

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Политехнический институт
Факультет приборостроения,
информационных технологий и
электроники

Кафедра
«Информационная безопасность
систем и технологий»

Специальность 10.05.02 –

Специализация -

Информационная безопасность
телекоммуникационных систем
Разработка защищенных
телекоммуникационных систем

ДИПЛОМНЫЙ ПРОЕКТ

на тему

МОДЕЛЬ АППАРАТНОГО МОДУЛЯ, РЕАЛИЗУЮЩЕГО АЛГОРИТМ
ШИФРОВАНИЯ ГОСТ Р 34.12-2015, ПОСТРОЕННЫЙ НА БАЗИСНЫХ
ЛОГИЧЕСКИХ ЭЛЕМЕНТАХ

Студент _____ Читаев Расим Равильевич

Руководитель ведущий инженер АО «ПНИЭИ» _____ Алексеев С.В.

Нормоконтролёр _____ Иванов А.П.

Рецензент начальник НИС АО «ПНИЭИ» _____ Акмаев А.Ж.

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой _____ С.Л. Зефилов

Работа защищена с отметкой _____ (протокол заседания ГЭК
от _____ № _____)

Секретарь ГЭК _____ А.Г. Фатеев

Пенза 2018

Оглавление

Введение	4
1 Общие принципы шифрования.....	5
1.1 Шифрование	5
1.2 Общие сведения о симметричном шифровании	5
1.3 Режимы работы блочных шифров.....	11
1.4 Виды шифраторов	27
2 Обзор ГОСТ Р 34.12-2015	31
2.1 Общие сведения	31
2.2 Алгоритм блочного шифрования с длиной блока $n = 128$ бит.....	32
3 Реализация алгоритма блочного шифрования с длиной блока $n=128$ бит...	38
3.1 Описание функционирования модели аппаратного модуля	38
3.2 Обоснование выбора языка описания аппаратуры.....	38
3.3 Описание среды разработки ModelSim SE 10.1с	40
3.4 Разработка модели аппаратного модуля.....	43
4 Тестирование модели аппаратного модуля, с помощью набора тестовых последовательностей.....	49
4.1 Тестирование S, R и L преобразований	49
4.2 Тестирование алгоритма развертывания ключа	50
4.3 Тестирование алгоритма зашифрования	51
4.4 Тестирование алгоритма расшифрования	52
4.5 Тестирование процесса программы	53
5 Вредные психофизиологические факторы, влияющие на нервную систему специалиста по защите информации. Методы и средства физической культуры, снижающие их воздействие	55
Заключение	59
Список использованных источников	60
Приложение А. Листинг программы, реализующий алгоритм блочного шифрования с длиной блока $n=128$ бит	61

Приложение Б. Таблица истинности для S преобразования	86
Приложение В. Листинг программы испытательного стенда	94
Приложение Г. Протокол проверки ВКР в системе «Антиплагиат»	97

Введение

С 2016 года в РФ действует новый стандарт блочного шифрования ГОСТ - Р 34.12 2015, разработанный Центром защиты информации и специальной связи ФСБ России с участием ОАО «ИнфоТеКС». Новый стандарт включает в себя два шифра - для блоков длиной 128 и 64 бита. Эти шифры получили названия «Кузнечик» и «Магма» соответственно. Стоит отметить, что шифр «Магма» - это старый шифр ГОСТ 28147-89 при зафиксированных блоках подстановок. Определенные в стандарте алгоритмы криптографического преобразования предназначены для аппаратной или программной реализации.

Известно, что большинство криптографических операций при аппаратной реализации выполняется быстрее, чем при программной. Также аппаратная реализация шифра надежнее программного исполнения, поскольку есть возможность прямой загрузки ключей в шифратор, тем самым снимается угроза перехвата ключей, которые при использовании программных шифраторов циркулируют в оперативной памяти компьютера.

На данный момент существует много программных реализаций шифра «Кузнечик», например, эталонная реализация на языке С, что нельзя сказать об аппаратном исполнении. Тем самым актуальна задача исследования особенностей аппаратной реализации шифра «Кузнечик».

В результате выполнения работы должна быть получена модель аппаратного модуля на языке описания аппаратуры VHDL, реализующая функции криптографического преобразования информации.

1 Общие принципы шифрования

1.1 Шифрование

Шифрование – это метод, используемый для преобразования данных в зашифрованный текст для того, чтобы они были прочитаны только пользователем, обладающим соответствующим ключом шифрования для расшифровки содержимого. Шифрование используется тогда, когда требуется повышенный уровень защиты данных - при хранении данных в ненадежных источниках или передачи данных по незащищенным каналам связи.

В шифрование состоит из двух составляющих - зашифрование и расшифрование.

С помощью шифрования обеспечиваются три состояния безопасности информации:

- конфиденциальность. Шифрование используется для скрытия информации от неавторизованных пользователей при передаче или при хранении;
- целостность. Шифрование используется для предотвращения изменения информации при передаче или хранении;
- идентифицируемость. Шифрование используется для аутентификации источника информации и предотвращения отказа отправителя информации от того факта, что данные были отправлены именно им.

1.2 Общие сведения о симметричном шифровании

Симметричное шифрование - шифрование, которое использует один и тот же ключ для зашифрования и расшифровывания. Т.е. в частном случае существует некий секретный ключ, известный обеим сторонам, с помощью которого одна сторона зашифровывает информацию, а вторая - расшифровывает. Это означает, что любой, кто имеет доступ к ключу шифрования, может расшифровать сообщение. Соответственно с целью обеспечения защиты от несанкционированного доступа к зашифрованной

информации все ключи шифрования в симметричных криптосистемах должны держаться в секрете.

Поэтому симметричные криптосистемы называют криптосистемами с секретным ключом.

Криптография с симметричными ключами стойкая, что делает практически невозможным процесс дешифрования без знания ключа. При прочих равных условиях стойкость определяется длиной ключа. При использовании симметричных алгоритмов шифрования механизмы для распределения ключей должны обладать высокой надежностью. Также имеется проблема безопасного распространения симметричных ключей. В алгоритмах симметричного шифрования достигается высокая скорость шифрования больших объемов данных за счет использования ключей не очень большой длины. В симметричных криптосистемах конфиденциальность передачи информации зависит от надежности шифра и обеспечения конфиденциальности ключа шифрования. Симметричное шифрование остаётся самым актуальным и криптографически гарантированным методом защиты информации. Также в современных автоматизированных системах такое шифрование в основном используется из-за простоты и скорости реализаций. В симметричном шифровании, основанном на использовании составных ключей, идея состоит в том, что секретный ключ делится на две части, хранящиеся отдельно. Каждая часть сама по себе не позволяет выполнить зашифрование. Основным недостатком симметричного шифрования является то, что секретный ключ должен быть известен и отправителю, и получателю. Это создает проблему распространения ключей. Получатель на основании наличия зашифрованного и расшифрованного сообщения не может доказать, что он получил это сообщение от конкретного отправителя, поскольку такое же сообщение он мог сгенерировать самостоятельно.

На рисунке 1 представлена схема симметричной криптосистемы шифрования:

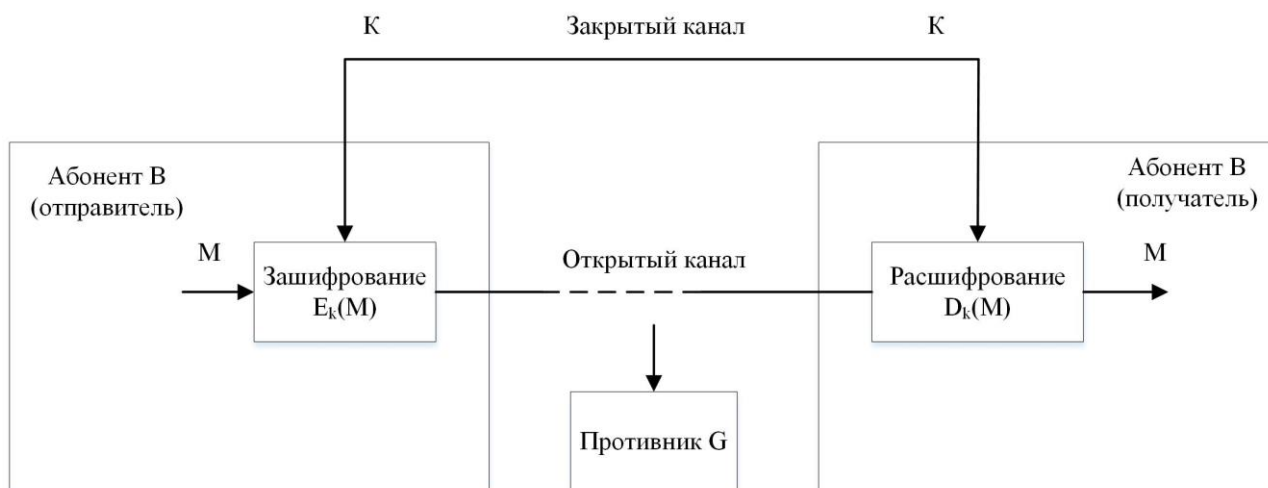


Рисунок 1 - Схема симметричной криптосистемы шифрования

Такие системы характеризуются наиболее высокой скоростью шифрования, и с их помощью обеспечивается конфиденциальность и подлинность передаваемой информации, а также ее целостность.

Этапы обмена информацией:

- отправитель передает получателю ключ (в сети, где есть больше двух абонентов у каждой пары абонентов должен быть свой ключ, отличный от ключей других пар);
- отправитель, используя ключ, зашифровывает сообщение, которое пересылается получателю;
- получатель получает сообщение и расшифровывает его этим же ключом.

Для повышения защищенности системы можно использовать уникальные ключи для каждого сеанса связи или менять их каждый день.

Обычно при симметричном шифровании используется сложная и многоступенчатая комбинация подстановок и перестановок исходных данных. Причем ступеней (проходов) может быть большое количество, каждой из которых должен соответствовать «ключ прохода». Операция подстановки выполняет первое требование, предъявляемое к симметричному шифру. Происходит избавление от любых статистических данных путем перемешивания битов сообщения по заданному закону. Перестановка необходима для выполнения второго требования – придания алгоритму

нелинейности. Это достигается за счет замены определенной части сообщения заданного объема на стандартное значение путем обращения к исходному массиву.

К каждой реализации симметричных шифров предъявляются требования к параметрам используемого алгоритма.

Стойкость (криптостойкость) алгоритма — это его способность противостоять криптоанализу. Стойким называют алгоритм, успешная атака на который требует от атакующего обладания недостижимым в реальности объемом вычислительных ресурсов или перехваченных открытых и зашифрованных сообщений либо значительных затрат времени на раскрытие, таких, что к этому моменту защищенная информация утратит свою актуальность. Зачастую криптостойкость алгоритмов невозможно математически доказать. Возможно только доказать уязвимости в криптографическом алгоритме либо (в случае криптосистем с открытым ключом) свести задачу взлома алгоритма к некоторой задаче, которая считается вычислительно сложной (то есть доказать, что взлом сложнее решения этой задачи).

Абсолютно стойкие системы. Об абсолютной стойкости (или теоретической стойкости) говорят там в случае, когда криптосистема не может быть взломана ни теоретически, ни практически, даже при наличии у атакующего бесконечно больших вычислительных ресурсов. Доказательство существования абсолютно стойких алгоритмов шифрования приведено в трудах Клода Шеннона «Теория связи в секретных системах». Здесь же определены требования к такого рода системам:

- для каждого сообщения генерируется новый ключ, т. е. каждый ключ используется один раз;
- статистическая надежность ключа (вероятности появления каждого из возможных символов равны, символы в ключевой последовательности независимы и случайны);
- длина ключа равна или больше длины сообщения;

- исходный (открытый) текст обладает некоторой избыточностью (это является критерием оценки правильности расшифровки)

Стойкость абсолютно стойких систем абсолютно не зависит от того, какие вычислительные возможности есть у криптоаналитика. Практическое применение подобных систем (удовлетворяющих требованиям абсолютной стойкости) ограничено из-за их высокой стоимости и удобства использования.

Шеннон в своих трудах доказал, что примером абсолютно стойкого алгоритма является шифр Вернама (одноразовый блокнот). При корректном использовании шифра Вернама злоумышленник не сможет получить никакой информации об открытом тексте.

Достаточно стойкие системы. В криптографических системах гражданского назначения в основном применяются практически стойкие или вычислительно стойкие системы. О вычислительной стойкости такой системы говорят в случае, если потенциальная возможность вскрыть шифр существует, но только при определенных параметрах и ключах шифрования. На практике атакующий, имея доступ к современным технологиям, не может обладать достаточными вычислительными ресурсами для вскрытия шифра за адекватный отрезок времени. От вычислительных возможностей криптоаналитика зависит стойкость таких систем.

Практическая стойкость таких систем основана на теории сложности и может оцениваться в расчете на определенный момент времени и с двух позиций:

- вычислительная сложность полного перебора;
- известные на данный момент уязвимости (слабости) таких систем и их влияние на вычислительную сложность.

В каждом конкретном случае могут существовать дополнительные критерии оценки стойкости.

Стойкость системы доказуема, если доказательство сводится к решению конкретной трудно решаемой математической проблемы, лежащей в основе

алгоритма. Поэтому, криптосистема RSA стойка, только если модуль алгоритма n невозможно легко факторизовать.

Раундом (циклом) криптографического протокола в криптографических протоколах с двумя участниками называют временной интервал, в котором активен только один из участников. По-другому этот процесс называют проходом протокола. Цикл (раунд) завершается формированием и отправкой сообщения участником, при этом активный участник переходит в состояние ожидания и передает активность другому участнику. В протоколах с тремя и более участниками в синхронном случае цикл — это период времени между двумя точками синхронизации.

К каждой точке синхронизации каждый участник должен послать все свои сообщения, которые он должен передать другим участникам в настоящем цикле.

В настоящее время имеется два вида симметричных шифров:

- блочные шифры. Входящее сообщение разбивают на блоки определенной длины (по 64 или 128 бит), а после этого обрабатывается. К каждому блоку применяется ключ в установленном порядке. Обычно это происходит в несколько циклов перемешивания и подстановки, такие циклы называются раундами. Результатом повторения раундов является лавинный эффект, то есть постепенно нарастающая потеря соответствия битов между блоками открытых и зашифрованных данных.

К достоинствам блочных шифров относят схожесть процедур зашифровывания и расшифровывания, которые, как правило, отличаются лишь порядком действий. Это упрощает процесс создания шифровальных устройств, так как позволяет использовать одни и те же блоки в цепях зашифровывания и расшифровывания;

- поточные шифры. В данном каждый бит либо байт исходного (открытого) текста шифруется с использованием гаммирования. Такой шифр может быть легко создан на основе блочного (например, по ГОСТ 28147-86 в режиме гаммирования), который запущен в специальном режиме.

1.3 Режимы работы блочных шифров.

Чтобы использовать алгоритмы блочного шифрования для различных криптографических задач существует несколько режимов их работы. Наиболее часто встречающимися в практике являются следующие режимы:

- режим простой замены;
- режим гаммирования;
- режим гаммирования с обратной связью по выходу;
- режим простой замены с зацеплением;
- режим гаммирования с обратной связью по шифртексту;
- режим выработки имитовставки.

1.3.1 Режим простой замены. Длина сообщений, зашифровываемых в режиме простой замены, должна быть кратна длине блока базового алгоритма блочного шифрования n , поэтому, при необходимости, к исходному сообщению должна быть предварительно применена процедура дополнения.

