[i] = T[i] ^ Sy[i];
    }
}

int ImitPack(unsigned char* input, unsigned char* output,
unsigned char* key, unsigned char* keyIndex)
{
    unsigned int a1 = uint8ToUInt32(input);
    unsigned int a0 = uint8ToUInt32(input + 4);
    int i;
    for (i = 0; i < 16; ++i)
    {
        Round(&a1, &a0, uint8ToUInt32(key + keyIndex[i]));
    }

    uint32ToUInt8(a1, output);
    uint32ToUInt8(a0, output + 4);

    return 0;
}

```

Приложение Б

(обязательное)

Результаты тестирования программного обеспечения

Тест 1

Используются:

Открытый текст To-1

Заполнение КЗУ КЗУ- I

Синхропосылка s1

Зашифрование

Предполагаемый результат:

Такт1 7A F7 60 9A 80 CB 81 AF

Такт2 12 BB 6A D7 CD 9B 60 43

На практике получен результат, представленный на рисунке Б1.

Expression	Value	Expression	Value
OpenData	<array>	EncData	<array>
[0]	0x4F5CA2AE	[0]	0x7AF7609A
[1]	0x30A9F0C9	[1]	0x80CB81AF
[2]	0xE24A75D8	[2]	0x12BB6AD7
[3]	0xF4ABEF98	[3]	0xCD9B6043

Рисунок Б1 – Результаты теста 1

Тест 2

Используются:

Открытый текст To-1

Заполнение КЗУ КЗУ-1

Синхропосылка sl

Расшифрование

Предполагаемый результат:

Такт1 4F 5C A2 AE 30 A9 F0 C9

Такт2 E2 4A 75 D8 F4 AB EF 98

На практике получен результат, представленный на рисунке Б2.

Expression	Value	Expression	Value
EncData	<array>	DecData	<array>
[0]	0x7AF7609A	[0]	0x4F5CA2AE
[1]	0x80CB81AF	[1]	0x30A9F0C9
[2]	0x12BB6AD7	[2]	0xE24A75D8
[3]	0xCD9B6043	[3]	0xF4ABEF98
[4]	0x00000000	[4]	0x00000000
[5]	0x00000000	[5]	0x00000000
[6]	0x00000000	[6]	0x00000000
[7]	0x00000000	[7]	0x00000000
[8]	0x00000000	[8]	0x00000000
[9]	0x00000000	[9]	0x00000000
[10]	0x00000000	[10]	0x00000000
[11]	0x00000000	[11]	0x00000000
[12]	0x00000000	[12]	0x00000000
[13]	0x00000000	[13]	0x00000000
[14]	0x00000000	[14]	0x00000000
[15]	0x00000000	[15]	0x00000000
[16]	0x00000000	[16]	0x00000000
[17]	0x00000000	[17]	0x00000000
[18]	0x00000000	[18]	0x00000000
[19]	0x00000000	[19]	0x00000000
[20]	0x00000000	[20]	0x00000000
[21]	0x00000000	[21]	0x00000000
[22]	0x00000000	[22]	0x00000000
[23]	0x00000000	[23]	0x00000000
[24]	0x00000000	[24]	0x00000000
[25]	0x00000000	[25]	0x00000000
[26]	0x00000000	[26]	0x00000000
[27]	0x00000000	[27]	0x00000000
[28]	0x00000000	[28]	0x00000000
[29]	0x00000000	[29]	0x00000000
[30]	0x00000000	[30]	0x00000000
[31]	0x00000000	[31]	0x00000000

Рисунок Б2 – Результаты теста 2

Тест 3

Используются:

Открытый текст To-2

Заполнение КЗУ КЗУ-2

Синхропосылка s1

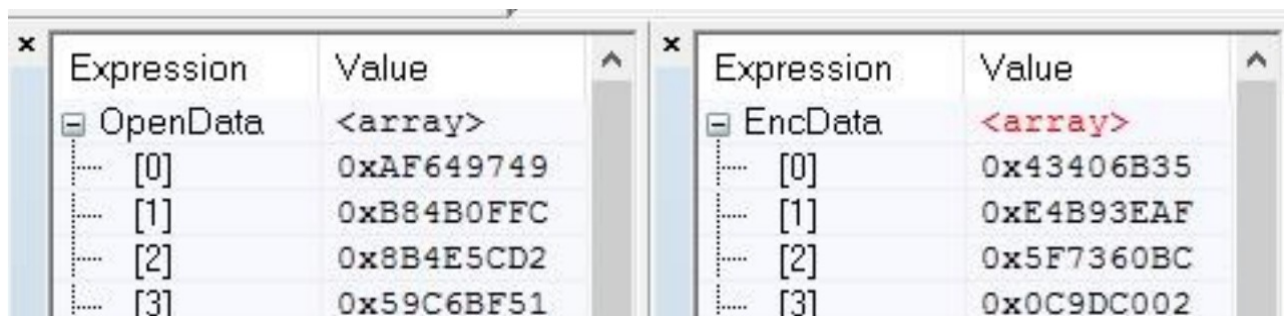
Зашифрование

Предполагаемый результат:

Такт1 43 40 6B 35 E4 B9 3E AF

Такт2 5F 73 60 BC 0C 9D C0 02

На практике получен результат, представленный на рисунке Б3.