Зашифровывание (расшифровывание) в режиме простой замены заключается в зашифровывании (расшифровывании) каждого блока текста с использованием базового алгоритма блочного шифрования.

Зашифрование. Открытый и, при необходимости, дополненный текст $P \in V^*$, $|P| = n \cdot q$, представляется в виде: $P = P_1 || P_2 || \dots || P_q$, $P_i \in V_n$, $i = 1, 2, \dots, q$. Блоки шифртекста вычисляются по следующему правилу:

$$C_i = e_K(P_i), i = 1, 2, \dots, q. \quad (1)$$

Результирующий шифртекст имеет вид:

$$C = C_1 || C_2 || \dots || C_q.$$

Зашифрование в режиме простой замены проиллюстрировано на рисунке 2.

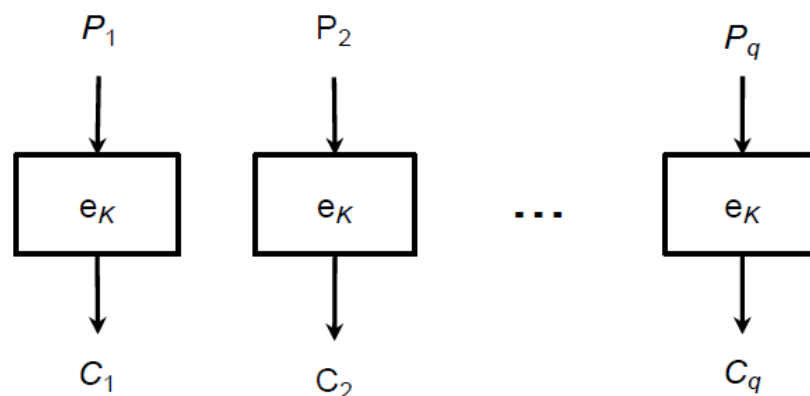


Рисунок 2 – Зашифрование в режиме простой замены

Расшифрование. Шифртекст представляется в виде: $C = C_1 || C_2 || \dots || C_q$, $C_i \in V_n$, $i = 1, 2, \dots, q$. Блоки открытого текста вычисляются по следующему правилу:

$$P_i = d_K(C_i), i = 1, 2, \dots, q. \quad (2)$$

Исходный (дополненный) открытый текст имеет вид:

$$P = P_1 || P_2 || \dots || P_q.$$

Если к исходному открытому тексту была применена процедура дополнения, то после расшифровывания следует произвести обратную процедуру. Для однозначного восстановления сообщения может потребоваться знание длины исходного сообщения.

Иллюстрация расшифровывания в режиме простой замены приведена на рисунке 3.

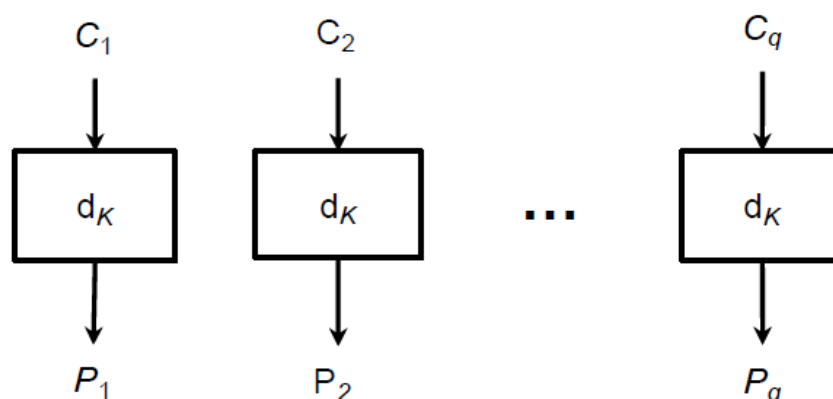


Рисунок 3 – Расшифрование в режиме простой замены

1.3.2 Режим гаммирования.

Параметром режима гаммирования является целочисленная величина s $0 < s \leq n$. При использовании режима гаммирования не требуется применение процедуры дополнения сообщения.

Для зашифрования (расшифрования) каждого отдельного открытого текста на одном и том же ключе используется значение уникальной синхропосылки $IV \in V_{\frac{n}{2}}$.

Зашифрование в режиме гаммирования заключается в покомпонентном сложении открытого текста с гаммой шифра. Гамма вырабатывается блоками длины s путем зашифрования последовательности значений счетчика $CTR_i \in V_n, i = 1, 2, \dots$, базовым алгоритмом блочного шифрования с последующим усечением. Начальным значением счетчика является $CTR_1 = I_n(IV) = IV \parallel 0_2^n$. Последующие значения счетчика вырабатываются с помощью функции $Add: V_n \rightarrow V_n$ следующим образом:

$$CTR_{i+1} = Add(CTR_i) = Vec_n(Int_n(CTR_i) \boxplus 1). \quad (3)$$

Зашифрование. Открытый текст $P \in V^*$ представляется в виде $P = P_1 \parallel P_2 \parallel \dots \parallel P_q, P_i \in V_s, i = 1, 2, \dots, q-1, P_q \in V_r, r \leq s$. Блоки шифртекста вычисляются по следующему правилу:

$$\begin{cases} C_1 = P_1 \oplus T_s(e_K(CTR_1)), i = 1, 2, \dots, q-1, \\ C_q = P_q \oplus T_r(e_K(CTR_q)). \end{cases} \quad (4)$$

Результирующий шифртекст имеет вид:

$$C = C_1 \parallel C_2 \parallel \dots \parallel C_q.$$

Зашифрование в режиме гаммирования проиллюстрировано на рисунке 4.

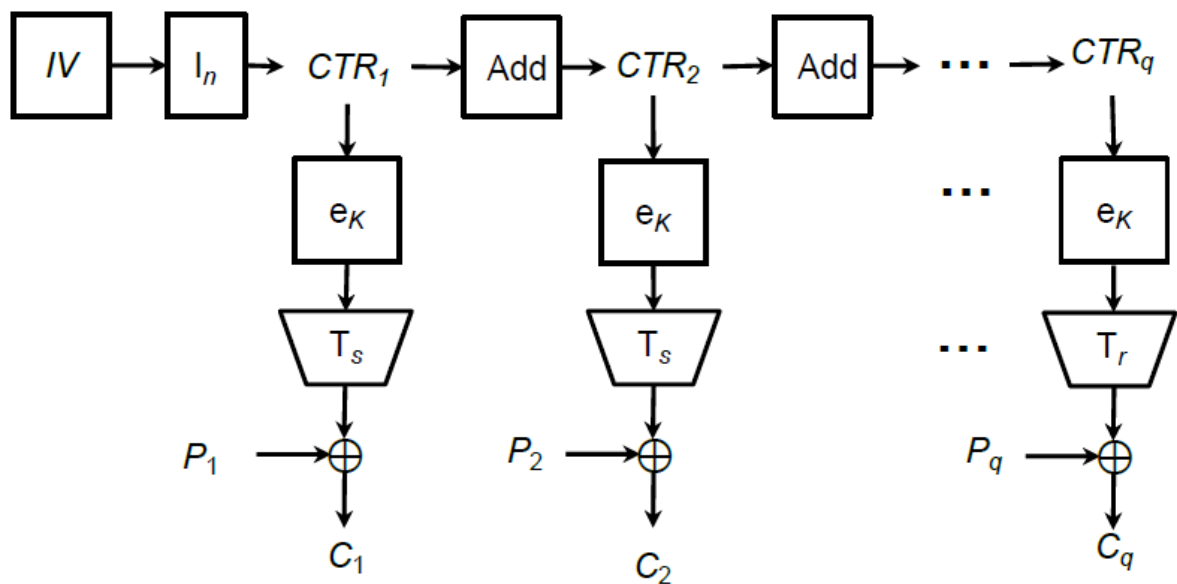


Рисунок 4 – Зашифрование в режиме гаммирования

Расшифрование. Шифртекст представляется в виде: $C = C1||C2||...||Cq$, $Ci \in Vs, i = 1, 2, \dots, q-1, Cq \in Vr, r \leq s$.

Блоки открытого текста вычисляются по следующему правилу:

$$\begin{cases} P_1 = C_1 \oplus T_s(e_K(CTR_1)), i = 1, 2, \dots, q-1, \\ P_q = C_q \oplus T_r(e_K(CTR_q)). \end{cases} \quad (5)$$

Исходный открытый текст имеет вид

$$P = P1||P2||...||Pq.$$

Иллюстрация расшифровывания в режиме гаммирования приведена на рисунке 5.

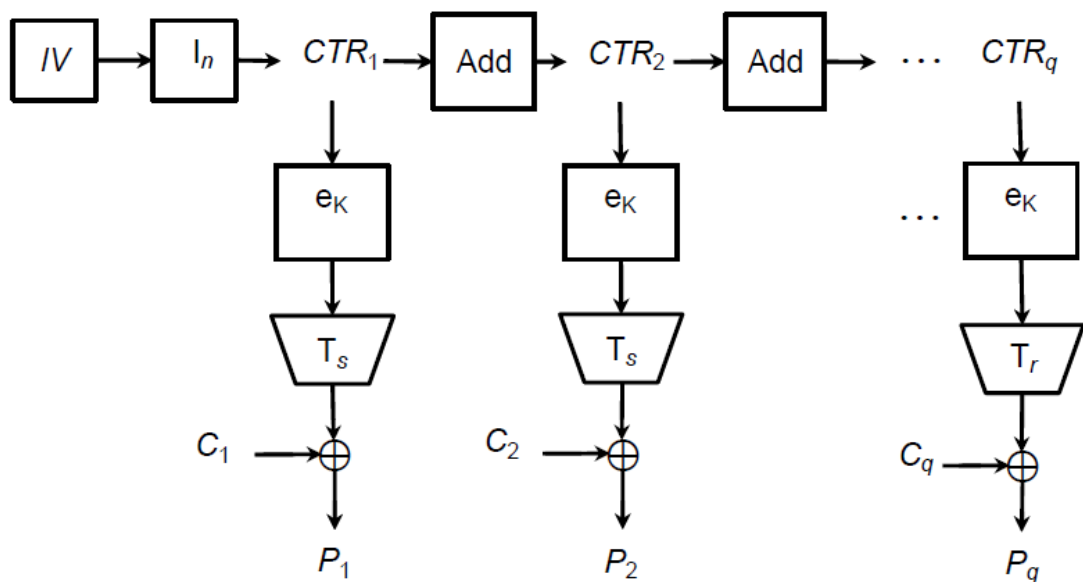


Рисунок 5 – Расшифрование в режиме гаммирования

1.3.3 Режим гаммирования с обратной связью по выходу

Параметрами режима гаммирования с обратной связью по выходу являются целочисленные величины s и m , $0 < s \leq n$, $m = n \cdot z$, $z \geq 1$ – целое число.

Применение процедуры дополнения сообщения не требуется при использовании режима гаммирования с обратной связью по выходу.

При шифровании на одном ключе для каждого открытого текста используется значение уникальной или непредсказуемой (случайной или псевдослучайной) синхропосылки $IV \in V_m$.

При шифровании в режиме гаммирования с обратной связью по выходу используется двоичный регистр сдвига R длины m . Начальным заполнением регистра является значение синхропосылки IV .

Зашифровывание в режиме гаммирования с обратной связью по выходу заключается в покомпонентном сложении открытого текста с гаммой шифра, вырабатываемой блоками длины s . При вычислении очередного блока гаммы выполняется зашифрование n разрядов регистра сдвига с большими номерами базовым алгоритмом блочного шифрования. После заполнения регистр сдвигается на n бит в сторону разрядов с большими номерами, при этом в разряды с меньшими номерами записывается полученный выход базового алгоритма блочного шифрования. Блок гаммы вычисляется путем усечения выхода базового алгоритма блочного шифрования.

Зашифрование. Открытый текст $P \in V^*$ представляется в виде $P = P1||P2||...||Pq$, $Pi \in V_s$, $i = 1, 2, \dots, q-1$, $P_q \in V_r$, $r \leq s$. Блоки шифртекста вычисляются по следующему правилу:

$$\begin{aligned} R_1 &= IV, \\ \left\{ \begin{array}{l} Y_1 = e_k(MSB_n(R_1)), \\ C_1 = P_1 \oplus T_s(Y_1), i = 1, 2, \dots, q-1, \\ P_{i+1} = LSB_{m-n}(R_i) \parallel Y_i, \end{array} \right. & (6) \\ Y_q &= e_k(MSB_n(R_q)), \\ C_q &= P_q \oplus T_r(Y_q). \end{aligned}$$

Результирующий шифртекст имеет вид:

$$C = C_1 || C_2 || \dots || C_q.$$

Иллюстрация зашифрования в режиме гаммирования с обратной связью по выходу приведена на рисунке 6.

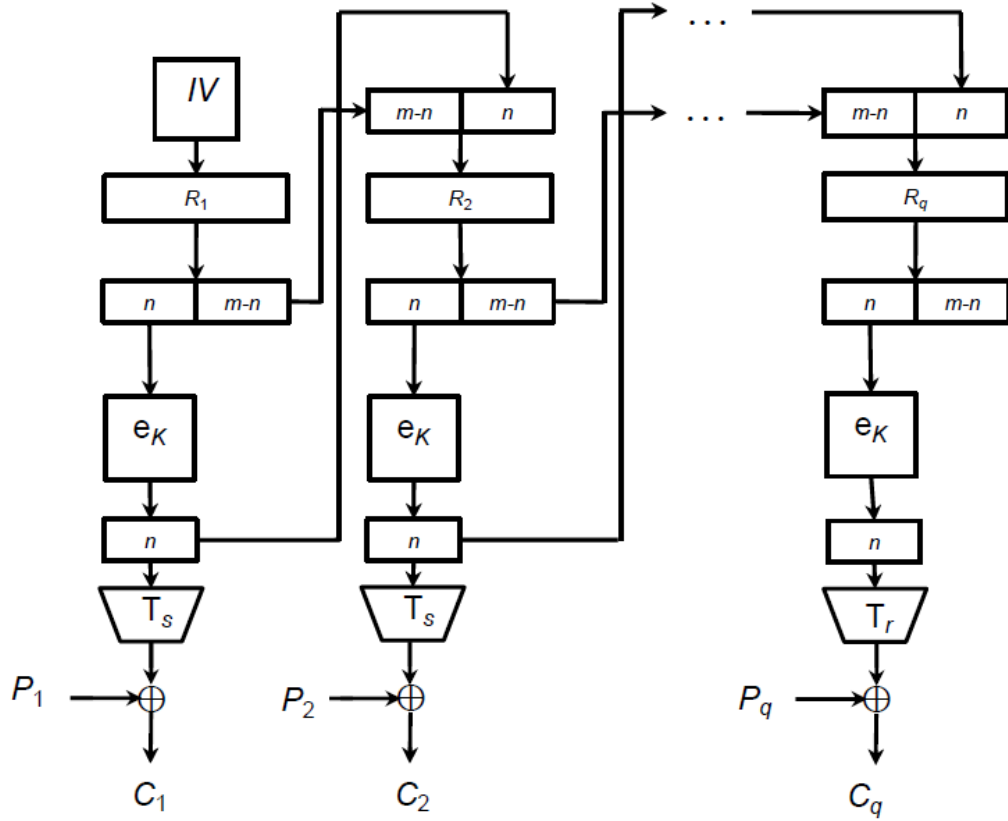


Рисунок 6 – Зашифрование в режиме гаммирования с обратной связью по выходу

Расшифрование. Шифртекст представляется в виде: $C = C_1 || C_2 || \dots || C_q$, $C_i \in V_s, i = 1, 2, \dots, q-1, C_q \in V_r, r \leq s$.

Блоки открытого текста вычисляются по следующему правилу:

$$R_1 = IV,$$

$$\begin{cases} Y_1 = e_k(MSB_n(R_1)), \\ P_1 = C_1 \oplus T_s(Y_1), i = 1, 2, \dots, q-1, \\ P_{i+1} = LSB_{m-n}(R_i) || Y_i, \end{cases} \quad (7)$$

$$Y_q = e_k(MSB_n(R_q)),$$

$$P_q = C_q \oplus T_r(Y_q)$$

Исходный открытый текст имеет вид

$$P = P_1 || P_2 || \dots || P_q.$$

Иллюстрация расшифрования в режиме гаммирования с обратной связью по выходу приведена на рисунке 7.

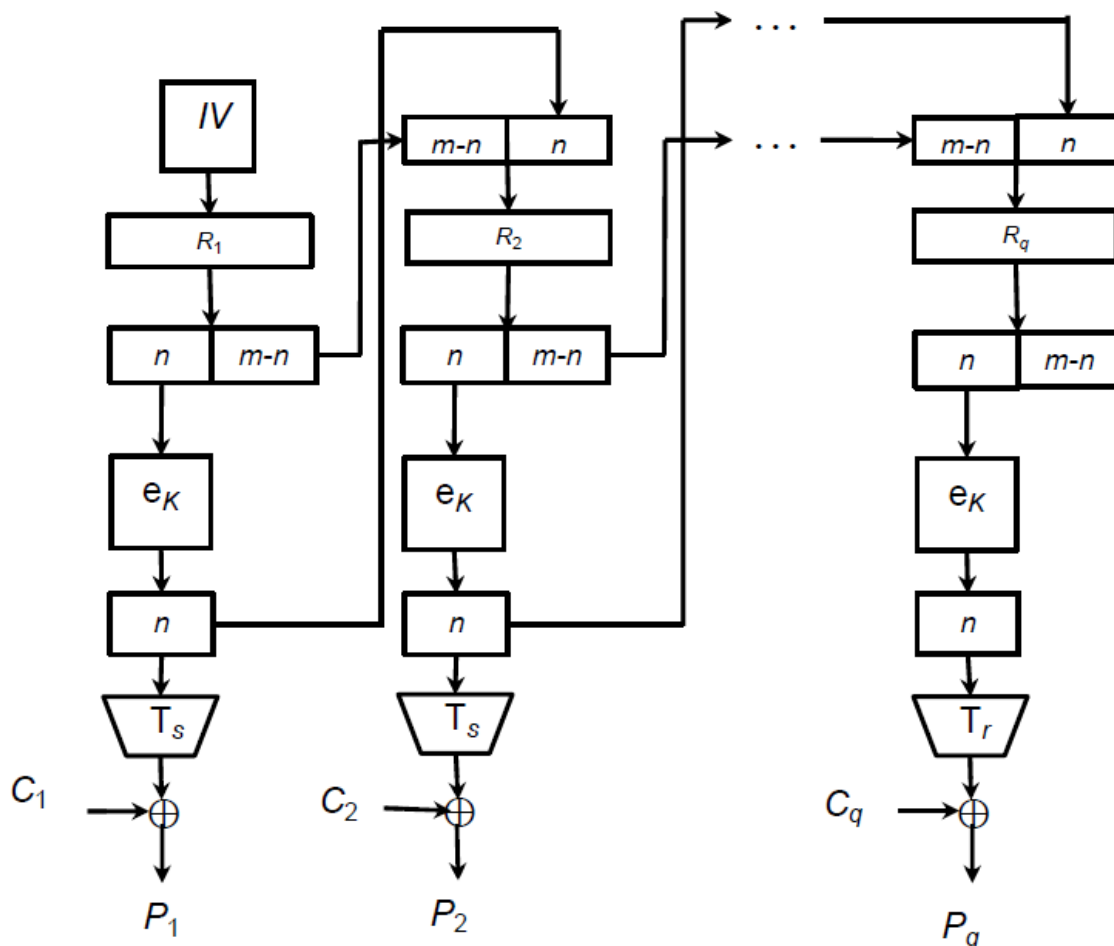


Рисунок 7 – Расшифрование в режиме гаммирования

1.3.4 Режим простой замены с зацеплением

Параметром режима простой замены с зацеплением является целочисленная величина m , $m = n \cdot z$, $z \geq 1$ – целое число.

Длина сообщений, которые зашифровываются в режиме простой замены с зацеплением, должна быть кратна длине блока базового алгоритма блочного шифрования n . При необходимости к исходному сообщению должна быть предварительно применена процедура дополнения.

Для каждого отдельного открытого текста при шифровании на одном ключе используется значение непредсказуемой (случайной или псевдослучайной) синхропосылки $IV \in V_m$.

При шифровании в режиме простой замены с зацеплением используется двоичный регистр сдвига R длины m . Начальное заполнение регистра — значение синхропосылки IV .

В режиме простой замены с зацеплением очередной блок шифртекста получается путем зашифровывания результата покомпонентного сложения значения каждого следующего блока открытого текста со значением n разрядов регистра сдвига с большими номерами. После этого регистр сдвигается на один блок в сторону разрядов с большими номерами. В разряды с меньшими номерами записывается значение блока шифртекста.

Зашифрование. Открытый и, при необходимости, дополненный текст $P \in V^*$, $|P| = n \cdot q$, представляется в виде: $P = P_1 || P_2 || \dots || P_q$, $P_i \in V_n$, $i = 1, 2, \dots, q$. Блоки шифртекста вычисляются по следующему правилу:

$$\begin{aligned} R_1 &= IV, \\ \begin{cases} C_1 = e_k(P_1 \oplus MSB_n(R_1)), \\ R_{i+1} = LSB_{m-n}(R_i) || C_i, \end{cases} & i=1, 2, \dots, q, \\ C_q &= e_k(P_q \oplus MSB_n(R_q)). \end{aligned} \quad (8)$$

Результирующий шифртекст имеет вид:

$$C = C_1 || C_2 || \dots || C_q.$$

Зашифрование в режиме простой замены с зацеплением проиллюстрировано на рисунке 8.

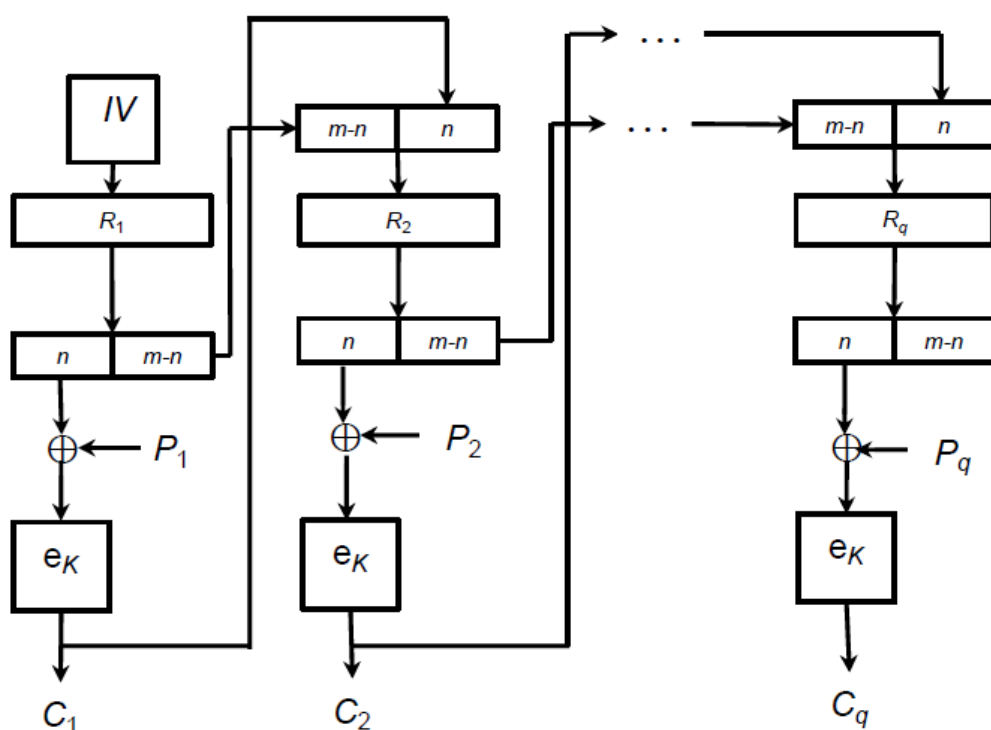


Рисунок 8 – Зашифрование в режиме простой замены с зацеплением

Расшифрование. Шифртекст представляется в виде: $C = C_1 || C_2 || \dots || C_q$, $C_i \in V_n$, $i = 1, 2, \dots, q$. Блоки открытого текста вычисляются по следующему правилу:

$$\begin{aligned}
 R_1 &= IV, \\
 \begin{cases} P_1 = d_k(C_1) \oplus MSB_n(R_1), \\ R_{i+1} = LSB_{m-n}(R_i) || C_i, \end{cases} & i=1, 2, \dots, q-1 \\
 P_q &= d_k(C_q) \oplus MSB_n(R_q).
 \end{aligned} \tag{9}$$

Исходный (дополненный) открытый текст имеет вид:

$$P = P_1 || P_2 || \dots || P_q.$$

Если к исходному открытому тексту была применена процедура дополнения, то после расшифрования следует произвести обратную дополнению процедуру. Для однозначного восстановления сообщения может потребоваться знание длины исходного сообщения.

Расшифрование в режиме простой замены с зацеплением проиллюстрировано на рисунке 9.

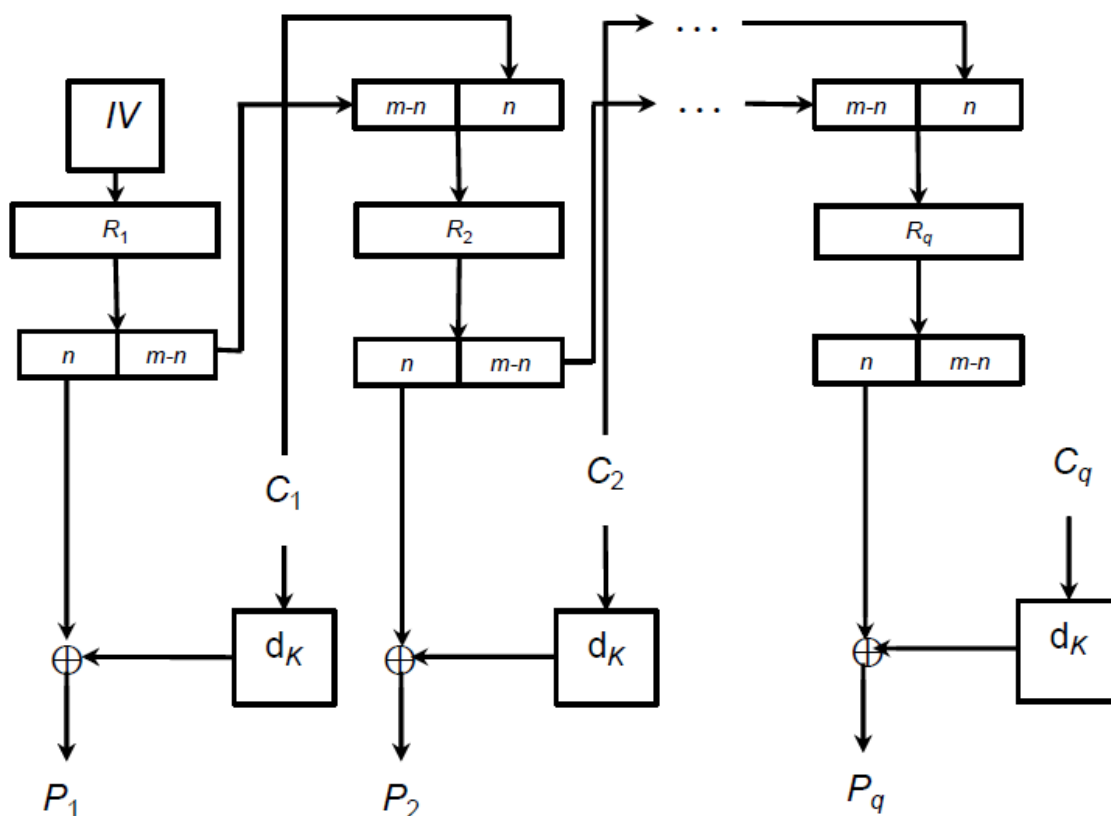


Рисунок 9 – Расшифрование в режиме простой замены с зацеплением

1.3.5 Режим гаммирования с обратной связью по шифртексту

Параметрами режима гаммирования с обратной связью по шифртексту являются целочисленные величины s и m , $0 < s \leq n$, $n \leq m$.

В определенной системе обработки информации на длину сообщения P может накладываться ограничение $|P| = s \cdot q$ либо не накладываться никаких ограничений. В случае если такое ограничение накладывается, то к исходному сообщению, в случае необходимости, должна быть предварительно применена процедура дополнения.

При шифровании нескольких открытых текстов на одном ключе для каждого используется значение непредсказуемой (случайной или псевдослучайной) синхропосылки $IV \in V_m$.

При шифровании в режиме гаммирования с обратной связью по шифртексту используется двоичный регистр сдвига R длины m . Начальным заполнением регистра является значение синхропосылки IV .

Зашифрование в режиме гаммирования с обратной связью по шифртексту заключается в покомпонентном сложении открытого текста с гаммой шифра, которая вырабатывается блоками длины s . При вычислении очередного блока гаммы выполняется зашифрование n разрядов регистра сдвига с большими номерами базовым алгоритмом блочного шифрования с последующим усечением. После этого заполнение регистра сдвигается на s разрядов в сторону разрядов с большими номерами, при этом в разряды с меньшими номерами записывается полученный блок шифртекста. Этот блок является результатом покомпонентного сложения гаммы шифра и блока открытого текста.

Зашифрование. Открытый текст $P \in V^*$ представляется в виде $P = P_1 || P_2 || \dots || P_q$, $P_i \in V_s$, $i = 1, 2, \dots, q-1$, $P_q \in V_r$, $r \leq s$. Блоки шифртекста вычисляются по следующему правилу:

$$\begin{aligned} R_1 &= IV, \\ \begin{cases} C_1 = P_i \oplus T_s(e_k(MSB_n(R_i))), \\ R_{i+1} = LSB_{m-n}(R_i) || C_i, \end{cases} & i=1, 2, \dots, q-1 \\ C_q &= P_q \oplus T_r(e_k(MSB_n(R_q))). \end{aligned} \quad (10)$$

Результирующий шифртекст имеет вид:

$$C = C_1 || C_2 || \dots || C_q.$$

Зашифрование в режиме гаммирования с обратной связью по шифртексту проиллюстрировано на рисунке 10.