Expression	Value
OpenData	<array>
[0]	0xAF649749
[1]	0xB84B0FFC
[2]	0x8B4E5CD2
[3]	0x59C6BF51

Expression	Value
EncData	<array>
[0]	0x43406B35
[1]	0xE4B93EAF
[2]	0x5F7360BC
[3]	0x0C9DC002

Рисунок Б3 – Результаты теста 3

Тест 4

Используются:

Открытый текст To

Заполнение КЗУ КЗУ

Синхропосылка s1

Зашифрование

Предполагаемый результат:

Такт 1 7A F7 60 9A 80 CB 81 AF

Такт 2 12 BB 6A D7 CD 9B 60 43

Такт 3 43 40 6B 35 E4 B9 3E AF

Такт 4 5F 73 60 BC 0C 9D C0 02

Такт 5 B8 5F 3A B4 65 32 B5 0B

Такт 6 EB 66 60 A8 A3 32 5A 2E

Такт 7 8C AA 37 86 67 08 73 7E

Такт 8 5E 4B DF AB AD 93 23 EE

Такт 9 BB 95 EC 35 79 33 FE 5F

Такт 10 82 E4 0F D5 79 7C 7E 78

Такт 11 B3 EC CC 5D D0 02 D5 47

Такт 12 A8 87 4E E9 28 0E 06 1B

Такт 13 D3 80 1A 0D 1F DB 40 9F

Такт 14 7C B8 C2 43 50 18 5B 67

Такт 15 2B 66 A9 8A 49 92 97 35

Такт 16 C8 86 A2 75 3F 8B 3F 24

На практике получен результат, представленный на рисунке Б4.

Expression	Value	Expression	Value
OpenData	<array>	EncData	<array>
[0]	0x4F5CA2AE	[0]	0x7AF7609A
[1]	0x30A9F0C9	[1]	0x80CB81AF
[2]	0xE24A75D8	[2]	0x12BB6AD7
[3]	0xF4ABEF98	[3]	0xCD9B6043
[4]	0xAF649749	[4]	0x43406B35
[5]	0xB84B0FFC	[5]	0xE4B93EAF
[6]	0x8B4E5CD2	[6]	0x5F7360BC
[7]	0x59C6BF51	[7]	0x0C9DC002
[8]	0x8C8EC584	[8]	0xB85F3AB4
[9]	0x87A12E03	[9]	0x6532B50B
[10]	0x2AB4925D	[10]	0xEB6660A8
[11]	0x40CD0894	[11]	0xA3325A2E
[12]	0x5CA9A9B5	[12]	0x8CAA3786
[13]	0x40F96446	[13]	0x6708737E
[14]	0x2D7D6275	[14]	0x5E4BDFAB
[15]	0x4BD8735D	[15]	0xAD9323EE
[16]	0x313DB846	[16]	0xBB95EC35
[17]	0xAF39F4AD	[17]	0x7933FE5F
[18]	0x8D537356	[18]	0x82E40FD5
[19]	0x2A0D33BE	[19]	0x797C7E78
[20]	0x29A17D01	[20]	0xB3ECCC6D
[21]	0xD825F9B1	[21]	0xD002D547
[22]	0xE1F52535	[22]	0xA8874EE9
[23]	0x10F12035	[23]	0x280E061B
[24]	0xC60FB1D1	[24]	0xD3801A0D
[25]	0x2DC555CF	[25]	0x1FDB409F
[26]	0xABB1A07D	[26]	0x7CB8C243
[27]	0x22724EC8	[27]	0x50185B67
[28]	0x0BBB513F	[28]	0x2B66A98A