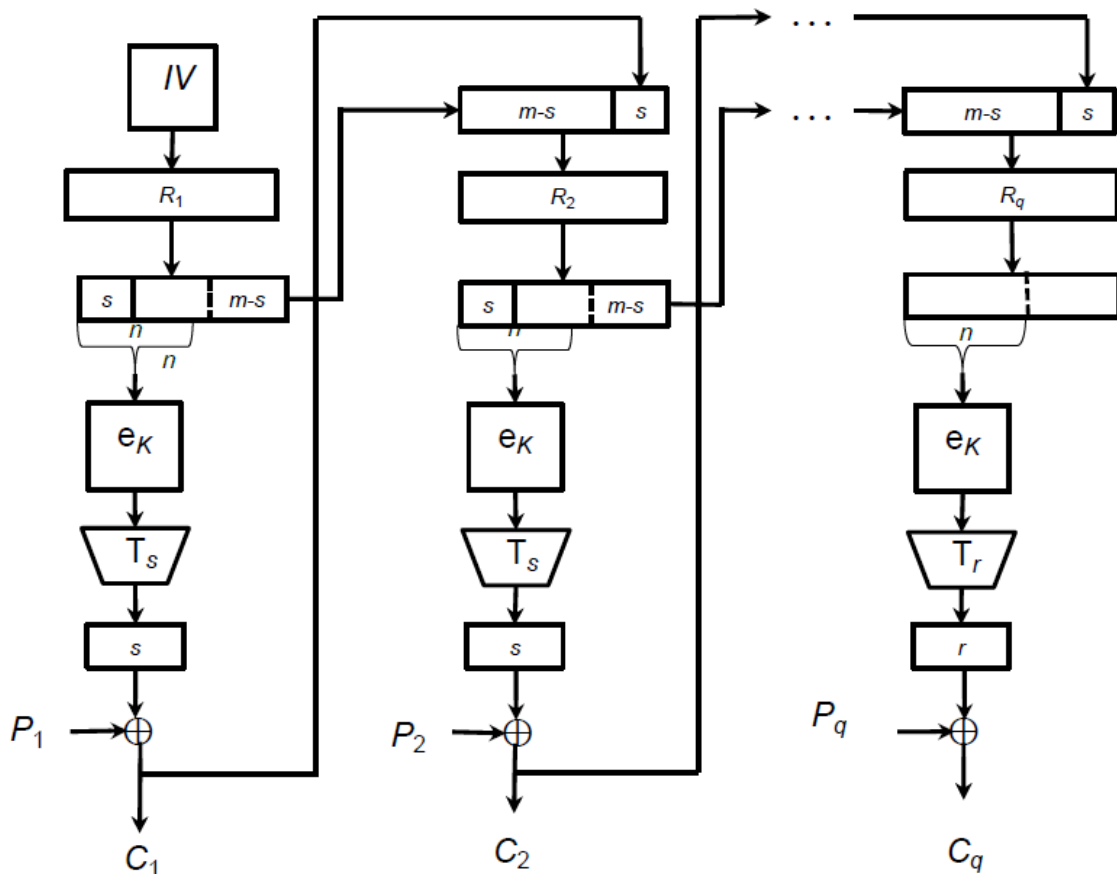


Рисунок 10 – Зашифрование в режиме гаммирования с обратной связью по шифртексту

Расшифрование. Шифртекст представляется в виде: $C = C_1 || C_2 || \dots || C_q$, $C_i \in V_s$, $i = 1, 2, \dots, q-1$, $C_q \in V_r$, $r \leq s$. Блоки открытого текста вычисляются по следующему правилу:

$$\begin{aligned}
 R_1 &= IV, \\
 \begin{cases} P_i = C_i \oplus T_s(e_k(MSB_n(R_i))) \\ R_{i+1} = LSB_{m-n}(R_i) || C_i \end{cases} & i=1, 2, \dots, q-1, \\
 P_q &= C_q \oplus T_r(e_k(MSB_n(R_q))).
 \end{aligned} \tag{11}$$

Исходный открытый текст имеет вид:

$$P = P_1 || P_2 || \dots || P_q.$$

Если к исходному открытому тексту была применена процедура дополнения, то после расшифрования следует произвести обратную дополнению процедуру. Для однозначного восстановления сообщения может потребоваться знание длины исходного сообщения.

Расшифрование в режиме гаммирования с обратной связью по шифртексту проиллюстрировано на рисунке 11.

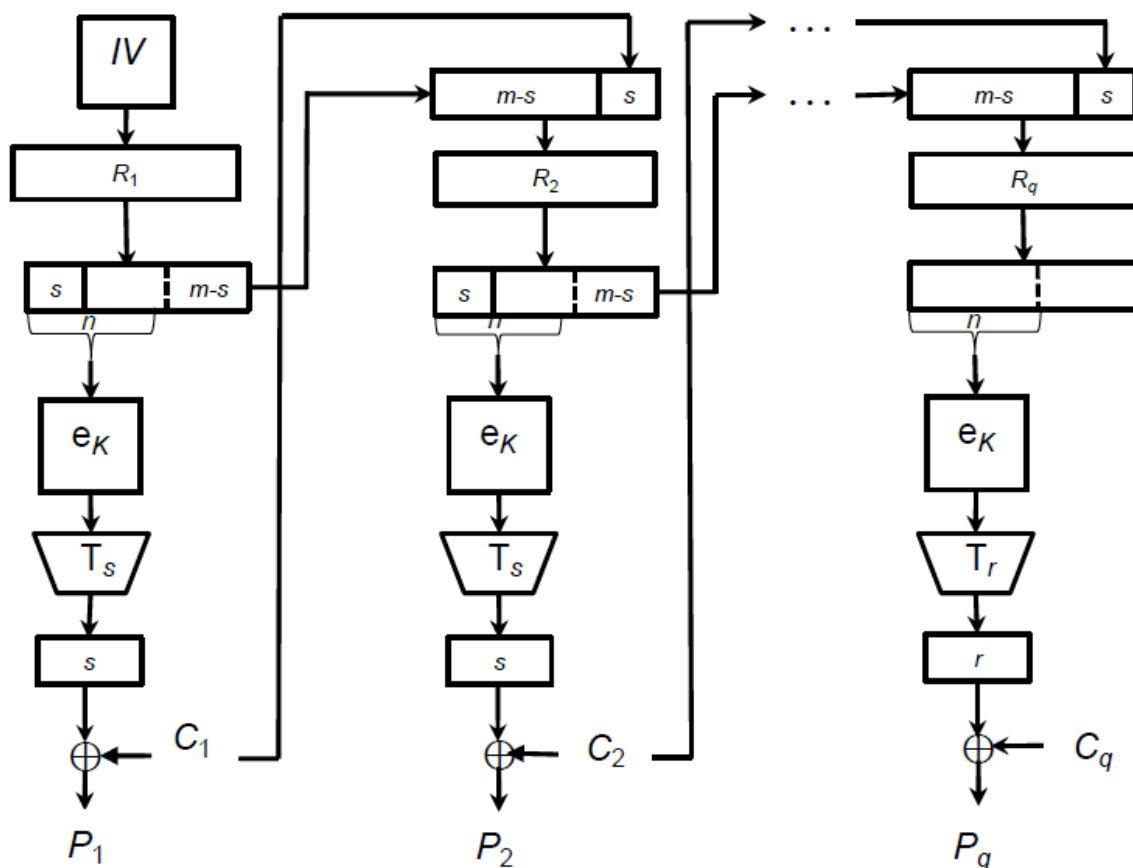


Рисунок 11 – Расшифрование в режиме гаммирования с обратной связью по шифртексту

1.3.6 Режим выработки имитовставки

Режим выработки имитовставки, описание которого представлено ниже, реализует конструкцию OMAC1 (стандартизован в ISO под названием CMAC [1]).

Параметром режима является длина имитовставки в битах $0 < s \leq n$.

Выработка вспомогательных ключей. Для вычисления значения имитовставки используются вспомогательные ключи, которые вычисляются с использованием ключа K . Длины вспомогательных ключей равны длине блока n базового алгоритма блочного шифрования.

Процедура выработки вспомогательных ключей может быть представлена в следующей форме

$$R = e_K(0^n);$$

$$\begin{cases} R \ll 1, \text{ если } MSB_1(R) = 0, \\ (R \ll 1) \oplus B_n, \text{ иначе;} \\ K \ll 1, \text{ если } MSB_1(K_1) = 0, \\ (K \ll 1) \oplus B_n, \text{ иначе;} \end{cases}$$

где $B_{64} = 0^{59} \parallel 11011$, $B_{128} = 0^{120} \parallel 10000111$.

Если значение n не равно 64 или 128, следует использовать следующую процедуру определения значения константы B_n . Рассмотрим множество примитивных многочленов степени n над полем $GF(2)$ с наименьшим количеством ненулевых коэффициентов, далее упорядочим это множество лексикографически по возрастанию векторов коэффициентов и обозначим через $f_n(x)$ первый многочлен в этом упорядоченном множестве.

Рассмотрим поле $GF(2n)[x]/(f_n(x))$, зафиксируем в нем степенной базис и будем обозначать операцию умножения в этом поле символом $\otimes$. Вспомогательные ключи K_1 и K_2 вычисляются следующим образом:

$$\begin{cases} R = e_K(0^n), \\ K_1 = Poly_n^{-1}(Poly_n(R) \otimes x), \\ K_2 = Poly_n^{-1}(Poly_n(R) \otimes x^2). \end{cases} \quad (12)$$

Вспомогательные ключи K_1 , K_2 и промежуточное значение R наряду с ключом K являются секретными параметрами. Компрометация какого-либо из этих значений приводит к возможности построения эффективных методов анализа всего алгоритма.

Вычисление значения имитовставки. Процедура вычисления значения имитовставки похожа на процедуру зашифрования в режиме простой замены с зацеплением при $m = n$ и инициализации начального заполнения регистра сдвига значением 0^n . На вход алгоритму шифрования подается результат покомпонентного сложения очередного блока текста и результата зашифрования на предыдущем шаге. Основное отличие заключается в процедуре обработки последнего блока: на вход базовому алгоритму блочного шифрования подается результат покомпонентного сложения последнего блока, результата зашифрования на предыдущем шаге и одного из вспомогательных ключей. В зависимости от того, является ли последний блок

исходного сообщения полным или нет, выбирается конкретный вспомогательный ключ. Значением имитовставки MAC является результат применения процедуры усечения к выходу алгоритма шифрования при обработке последнего блока.

Исходное сообщение $P \in V^*$, для которого требуется вычислить имитовставку, представляется в виде:

$$P = P_1 || P_2 || \dots || P_q,$$

где $P_i \in V_n, i = 1, 2, \dots, q-1, P_q \in V_r, r \leq n$.

Процедура вычисления имитовставки описывается следующим образом:

$$C_0 = 0_n,$$

$$C_i = eK(P_i \oplus C_{i-1}), i = 1, 2, \dots, q-1, \quad (13)$$

$$MAC = Ts(eK(P_q^* \oplus C_{q-1} \oplus K^*)),$$

где

$$K^* = \begin{cases} K_1, \text{ если } |P_q| = n, \\ K_2, \text{ иначе,} \end{cases}$$

P_q^* – последний блок сообщения, полученного в результате дополнения исходного сообщения с помощью процедуры 3.

Процедура вычисления имитовставки проиллюстрирована на рисунках 12 – 14.

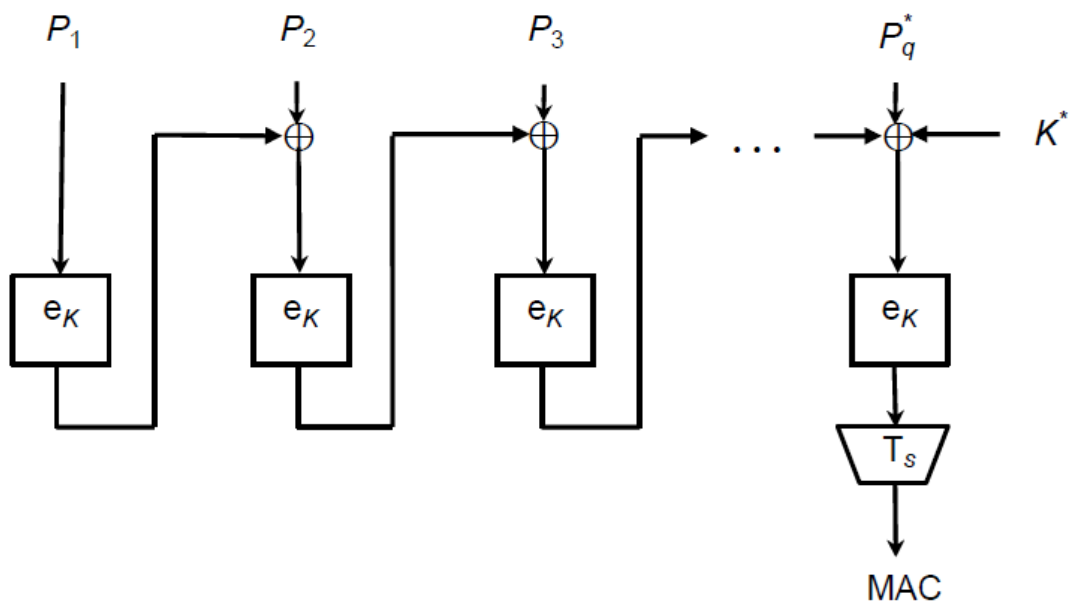


Рисунок 12 – Вычисление значения имитовставки – общий вид

Не рекомендуется использовать ключ режима выработки имитовставки в других криптографических алгоритмах.

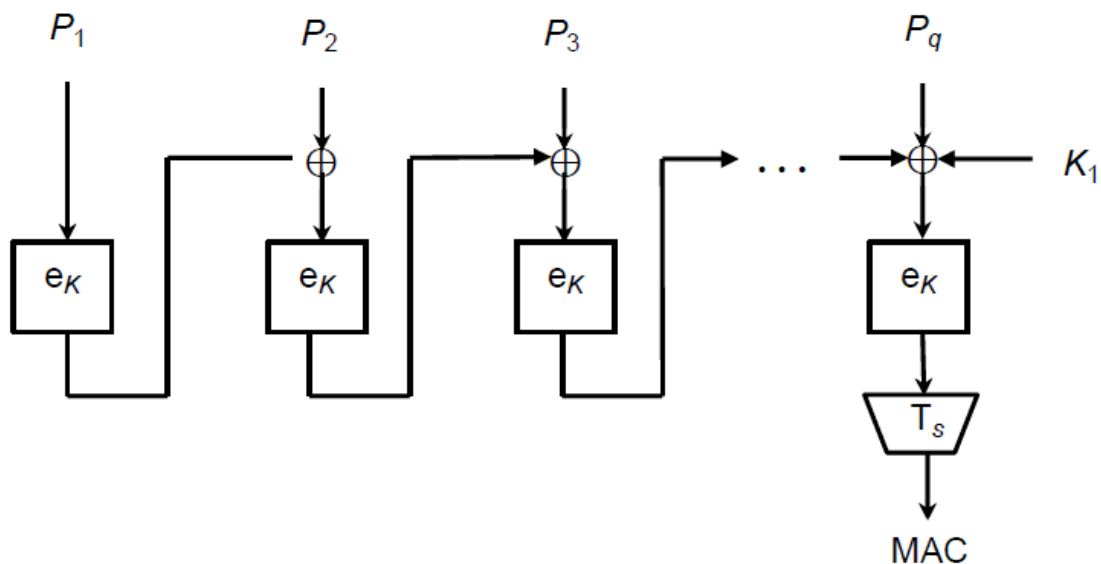


Рисунок 13 – Вычисление значения имитовставки – случай полного последнего блока

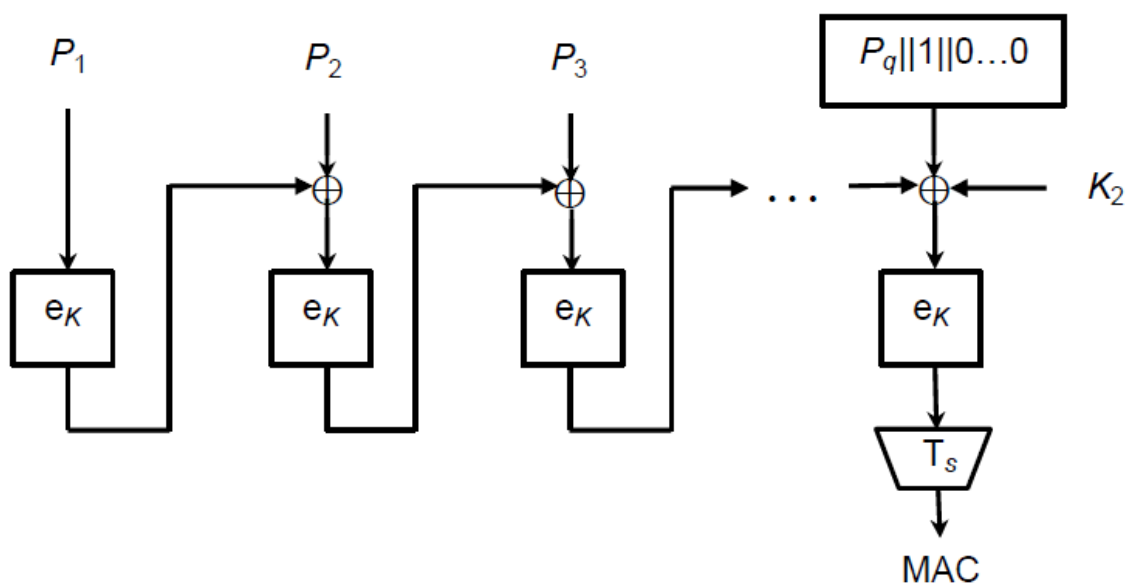


Рисунок 14 – Вычисление значения имитовставки – случай с дополнением последнего блока

1.4 Виды шифраторов

В настоящее время для шифрования данных наиболее широко применяют три вида шифраторов: аппаратные, программно-аппаратные и программные. Их основное различие заключается не только в способе реализации шифрования и степени надёжности защиты данных, но и в цене, что часто является для пользователей определяющим фактором. Самые дешёвые устройства шифрования — программные, затем — программно-аппаратные средства и, самыми дорогостоящими являются аппаратные. Несмотря на то, что цена аппаратных шифраторов существенно выше программных, разница в цене не сравнима с значительным повышением качества защиты информации.

Программное шифрование.

Любой криптографический алгоритм может быть реализован в виде соответствующей программы. Преимущества такой реализации очевидны: программные средства шифрования просты в использовании, легко копируются, их нетрудно модифицировать в соответствии с конкретными потребностями и как было сказано ранее весьма дешевы.

Все операционные системы имеют встроенные средства шифрования файлов. Обычно они используются для шифрования отдельных файлов, и работа с ключами возлагается на пользователя. Поэтому применение подобных средств требует особого внимания. Ни в коем случае нельзя хранить ключи на диске вместе с зашифрованными с их помощью файлами и незашифрованные копии файлов необходимо стереть сразу же после их шифрования.

Аппаратно-программные средства криптографической защиты информации сочетают гибкость программного решения с надёжностью аппаратного. При этом за счет гибкой программной оболочки можно быстро менять пользовательский интерфейс, переконфигурировать функции продукта, производить его конечную настройку; а аппаратные компоненты

позволяют защитить от модификации алгоритм криптографического примитива, обеспечить высокую защищенность ключевого материала и зачастую более высокую скорость работы.

Аппаратное шифрование – это процесс шифрования, который производится при помощи специализированных вычислительных устройств. Современные средства криптографической защиты информации нельзя отнести только к аппаратным, их было бы правильнее называть аппаратно-программными, однако, поскольку их программная часть неподконтрольна операционной системе, их часто называют аппаратными. Основной особенностью аппаратных СКЗИ является аппаратная реализация (за счет создания и применения специализированных процессоров) основных криптографических функций – различных криптографических преобразований, управления ключами, используемых криптографических протоколов и т. д.

Большое количество средств шифрования данных создаётся в виде специализированных физических устройств. Достоинства аппаратных шифраторов:

- аппаратный датчик случайных чисел генерирует случайные числа, используемые для формирования надёжных ключей шифрования и электронной цифровой подписи;
- целостность криптоалгоритма гарантируется его аппаратной реализацией;
- шифрование и хранение ключей происходит в самой плате шифратора, а не в оперативной памяти компьютера;
- загрузка ключей шифрования в шифрующее устройство с электронных ключей Touch Memory (i-Button) и смарт-карт производится напрямую, а не через системную шину компьютера и ОЗУ, это исключает возможность перехвата ключей;

- с помощью аппаратных шифраторов можно реализовать системы разграничения доступа к компьютеру и защиты информации от несанкционированного доступа;

- применение специализированного процессора для выполнения всех вычислений разгружает центральный процессор компьютера. Также можно установить нескольких аппаратных шифраторов на одном компьютере, что повышает скорость обработки информации (это преимущество присуще шифраторам для шины PCI);

- применение парафазных шин при создании шифрпроцессора исключает угрозу считывания ключевой информации по колебаниям электромагнитного излучения, возникающим при шифровании данных, в цепях «земля — питание».

При установке на компьютер специализированного шифровального оборудования будет возникать меньше проблем, чем при добавлении в системное программное обеспечение функций шифрования данных. В идеале шифрование должно производиться так, чтобы пользователь не замечал его. Чтобы осуществить это при помощи программных средств, функции шифрования должны быть замаскированы. Прodelать эту операцию в отлаженной операционной системой очень непросто, но при этом подсоединить шифровальное устройство к персональному компьютеру или к модему сможет любой, даже непрофессионал.

Виды устройств аппаратного шифрования. Современный рынок предлагает три разновидности аппаратных средств шифрования информации потенциальным покупателям:

- блоки шифрования в каналах связи;
- самодостаточные шифровальные модули (самостоятельно выполняют всю работу с ключами);
- шифровальные платы расширения для установки в персональные компьютеры.

Почти все устройства первых двух типов узко специализированны, поэтому нужно досконально исследовать ограничения, которые имеются у каждого устройства, прикладное программное обеспечение и операционные системы до того, как принимать конечное решение об их покупке. В противном случае можно зря потратить деньги, ни на шаг не приблизившись к желаемой цели. Правда, существуют компании, которые продают коммуникационное оборудование с уже заранее установленными устройствами аппаратного шифрования, что часто облегчает выбор.

Платы расширения для персональных компьютеров — наиболее универсальные средства аппаратного шифрования и, как правило, их очень легко настроить так, чтобы они шифровали всю информацию, записываемую на жёсткий диск или пересылаемую в порты и дисководы. Обычно защита от электромагнитного излучения в платах расширения для аппаратного шифрования отсутствует потому что бессмысленно защищать эти платы, если отсутствует защита всего компьютера.

2 Обзор ГОСТ Р 34.12-2015

2.1 Общие сведения

ГОСТ Р 34.12-2015 «Информационная технология. Криптографическая защита информации. Блочные шифры» утвержден приказом от 19 июня 2015 года № 749-ст. Стандарт вступил в действие с 1 января 2016 года. Шифр разработан Центром защиты информации и специальной связи ФСБ России с участием ОАО «Информационные технологии и коммуникационные системы» (ОАО «ИнфоТеКС»). Внесен Техническим комитетом по стандартизации ТК 26 «Криптографическая защита информации».

Блочный шифр «Кузнечик» – симметричный алгоритм блочного шифрования с длиной блока 128 бит и длиной ключа 256 бит, для генерации которого используется сеть Фейстеля.

Результатом алгоритма является блок шифрованного текста длиной 128 бит.

В качестве справочной информации доступны тестовые значения для проверки результатов – как функций, так и всего алгоритма.

В отличие от ГОСТ 28147-89 новый шифр представляет собой не сеть Фейстеля, а т.н. SP-сеть: преобразование, состоящее из нескольких одинаковых раундов, при этом каждый раунд состоит из нелинейного и линейного преобразований, а также операции наложения ключа. В отличие от сети Фейстеля, при использовании SP-сети преобразуется весь входной блок, а не его половина. Такая структура иногда также называется AES-like (похожей на AES), однако, в отличие от последнего у «Кузнечика» есть ряд своих особенностей:

- линейное преобразование может быть реализовано с помощью регистра сдвига;
- ключевая развертка реализована с помощью сети Фейстеля, в которой в качестве функции используется раундовое преобразование исходного алгоритма.

Шифрование основано на последовательном применении нескольких однотипных раундов, каждый из которых содержит три преобразования: сложение с раундовым ключом, преобразование блоком подстановок и линейное преобразование.

2.2 Алгоритм блочного шифрования с длиной блока $n = 128$ бит

2.2.1 Значение параметров.

Нелинейное биективное преобразование.

В качестве нелинейного биективного преобразования выступает подстановка $\pi = \text{Vec}_8 \pi' \text{Int}_8: V_8 \rightarrow V_8$, где $\pi': \mathbb{Z}_2^8 \rightarrow \mathbb{Z}_2^8$. Значения подстановки π' записаны ниже в виде массива $\pi' = (\pi'(0), \pi'(1), \dots, \pi'(255))$:

$\pi' = (252, 238, 221, 17, 207, 110, 49, 22, 251, 196, 250, 218, 35, 197, 4, 77, 233, 119, 240, 219, 147, 46, 153, 186, 23, 54, 241, 187, 20, 205, 95, 193, 249, 24, 101, 90, 226, 92, 239, 33, 129, 28, 60, 66, 139, 1, 142, 79, 5, 132, 2, 174, 227, 106, 143, 160, 6, 11, 237, 152, 127, 212, 211, 31, 235, 52, 44, 81, 234, 200, 72, 171, 242, 42, 104, 162, 253, 58, 206, 204, 181, 112, 14, 86, 8, 12, 118, 18, 191, 114, 19, 71, 156, 183, 93, 135, 21, 161, 150, 41, 16, 123, 154, 199, 243, 145, 120, 111, 157, 158, 178, 177, 50, 117, 25, 61, 255, 53, 138, 126, 109, 84, 198, 128, 195, 189, 13, 87, 223, 245, 36, 169, 62, 168, 67, 201, 215, 121, 214, 246, 124, 34, 185, 3, 224, 15, 236, 222, 122, 148, 176, 188, 220, 232, 40, 80, 78, 51, 10, 74, 167, 151, 96, 115, 30, 0, 98, 68, 26, 184, 56, 130, 100, 159, 38, 65, 173, 69, 70, 146, 39, 94, 85, 47, 140, 163, 165, 125, 105, 213, 149, 59, 7, 88, 179, 64, 134, 172, 29, 247, 48, 55, 107, 228, 136, 217, 231, 137, 225, 27, 131, 73, 76, 63, 248, 254, 141, 83, 170, 144, 202, 216, 133, 97, 32, 113, 103, 164, 45, 43, 9, 91, 203, 155, 37, 208, 190, 229, 108, 82, 89, 166, 116, 210, 230, 244, 180, 192, 209, 102, 175, 194, 57, 75, 99, 182).$

Значения подстановки π^{-1} записаны ниже в виде массива $\pi^{-1} = (\pi^{-1}(0), \pi^{-1}(1), \dots, \pi^{-1}(255))$:

$\pi^{-1} = (165, 45, 50, 143, 14, 48, 56, 192, 84, 230, 158, 57, 85, 126, 82, 145, 100, 3, 87, 90, 28, 96, 7, 24, 33, 114, 168, 209, 41, 198, 164, 63, 224, 39, 141, 12, 130, 234, 174, 180, 154, 99, 73, 229, 66, 228, 21, 183, 200, 6, 112, 157, 65, 117, 25,$

201, 170, 252, 77, 191, 42, 115, 132, 213, 195, 175, 43, 134, 167, 177, 178, 91, 70, 211, 159, 253, 212, 15, 156, 47, 155, 67, 239, 217, 121, 182, 83, 127, 193, 240, 35, 231, 37, 94, 181, 30, 162, 223, 166, 254, 172, 34, 249, 226, 74, 188, 53, 202, 238, 120, 5, 107, 81, 225, 89, 163, 242, 113, 86, 17, 106, 137, 148, 101, 140, 187, 119, 60, 123, 40, 171, 210, 49, 222, 196, 95, 204, 207, 118, 44, 184, 216, 46, 54, 219, 105, 179, 20, 149, 190, 98, 161, 59, 22, 102, 233, 92, 108, 109, 173, 55, 97, 75, 185, 227, 186, 241, 160, 133, 131, 218, 71, 197, 176, 51, 250, 150, 111, 110, 194, 246, 80, 255, 93, 169, 142, 23, 27, 151, 125, 236, 88, 247, 31, 251, 124, 9, 13, 122, 103, 69, 135, 220, 232, 79, 29, 78, 4, 235, 248, 243, 62, 61, 189, 138, 136, 221, 205, 11, 19, 152, 2, 147, 128, 144, 208, 36, 52, 203, 237, 244, 206, 153, 16, 68, 64, 146, 58, 1, 38, 18, 26, 72, 104, 245, 129, 139, 199, 214, 32, 10, 8, 0, 76, 215, 116).

Линейное преобразование.

Линейное преобразование задается отображением $\ell: V_8^{16} \rightarrow V_8$, которое определяется следующим образом:

$$\ell(a_{15}, \dots, a_0) = \nabla(148 \cdot \Delta(a_{15}) + 32 \cdot \Delta(a_{14}) + 133 \cdot \Delta(a_{13}) + 16 \cdot \Delta(a_{12}) + 194 \cdot \Delta(a_{11}) + 192 \cdot \Delta(a_{10}) + 1 \cdot \Delta(a_9) + 251 \cdot \Delta(a_8) + 1 \cdot \Delta(a_7) + 192 \cdot \Delta(a_6) + 194 \cdot \Delta(a_5) + 16 \cdot \Delta(a_4) + 133 \cdot \Delta(a_3) + 32 \cdot \Delta(a_2) + 148 \cdot \Delta(a_1) + 1 \cdot \Delta(a_0))$$

для любых $a_i \in V_8$, $i = 0, 1, \dots, 15$, где операции сложения и умножения осуществляются в поле $\mathbb{F}$, а константы являются элементами поля в указанном ранее смысле.

2.2.2 Преобразования.

1) 128-битный входной вектор очередного раунда складывается побитно с раундовым ключом:

$$X[k]: V_{128} \rightarrow V_{128}, \quad (14)$$

$$X[k](a) = k \oplus a, \text{ где } k, a \in V_{128}.$$

2) Нелинейное преобразование представляет собой применение к каждому 8-битному подвектору 128-битного входного вектора фиксированной подстановки:

$$S: V_{128} \rightarrow V_{128}, \quad (15)$$

$$S(a) = S(a_{15} \parallel \dots \parallel a_0) = \pi(a_{15} \parallel \dots \parallel a_0),$$

где $a = a_{15} \parallel \dots \parallel a_0 \in V_{128}, a_i \in V_8, i = 0, 1, \dots, 15$.