[29]	0xD81C60EB	[29]	0x49929735
[30]	0x3F8B3E24	[30]	0xC886A275
[31]	0x3F8B3E24	[31]	0x3F8B3E24

Рисунок Б4 – Результаты теста 4

Тест 5

Используются:

Открытый текст To

Заполнение КЗУ КЗУ

Синхропосылка s1

Расшифрование

Предполагаемый результат:

Такт1	4F 5C A2 AE	30 A9 F0 C9
Такт2	E2 4A 75 D8	F4 AB EF 98
Такт3	AF 64 97 49	B8 4B 0F FC
Такт4	59 C6 BF 51	8C 8E C5 84
Такт5	87 A1 2E 03	2A B4 92 5D
Такт6	40 CD 08 94	5C A9 A9 B5
Такт7	40 F9 64 46	2D 7D 62 75
Такт8	4B D8 73 5D	31 3D B8 46
Такт9	31 3D B8 46	AF 39 F4 AD
Такт10	8D 53 73 56	2A 0D 33 BE
Такт11	29 A1 7D 01	D8 25 F9 B1
Такт12	E1 F5 25 35	10 F1 20 35
Такт13	C6 0F B1 D1	2D C5 55 CF
Такт14	AB B1 A0 7D	22 72 4E C8
Такт15	0B BB 51 3F	D8 1C 60 EB
Такт16	92 E1 2F BD	84 EC EB 59

На практике получен результат, представленный на рисунке Б5.

Expression	Value	Expression	Value
EncData	<array>	DecData	<array>
[0]	0x7AF7609A	[0]	0x4F5CA2AE
[1]	0x80CB81AF	[1]	0x30A9F0C9
[2]	0x12BB6AD7	[2]	0xE24A75D8
[3]	0xCD9B6043	[3]	0xF4ABEF98
[4]	0x43406B35	[4]	0xAF649749
[5]	0xE4B93EAF	[5]	0xB84B0FFC
[6]	0x5F7360BC	[6]	0x8B4E5CD2
[7]	0x0C9DC002	[7]	0x59C6BF51
[8]	0xB85F3AB4	[8]	0x8C8EC584
[9]	0x6532B50B	[9]	0x87A12E03
[10]	0xEB6660A8	[10]	0x2AB4925D
[11]	0xA3325A2E	[11]	0x40CD0894
[12]	0x8CAA3786	[12]	0x5CA9A9B5
[13]	0x6708737E	[13]	0x40F96446
[14]	0x5E4BDFAB	[14]	0x2D7D6275
[15]	0xAD9323EE	[15]	0x4BD8735D
[16]	0xBB95EC35	[16]	0x313DB846
[17]	0x7933FE5F	[17]	0xAF39F4AD
[18]	0x82E40FD5	[18]	0x8D537356
[19]	0x797C7E78	[19]	0x2A0D33BE
[20]	0xB3ECCC6D	[20]	0x29A17D01
[21]	0xD002D547	[21]	0xD825F9B1
[22]	0xA8874EE9	[22]	0xE1F52535
[23]	0x280E061B	[23]	0x10F12035
[24]	0xD3801A0D	[24]	0xC60FB1D1
[25]	0x1FDB409F	[25]	0x2DC555CF
[26]	0x7CB8C243	[26]	0xABB1A07D
[27]	0x50185B67	[27]	0x22724EC8
[28]	0x2B66A98A	[28]	0x0BBB513F
[29]	0x49929735	[29]	0xD81C60EB
[30]	0x80000000	[30]	0x92E12FBD
[31]	0x00000000	[31]	0x84FCFB50

Рисунок Б5 – Результаты теста 5