Преобразование, обратное к преобразованию S:

$$S^{-1}: V_{128} \rightarrow V_{128}, \quad (16)$$

$$S(a) = S^{-1}(a_{15} \parallel \dots \parallel a_0) = \pi^{-1}(a_{15} \parallel \dots \parallel a_0),$$

где $a = a_{15} \parallel \dots \parallel a_0 \in V_{128}, a_i \in V_8, i = 0, 1, \dots, 15$,

π^{-1} - подстановка обратная к подстановке π .

3) Линейное преобразование может быть реализовано не только как обычно в блочных шифрах — матрицей, но и с помощью РСЛОС — линейного 15 регистра сдвига с обратной связью, который движется 16 раз.

$$R: V_{128} \rightarrow V_{128}, \quad (17)$$

$$R(a) = R(a_{15} \parallel \dots \parallel a_0) = \ell(a_{15}, \dots, a_0) \parallel a_{15} \parallel \dots \parallel a_1,$$

где $a = a_{15} \parallel \dots \parallel a_0 \in V_{128}, a_i \in V_8, i = 0, 1, \dots, 15$.

$$L: V_{128} \rightarrow V_{128}, \quad (18)$$

$$L(a) = R^{16}(a), \text{ где } a \in V_{128}.$$

Обратное линейное преобразование:

$$R^{-1}: V_{128} \rightarrow V_{128}, \quad (19)$$

$$R^{-1}(a) = R^{-1}(a_{15} \parallel \dots \parallel a_0) = a_{14} \parallel a_{13} \parallel \dots \parallel a_0 \ell(a_{14}, a_{13}, \dots, a_0, a_{15}),$$

где $a = a_{15} \parallel \dots \parallel a_0 \in V_{128}, a_i \in V_8, i = 0, 1, \dots, 15$.

$$L^{-1}: V_{128} \rightarrow V_{128}, \quad (20)$$

$$L^{-1}(a) = (R^{-1})^{16}(a), \text{ где } a \in V_{128}.$$

4)

$$F[k]: V_{128} \times V_{128} \rightarrow V_{128} \times V_{128}, \quad (21)$$

$$F[k](a_1, a_0) = (LSX[k](a_1) \oplus a_0, a_1), \text{ где } k, a_0, a_1 \in V_{128}.$$

2.2.3 Алгоритм развертывания ключа. Процедура генерации раундовых ключей из мастера ключа. Первые два ключа получаются разбиением мастер ключа пополам. Далее для выработки очередной пары раундовых ключей используется 8 итерации сети Фейстеля, где, в свою очередь, в качестве раундовых ключей используется счетчиковая последовательность, прошедшая через линейное преобразование алгоритма:

$$C_i = L(Vec_{128}(i)), i = 1, 2, \dots, 32. \quad (22)$$

$$(K_{2i+1}, K_{2i+2}) = F[C_{8(i-1)+8}] \dots F[C_{8(i-1)+1}](K_{2i-1}, K_{2i}), i = 1, 2, 3, 4. \quad (23)$$

Алгоритм развертывания ключа предоставлен на рисунке 15.

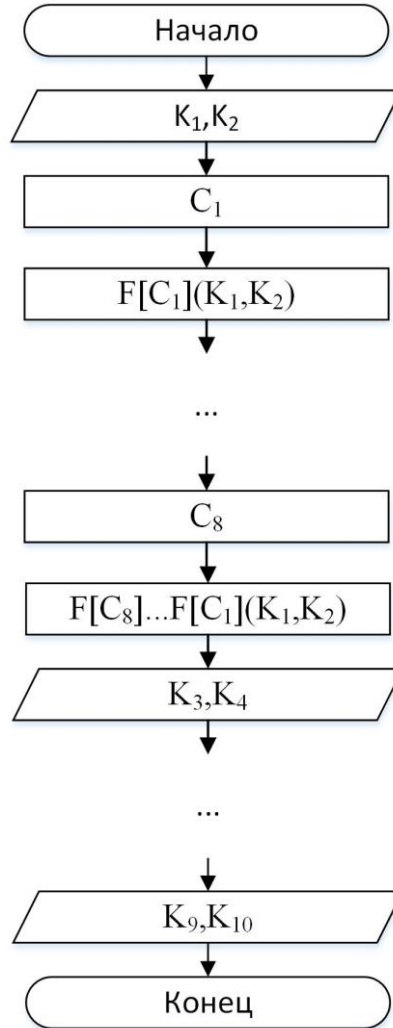


Рисунок 15 – Алгоритм развертывания ключа

2.2.4 Алгоритм зашифрования. Алгоритм зашифрования в зависимости от значений итерационных ключей $K_i \in V_{128}, i = 1, 2, \dots, 10$, реализует подстановку $E_{K_1, \dots, K_{10}}$, заданную на множестве V_{128} в соответствии с равенством

$$E_{K_1, \dots, K_{10}}(a) = X[K_{10}]LXS[K_9] \dots LXS[K_2]LXS[K_1](a), \text{ где } a \in V_{128} \quad (24)$$

Алгоритм зашифрования представлен на рисунке 16.

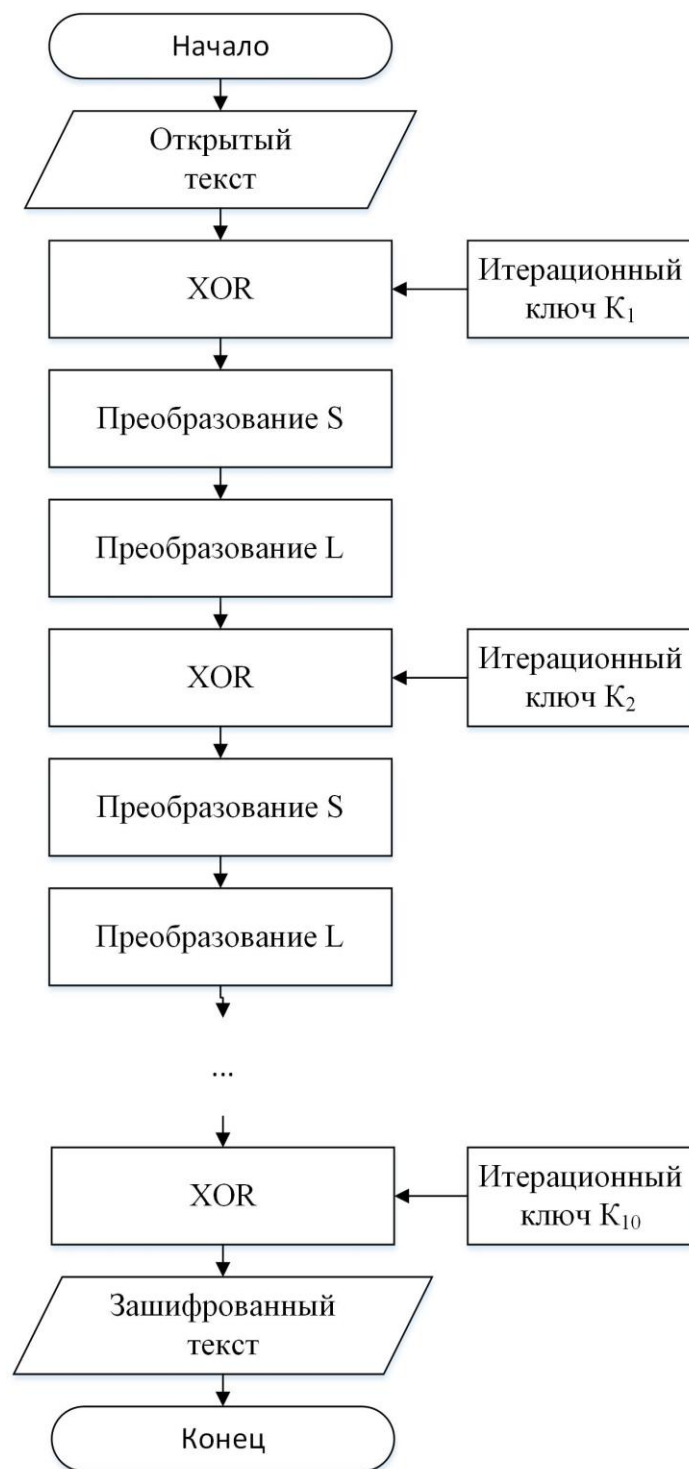


Рисунок 16 – Алгоритм зашифрования

2.2.5 Алгоритм расшифрования. Алгоритм расшифрования в зависимости от значений итерационных ключей $K_i \in V_{128}, i = 1, 2, \dots, 10$, реализует подстановку $D_{K_1, \dots, K_{10}}$, заданную на множестве V_{128} в соответствии с равенством

$$D_{K_1, \dots, K_{10}}(a) = X[K_1]S^{-1}L^{-1}X[K_2] \dots S^{-1}L^{-1}X[K_9]S^{-1}L^{-1}X[K_{10}](a), \quad (25)$$

где $a \in V_{128}$.

Алгоритм расшифрования представлен на рисунке 17.

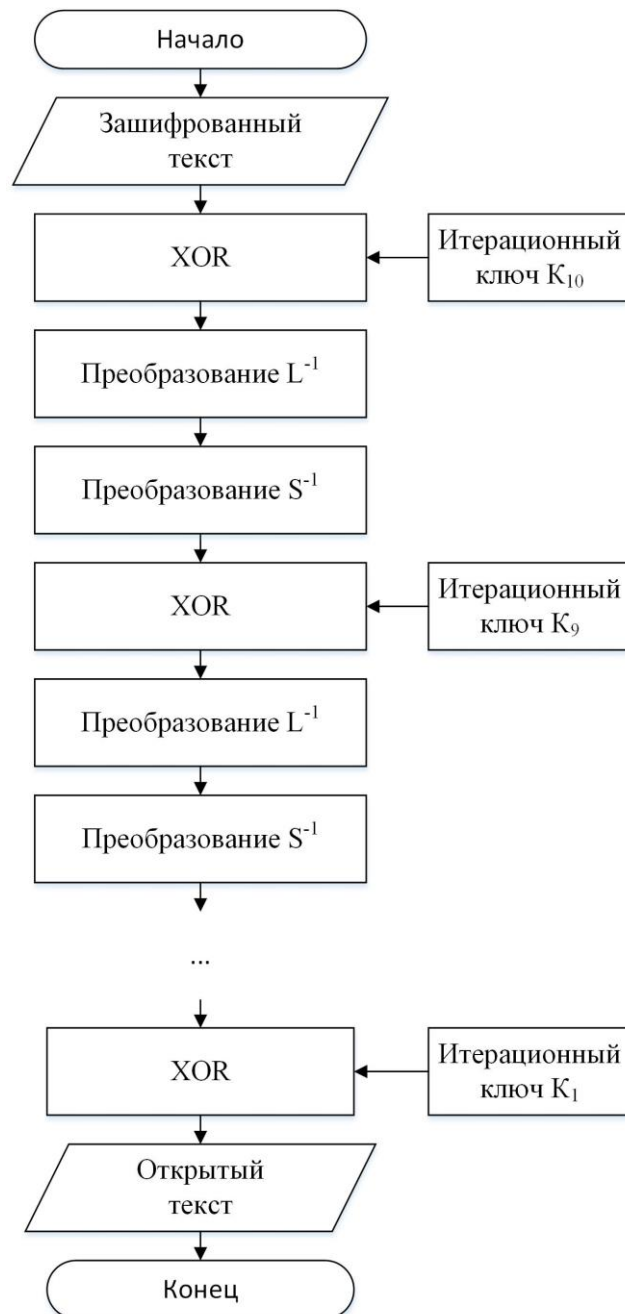


Рисунок 17 – Алгоритм расшифрования

3 Реализация алгоритма блочного шифрования с длиной блока $n=128$ бит

3.1 Описание модели аппаратного модуля

Модель аппаратного модуля реализует алгоритм криптографического преобразования ГОСТ Р 34.12 – 2015 с длиной блока $n=128$ бит в режиме простой замены, на базисных элементах И, ИЛИ, НЕ, на языке описания аппаратуры VHDL.

Для загрузки ключевой информации в модель аппаратного модуля была выбрано асинхронная шина. Для того чтобы можно было загрузить или удалить ключ вне зависимости от тактового сигнала.

Так же можно использовать эту модель при создании криптопроцессора или же можно загрузить проект на плату ПЛИС. Например, на Xilinx Spartan – 3.

3.2 Обоснование выбора языка описания аппаратуры

До настоящего времени одним из самых трудоёмких этапов проектирования электронных устройств являлось создание электрических схем. Это связано с тем, что множество электронных компонентов и связей между ними приходится вводить вручную. После завершения проектирования схемы много времени занимает проверка ее правильности. САПР (системы автоматизированного проектирования) улучшили ситуацию, но и они также отнимают крайне много времени на верификацию проектов. Модернизация таких схем зачастую становится попросту невозможной без полного переделывания конструкции печатной платы. Современность требует четкого и емкого описания схем, именно поэтому три десятилетия назад при проектировании СБИС (сверх больших интегральных схем) от создания логических схем стали отказываться. Иногда доходило до того, что электрическая схема кристалла размером 1×1 см занимала несколько тысяч листов формата А3.

Язык Very high speed integrated circuits Hardware Description Language (VHDL) был разработан в 1983 г. по заказу Министерства Обороны США. Он был создан для описания логических схем. VHDL является стандартным языком с 1987 года, а в 1993 г. были введены многие усовершенствования. Наряду с языком Verilog, VHDL программирование является базовым для разработки логических схем. Однако, в России в настоящее время программирование на VHDL распространено в намного большей степени, чем работа с Verilog.

Язык VHDL – это весьма необычный язык, как для программистов, так и для разработчиков электронной аппаратуры. Это связано с тем, что по своей сути VHDL является языком параллельного программирования.

Преимущества VHDL программы:

- с помощью VHDL гораздо проще работать с большими проектами. Десятью строками VHDL можно описать как один, так и 100 000 триггеров;
- VHDL проекты нет необходимости макетировать. Достаточно запустить их VHDL-код через симулятор HDL;
- VHDL программы не требуют представления в виде схемы;
- VHDL программу достаточно легко проверить;
- VHDL программы надёжны. Синтаксический анализ, программное моделирование и компиляция в логическую схему позволяют быстро выявлять ошибки в проекте;
- VHDL программы универсальны. Любой VHDL-компонент может быть использован во множестве других проектов. При этом функции и параметры этого компонента можно подстраивать под новые задачи на любом этапе проектирования. Помимо этого универсальность заключается в том, что VHDL проекты с одной элементной базы можно переносить на другую (например, микросхемы различных технологий).

На Западе наибольшее распространение имеет язык Verilog. Двадцать лет назад этот язык одержал победу в конкурентной борьбе с другими HDL языками благодаря тому, что ему требовалось мало вычислительных ресурсов.

VHDL более универсальный и быстрый язык, но он проиграл в быстродействии языку Verilog, особенно при моделировании на уровне вентилей.

VHDL код долговечен, в отличие от электрической схемы, которая всегда разрабатывается под конкретную элементную базу. Обычно по прошествии 3-5 лет элементная база сменяется, сменяются и электрические схемы, использующие её, а техническое решение на языке VHDL может быть использовано в течении десятилетий.

3.3 Описание среды разработки ModelSim10

При этом возрастает роль средств моделирования в процессе разработки устройств, меняются методы и требования, предъявляемые к процедурам верификации. Для своевременного обнаружения возможных ошибок средства моделирования должны обеспечивать возможность контроля результатов каждого этапа процесса проектирования: создания исходных HDL-описаний, синтеза, размещения и их трассировки в кристалл. При данном подходе время, затрачиваемое на разработку устройства, минимальна, а также сокращается стоимость процесса, так как цена ошибки возрастает с каждым последующим шагом проектирования.

Пакет программных средств ModelSim от корпорации Model Technology (одного из подразделений компании Mentor Graphics) в настоящее время является одной из наиболее распространенных систем HDL-моделирования.

Назначение и основные характеристики пакета программных средств ModelSim:

- полная поддержка основных стандартов языков VHDL и Verilog и их расширений (Verilog 2001, VHDL2002);
- полная совместимость со спецификациями 1.0 — 3.0 стандартного формата описания задержек SDF (Standard Delay Format) обеспечивает возможность обратной аннотации временных параметров. Кроме этого, пакет ModelSim удовлетворяет требованиям VCD (Value Change Dump) по

формированию стандартных выходных векторов для VHDL и Verilog и PLI (Programming Language Interface);

- поддержка инновационных технологий и языков моделирования аппаратуры, таких как: System C 2.0, SystemVerilog, PSL, и т.д;

- единое ядро моделирования пакета SKS (Single Kernel Simulation), обеспечивающее возможность полной отладки проектов, одновременно содержащих модули, написанные на VHDL и Verilog и др;

- высокая скорость компиляции и моделирования (по сравнению с другими пакетами моделирования ActiveHDL, NC-Verilog), обеспечивающая минимальное время отладки систем с различными уровнями сложности. Одним из главных факторов, повышающих производительность, является использование принципа оптимизированной прямой компиляции. В соответствии с этим принципом исходные (VHDL, Verilog, System C) описания компилируются в машинно-независимый объектный код, который будет исполняться на любой поддерживаемой платформе;

- открытая архитектура программных средств ModelSim, обеспечивающая тесную интеграцию с пакетами САПР. Пользователь может выполнять этапы моделирования фактически в рамках основной системы проектирования, в среде которой осуществляется процесс разработки устройства;

- средства управления пользовательским интерфейсом Tool command language (Tcl) и Tool kit (Tk) предоставляют возможность прямого доступа к моделирующему ядру ModelSim, загрузки информации о выполнении процесса моделирования и его результатов в среду используемой САПР и управления работой системы ModelSim через интерфейс используемых средств проектирования. Возможен и противоположный способ интеграции с программным обеспечением других компаний, в этом случае в качестве основной системы используется ModelSim, интерфейсная оболочка которой адаптируется для управления выбранным пакетом САПР;

- наличие защищенного режима компиляции моделей, гарантирующего выполнение требований, связанных с охраной интеллектуальной собственности, к которой относятся коммерчески распространяемые модули (IP Core). В обычном режиме разрешается полная отладка моделей с доступом к исходному коду и внутренней структуре объекта. Есть возможность использования интеллектуальных продуктов, распространяемых производителем в скомпилированном виде без передачи исходного кода модели;

- встроенный индикатор активности кода (анализатор тестового покрытия), повышающий эффективность отладки проекта, а также позволяющий быстро создавать надежные тестовые эксперименты. Этот инструмент предоставляет возможность выявить строки исходного кода, которые не используются в процессе моделирования, и выводит в графической форме отчет обо всех файлах проекта. Индикатор активности кода может быть использован и для любого отдельного блока и для всей системы в целом. Имеется встроенный анализатор производительности, позволяющий повысить скорость моделирования за счет обнаружения в проекте и последующего устранения факторов, оказывающих отрицательное влияние на быстродействие этого процесса. С помощью этого инструмента можно получить информацию о библиотечных элементах, обработка которых требует больших временных затрат, фрагментах исходного кода, написанных нерационально с точки зрения скорости его исполнения, неиспользуемых сигналах в выводимых списках, а также избыточном коде в тестовых последовательностях. Исключение перечисленных элементов из проекта позволяет снизить время моделирования;

- возможность работы в разных режимах, в том числе и пакетном. Разработав и отладив сценарий моделирования в интерактивном режиме, можно оформить его для последующего использования в виде пакетного командного файла;

- доступный и простой для понимания разработчиком визуальный пользовательский интерфейс, а также наличие подробной справочной системы, сокращают время освоения пакета моделирования. Средства управления пользовательским интерфейсом Tcl и Tk позволяют выполнить настройку его элементов (панелей кнопок, меню) в соответствии с требованиями каждого конкретного пользователя.

3.4 Разработка модели аппаратного модуля

Реализация алгоритма проводилась на языке описания аппаратуры VHDL в среде ModelSim SE v.10.1с. Листинг программы представлен в приложении А.

Описание функции программы:

- функция func_x.

На вход функции подается переменные x и y длиной 128 бит. На выходе получаем переменную длиной 128 бит.

Функция представляет собой сложение по модулю два двух переменных x и y. Сложение по модулю два на базисе И, ИЛИ, НЕ двух переменных на языке VHDL имеет следующий вид $((\text{not } x) \text{ and } y) \text{ or } (x \text{ and } (\text{not } y))$.

- функция func_s.

На вход функции подается переменная z длиной 128 бит. На выходе получаем переменную длиной 128 бит.

Функция представляет собой подстановку к каждому 8-битному подвектору 128-битного входного вектора фиксированной подстановки.

Для реализации функции на базисе И, ИЛИ, НЕ была составлена таблица истинности. Таблица истинности представлена в приложении Б.

Значения $x_0 - x_7$, это все возможные комбинации из 8 бита те позиции в таблице подстановки, а значения $y_0 - y_7$ их значения в двоичном виде.

Далее для каждого выходного бита было составлено карта Карно из 256 элементов. Пример для 0 элемента предоставлена на рисунке 18.

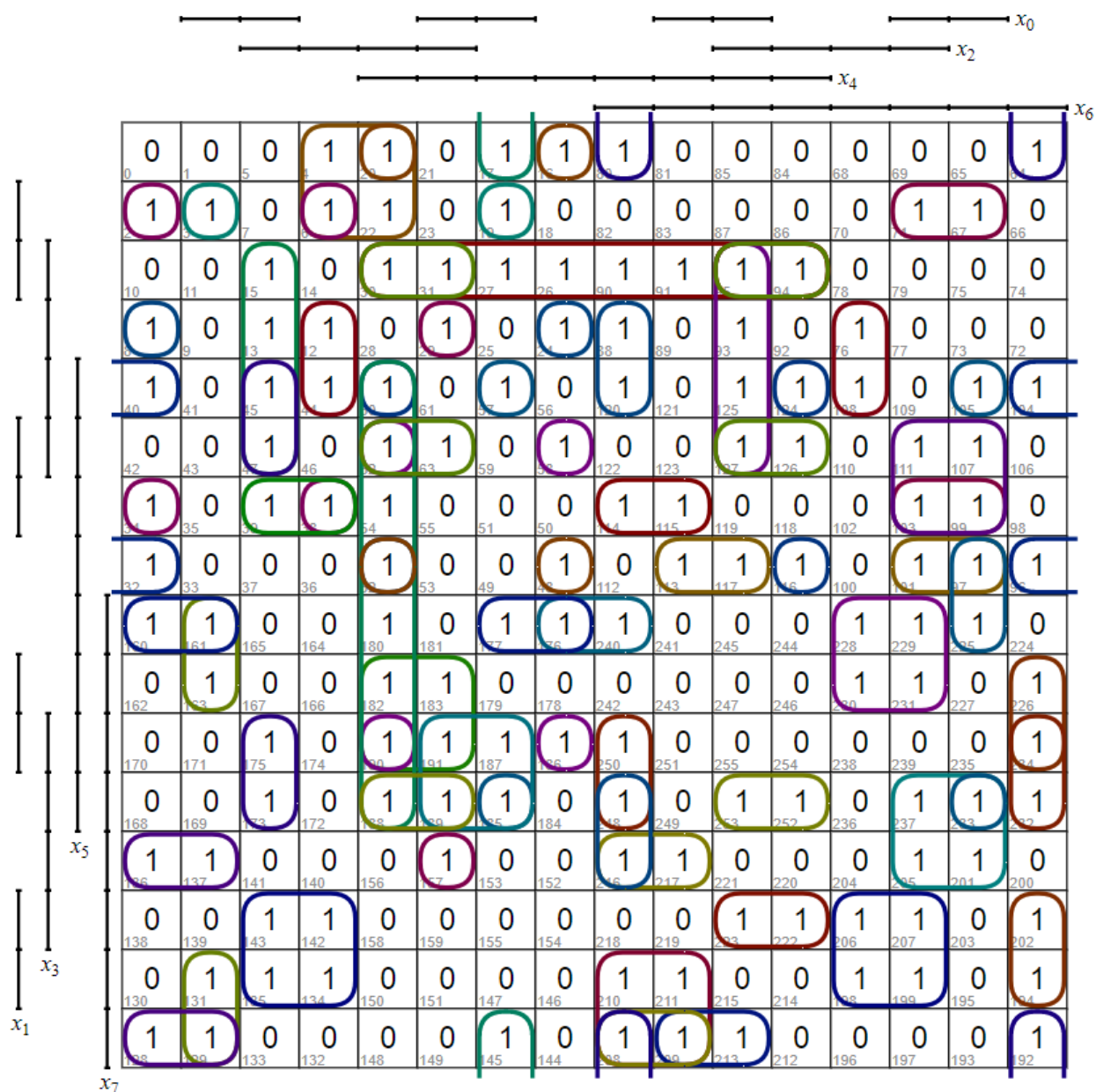


Рисунок 18 – Карта Карно для нулевого бита

Из карты Карно было выведено выражение для нулевого бита.

$$y_0 = (\bar{x}_7 \wedge \bar{x}_5 \wedge x_4 \wedge x_3 \wedge x_1) \vee (x_7 \wedge \bar{x}_5 \wedge \bar{x}_4 \wedge x_2 \wedge x_1) \vee (x_7 \wedge x_6 \wedge x_5 \wedge \bar{x}_4 \wedge \bar{x}_3 \wedge x_2) \vee (x_7 \wedge x_6 \wedge \bar{x}_5 \wedge x_4 \wedge \bar{x}_3 \wedge \bar{x}_2) \vee (x_7 \wedge \bar{x}_6 \wedge x_5 \wedge x_4 \wedge x_2 \wedge x_1) \vee (x_7 \wedge \bar{x}_6 \wedge \bar{x}_4 \wedge \bar{x}_3 \wedge \bar{x}_2 \wedge x_0) \vee (\bar{x}_7 \wedge \bar{x}_4 \wedge x_3 \wedge x_2 \wedge \bar{x}_1 \wedge \bar{x}_0) \vee (\bar{x}_7 \wedge x_6 \wedge x_5 \wedge x_4 \wedge \bar{x}_3 \wedge \bar{x}_2 \wedge x_1) \vee (x_7 \wedge x_6 \wedge \bar{x}_5 \wedge x_4 \wedge \bar{x}_3 \wedge \bar{x}_1 \wedge x_0) \vee (\bar{x}_6 \wedge x_5 \wedge x_4 \wedge x_3 \wedge \bar{x}_2 \wedge \bar{x}_1 \wedge x_0) \vee (\bar{x}_6 \wedge x_5 \wedge x_4 \wedge x_2 \wedge \bar{x}_0) \vee (x_5 \wedge x_4 \wedge x_3 \wedge x_2 \wedge \bar{x}_0) \vee (x_6 \wedge \bar{x}_5 \wedge x_4 \wedge x_3 \wedge x_2 \wedge x_1) \vee (x_7 \wedge x_6 \wedge \bar{x}_5 \wedge x_4 \wedge \bar{x}_2 \wedge \bar{x}_1) \vee (x_7 \wedge \bar{x}_6 \wedge \bar{x}_5 \wedge \bar{x}_4 \wedge \bar{x}_2 \wedge \bar{x}_1) \vee (x_7 \wedge \bar{x}_6 \wedge x_5 \wedge x_4 \wedge x_3 \wedge x_0) \vee (\bar{x}_7 \wedge x_6 \wedge x_4 \wedge x_3 \wedge x_2 \wedge x_0) \vee (\bar{x}_7 \wedge \bar{x}_6 \wedge \bar{x}_4 \wedge x_3 \wedge x_2 \wedge x_0) \vee (\bar{x}_7 \wedge x_6 \wedge x_5 \wedge \bar{x}_4 \wedge x_1 \wedge x_0) \vee (\bar{x}_7 \wedge x_6 \wedge \bar{x}_4 \wedge \bar{x}_3 \wedge x_1 \wedge x_0) \vee (\bar{x}_6$$

$$\begin{aligned} & \wedge x_5 \wedge x_3 \wedge x_2 \wedge x_1 \wedge x_0) \vee (x_7 \wedge x_6 \wedge \bar{x}_4 \wedge x_3 \wedge \bar{x}_1 \wedge x_0) \vee (\bar{x}_7 \wedge x_6 \wedge x_5 \wedge \bar{x}_3 \wedge \\ & \bar{x}_1 \wedge x_0) \vee (x_7 \wedge x_5 \wedge x_3 \wedge x_2 \wedge \bar{x}_1 \wedge x_0) \vee (x_6 \wedge x_5 \wedge \bar{x}_4 \wedge \bar{x}_2 \wedge \bar{x}_1 \wedge x_0) \vee (x_7 \wedge \\ & \bar{x}_6 \wedge \bar{x}_3 \wedge \bar{x}_2 \wedge \bar{x}_1 \wedge x_0) \vee (\bar{x}_7 \wedge \bar{x}_6 \wedge \bar{x}_5 \wedge \bar{x}_3 \wedge x_2 \wedge \bar{x}_0) \vee (x_7 \wedge x_6 \wedge x_5 \wedge x_3 \wedge \bar{x}_2 \\ & \wedge \bar{x}_0) \vee (\bar{x}_6 \wedge x_5 \wedge x_4 \wedge x_3 \wedge x_1 \wedge \bar{x}_0) \vee (\bar{x}_7 \wedge \bar{x}_6 \wedge \bar{x}_4 \wedge \bar{x}_3 \wedge x_1 \wedge \bar{x}_0) \vee (x_7 \wedge x_6 \\ & \wedge \bar{x}_4 \wedge \bar{x}_2 \wedge x_1 \wedge \bar{x}_0) \vee (\bar{x}_7 \wedge x_5 \wedge x_4 \wedge x_2 \wedge \bar{x}_1 \wedge \bar{x}_0) \vee (\bar{x}_7 \wedge x_5 \wedge \bar{x}_4 \wedge \bar{x}_2 \wedge \bar{x}_1 \wedge \\ & \bar{x}_0) \vee (x_6 \wedge x_4 \wedge x_3 \wedge \bar{x}_2 \wedge \bar{x}_1 \wedge \bar{x}_0) \vee (\bar{x}_6 \wedge x_5 \wedge \bar{x}_3 \wedge \bar{x}_2 \wedge \bar{x}_1 \wedge \bar{x}_0) \vee (x_6 \wedge \bar{x}_5 \wedge \\ & \bar{x}_3 \wedge \bar{x}_2 \wedge \bar{x}_1 \wedge \bar{x}_0) \vee (\bar{x}_7 \wedge \bar{x}_6 \wedge x_5 \wedge \bar{x}_4 \wedge \bar{x}_3 \wedge x_2 \wedge x_1) \vee (\bar{x}_7 \wedge \bar{x}_6 \wedge \bar{x}_5 \wedge x_4 \wedge \bar{x}_3 \\ & \wedge \bar{x}_2 \wedge \bar{x}_1) \vee (\bar{x}_7 \wedge \bar{x}_6 \wedge \bar{x}_5 \wedge \bar{x}_3 \wedge \bar{x}_2 \wedge x_1 \wedge x_0) \vee (\bar{x}_6 \wedge \bar{x}_5 \wedge x_4 \wedge x_3 \wedge x_2 \wedge \bar{x}_1 \wedge \\ & x_0) \vee (\bar{x}_7 \wedge \bar{x}_6 \wedge \bar{x}_5 \wedge x_3 \wedge \bar{x}_2 \wedge \bar{x}_1 \wedge \bar{x}_0) \vee (x_7 \wedge x_5 \wedge x_4 \wedge \bar{x}_3 \wedge \bar{x}_2 \wedge \bar{x}_1 \wedge \bar{x}_0). \end{aligned}$$

Далее был произведен расчет до 7 бита. Остальные биты повторяются.

- функция func_so.

Данная функция реализована аналогичным образом как функция func_s.

Различие между функциями только в таблице истинности.

- функция func_r.

На вход функции подается переменная z длиной 128 бит. На выходе получаем переменную длиной 128 бит.

В эталонной модели на языке C++ эта функция реализована следующим образом.

Для реализации операции умножения в поле Галуа используется блок памяти размером 256 на 256, который содержит все возможные результаты умножения на поле.

Результат с функции разбивается на 16 байт и каждый байт считывается по очереди начиная с пятнадцатого до нулевого, затем считанный байт приводится к целому типу и полученное число умножается на 256. К нему прибавляется константа из {148, 32, 133, 16, 194, 192, 1, 251, 1, 192, 194, 16, 133, 32, 148, 1} в зависимости от считанного байта: если пятнадцатый байт считывается, то прибавляется константа 148, если четырнадцатый байт, то прибавляется 32 и так далее до нулевого байта. Полученное число является позицией числа в таблице умножения. Из позиции считается число, это и будет

результатом умножения. Байты складываются между собой по модулю два, полученный байт будет результатом линейного преобразования.

Далее все 16 байта сдвигаются вправо на байт, а полученное результат из линейного преобразования записывается на место пятнадцатого байта.

Для реализации данной функции на базисе И, ИЛИ, НЕ были составлены таблицы истинности для каждой константы за исключением константы 1 так как умножение на 1 в поле $GF(2)$ дает тот же результат.

Далее были составлены карты Карно для каждого выходного бита, а из них были выведены выражения. Полученные выражения складываются по модулю два, полученный байт будет результатом линейного преобразования.

Далее все 16 байт сдвигаются вправо на байт, а полученный результат из линейного преобразования записывается на место пятнадцатого байта.

- функция `func_ro`.

Данная функция реализована аналогичным способом, как функция `func_r`. Исключением является порядок изменения констант {1, 148, 32, 133, 16, 194, 192, 1, 251, 1, 192, 194, 16, 133, 32, 148}, сдвиг происходит в лево на байт и в результате чего линейное преобразование записывается на нулевой байт.

- функция `func_l`.

На вход функции подается переменная x длиной 128 бит. На выходе функции получаем переменную длиной 128 бит.

Функция представляет собой повторение функции `func_r` 16 раз.

- функция `func_lo`.

Реализовано аналогичным способом, как функция `func_l`. Исключением будет повторение функции `func_ro` 16 раз.

- функция `func_lsx`.

Функция представляет собой поочерёдное применение функции `func_x`, `func_s`, `func_l`.

- функция `func_lsxo`.

Функция представляет собой поочерёдное применение функции x , `func_x`, `func_lo`, `func_so`

- функция `func_f`.

На вход функции подается переменные x длиной 256 бит и константа длиной 128 бит. На выходе функции получаем переменную длиной 256 бит.

В функции, входная переменная x делится на 2 части `key1` и `key2`.

`Key1` равна второй половине x переменного, а `Key2` равна первой половине x переменного.

Далее вызывается функция `func_lsx` на вход которого подается переменные `key1` и `s`. Далее происходит сложение по модулю два, переменной `key2` и результата функции `func_ls`. Результат сложения по модулю два будет равен второй половине выходной переменной, а первой половиной будет `key1`.

- функция `func_key`.

На вход функции подается переменная x длиной 128 бит и 8 констант. На выходе функции получаем переменную длиной 256 бит.

Функция представляет собой повторение функции `func_f` 8 раз.

- функция `func_za`.

На вход функции подается переменная открытого текста длиной 128 бит и 10 итерационных ключей, а на выходе получится шифртекст длиной 128 бит.

В данной функции первый итерационный ключ и открытый текст обрабатывается в функции `lsx`, далее берется результат и второй итерационный ключ, и обрабатывается в функции `lsx` до девятого ключа. Затем происходит сложение по модулю два результата последней функции `lsx` с десятым итерационным ключом. В результате получим шифртекст.

- функция `func_ras`.

На вход этой функции подается переменная шифртекст длиной 128 бит и 10 итерационных ключей, а на выходе получится открытый текст длиной 128 бит.

В данной функции десятый итерационный ключ и шифртекст обрабатывается в функции `lsx`, далее берется результат и первый итерационный ключ и обрабатывается в функции `lsx` до восьмого ключа. Затем

происходит сложение по модулю два результата последней функции `lsx` с десятым итерационным ключом. В результате получим открытый текст.

Описание процесса программы:

Для генерирования итерационных ключей необходимо подать на порт `key` ключевую информацию длиной 256 бит и на порт `key_download` подать 1.

Если подать 0 на порт `key_delete` то все значения итерационных ключей будет равной нулю.

Синхронизация происходит по фронту сигнала `clk` и если подать на порт `mode` единицу, то программа начнет зашифровать сигнал из порта `data_in` и выводить будет на порт `data_out`. Если же подать на порт `mode` нуль, то программа будет расшифровать значения из порта `data_in` и выводить будет на порт `data_out`.

4 Тестирование модели аппаратного модуля, с помощью набора тестовых последовательностей

Для проверки правильности работы структурного описания устройства был написан испытательный стенд на языке VHDL. Листинг программы испытательного стенда представлен в приложении Б.

4.1 Тестирование S, R и L преобразований

На рисунке 19 представлены временные диаграммы для проверки S, R и L преобразовании. Сигналы представлены в виде шестнадцатеричной системы счисления.

Описание сигналов:

- proverka_s. Сигнал для проверки сигнала proverka_s1;
- proverka_s1-4. Результаты S преобразования;
- proverka_so1-4. Результаты обратного S преобразования;
- proverka_r1-4. Результаты R преобразования;
- proverka_ro1-4. Результаты обратного R преобразования;
- proverka_l1-4. Результаты L преобразования;
- proverka_lo1-4. Результаты обратного L преобразования.

	Msgs	
+ /eeee_tb/DUT/proverka_s	FFEEDDCCB8AA99881122334455667700	FFEEDDCCB8AA99881122334455667700
+ /eeee_tb/DUT/proverka_so1	FFEEDDCCB8AA99881122334455667700	FFEEDDCCB8AA99881122334455667700
+ /eeee_tb/DUT/proverka_s1	B66CD8887D38E8D77765AEEA0C9A7EFC	B66CD8887D38E8D77765AEEA0C9A7EFC
+ /eeee_tb/DUT/proverka_so2	B66CD8887D38E8D77765AEEA0C9A7EFC	B66CD8887D38E8D77765AEEA0C9A7EFC
+ /eeee_tb/DUT/proverka_s2	559D8DD7BD06CBFE7E7B262523280D39	559D8DD7BD06CBFE7E7B262523280D39
+ /eeee_tb/DUT/proverka_so3	559D8DD7BD06CBFE7E7B262523280D39	559D8DD7BD06CBFE7E7B262523280D39
+ /eeee_tb/DUT/proverka_s3	0C3322FED531E4630D80EF5C5A81C50B	0C3322FED531E4630D80EF5C5A81C50B
+ /eeee_tb/DUT/proverka_so4	0C3322FED531E4630D80EF5C5A81C50B	0C3322FED531E4630D80EF5C5A81C50B
+ /eeee_tb/DUT/proverka_s4	23AE65633F842D29C5DF529C13F5ACDA	23AE65633F842D29C5DF529C13F5ACDA
+ /eeee_tb/DUT/proverka_r	000000000000000000000000000000100	000000000000000000000000000000100
+ /eeee_tb/DUT/proverka_ro1	000000000000000000000000000000100	000000000000000000000000000000100
+ /eeee_tb/DUT/proverka_r1	940000000000000000000000000000001	940000000000000000000000000000001
+ /eeee_tb/DUT/proverka_ro2	940000000000000000000000000000001	940000000000000000000000000000001
+ /eeee_tb/DUT/proverka_r2	A59400000000000000000000000000000	A59400000000000000000000000000000
+ /eeee_tb/DUT/proverka_ro3	A59400000000000000000000000000000	A59400000000000000000000000000000
+ /eeee_tb/DUT/proverka_r3	64A594000000000000000000000000000	64A594000000000000000000000000000
+ /eeee_tb/DUT/proverka_ro4	64A594000000000000000000000000000	64A594000000000000000000000000000
+ /eeee_tb/DUT/proverka_r4	0D64A5940000000000000000000000000	0D64A5940000000000000000000000000
+ /eeee_tb/DUT/proverka_lo1	64A594000000000000000000000000000	64A594000000000000000000000000000
+ /eeee_tb/DUT/proverka_l1	D456584DD0E3E84CC3166E487FA2890D	D456584DD0E3E84CC3166E487FA2890D
+ /eeee_tb/DUT/proverka_lo2	D456584DD0E3E84CC3166E487FA2890D	D456584DD0E3E84CC3166E487FA2890D
+ /eeee_tb/DUT/proverka_l2	79D26221B87B584CD42FBC4FFE5DE9A	79D26221B87B584CD42FBC4FFE5DE9A
+ /eeee_tb/DUT/proverka_lo3	79D26221B87B584CD42FBC4FFE5DE9A	79D26221B87B584CD42FBC4FFE5DE9A
+ /eeee_tb/DUT/proverka_l3	0E93691A0CFC60408B7B68F66B513C13	0E93691A0CFC60408B7B68F66B513C13
+ /eeee_tb/DUT/proverka_lo4	0E93691A0CFC60408B7B68F66B513C13	0E93691A0CFC60408B7B68F66B513C13
+ /eeee_tb/DUT/proverka_l4	E6A8094FEE0AA204FD97BCB08448B580	E6A8094FEE0AA204FD97BCB08448B580

Рисунок 19 – Временные диаграммы S, R и L преобразований

При сравнении результатов тестирования со значениями справочной информации ГОСТ Р 34.12 - 2015 был сделан вывод, что преобразования работают правильно.

4.2 Тестирование алгоритма развертывания ключа

На рисунке 20 представлены временные диаграммы для проверки алгоритма развертывания ключа. Сигналы представлены в виде шестнадцатеричной системы счисления.

Описание сигналов:

- key. Входной ключ с длиной 256 бит;
- key1-10. Сгенерированные итерационные ключи;
- const. Первая итерационная константа;
- x. Результат X преобразования между сигналами const и key1;
- sx. Результат S преобразования сигнала x;

- l_{sx}. Результат L преобразования сигнала s_x;
- f. результат F преобразования между сигналами key и const.

Signal	Value
/eeee_tb/DUT/key	8899AAB8CCDDEEFF0011223344556677FEDCBA98765432100123456789ABCDEF
/eeee_tb/DUT/key1	8899AAB8CCDDEEFF0011223344556677
/eeee_tb/DUT/key2	FEDCBA98765432100123456789ABCDEF
/eeee_tb/DUT/const	6EA276726C487A885D278D10DD849401
/eeee_tb/DUT/x	E638DCC9A09594475D369F2399D1F276
/eeee_tb/DUT/sx	0998CA37A7947AAB878F4A5AE81B748A
/eeee_tb/DUT/f	3D0940999DB75D6A9257071D5E6144A6
/eeee_tb/DUT/key1	C3D5FA01EBE36F7A9374427AD7CA89498899AAB8CCDDEEFF0011223344556677
/eeee_tb/DUT/key2	8899AAB8CCDDEEFF0011223344556677
/eeee_tb/DUT/key3	FEDCBA98765432100123456789ABCDEF
/eeee_tb/DUT/key4	DB31485315694343228D6AEF8CC78C44
/eeee_tb/DUT/key5	3D4553D8E9CFEC6815E8ADC40A9FFD04
/eeee_tb/DUT/key6	57646468C44A5E28D3E59246F429F1AC
/eeee_tb/DUT/key7	8D079435165C64328532E82834DA581B
/eeee_tb/DUT/key8	51E640757E8745DE705727265A009881
/eeee_tb/DUT/key9	5A7925017B9FDD3ED72A91A22286F984
/eeee_tb/DUT/key10	BB44E25378C73123A5F32F73CDB6E517

Рисунок 20 – Временные диаграммы сигналов для проверки алгоритма развертывания ключа

При сравнении результатов тестирования со значениями справочной информации ГОСТ Р 34.12 - 2015 был сделан вывод, что алгоритм развертывания ключа работает правильно.

4.3 Тестирование алгоритма зашифрования

На рисунке 21 представлены временные диаграммы для проверки алгоритма зашифрования. Сигналы представлены в виде шестнадцатеричной системы счисления.

Описание сигналов:

- data_{in}. Открытый текст длиной 128 бит;
- x. Результат X преобразования между сигналами data_{in} и key1;
- s_x. Результат s преобразования сигнала x;
- l_{sx1}. Результат L преобразования сигнала s_x;
- l_{sx2}. Результат LSX преобразования сигнала l_{sx1};
- l_{sx3}. Результат LSX преобразования сигнала l_{sx2};
- l_{sx4}. Результат LSX преобразования сигнала l_{sx3};
- l_{sx5}. Результат LSX преобразования сигнала l_{sx4};
- l_{sx6}. Результат LSX преобразования сигнала l_{sx5};

- lxx7. Результат LSX преобразования сигнала lxx6;
- lxx8. Результат LSX преобразования сигнала lxx7;
- lxx9. Результат LSX преобразования сигнала lxx8;
- x10. Результат X преобразования между сигналами lxx9 и key10.

Signal	Value	Msgs	Value
/eeee_tb/DUT/data_in	1122334455667700FFEEDDCCBBA9988		1122334455667700FFEEDDCCBBA9988
/eeee_tb/DUT/x	998B99FF998B99FFFFFFFFFFFFFFFF		998B99FF998B99FFFFFFFFFFFFFFFF
/eeee_tb/DUT/sx	E87DE8B6E87DE8B6B6B6B6B6B6B6B6B6		E87DE8B6E87DE8B6B6B6B6B6B6B6B6B6
/eeee_tb/DUT/lxx1	E297B686E355B0A1CF4A2F9249140830		E297B686E355B0A1CF4A2F9249140830
/eeee_tb/DUT/lxx2	285E497A0862D596B36F4258A1C69072		285E497A0862D596B36F4258A1C69072
/eeee_tb/DUT/lxx3	0187A3A429B567841AD50D29207CC34E		0187A3A429B567841AD50D29207CC34E
/eeee_tb/DUT/lxx4	EC9BDBA057D4F4D77C5D70619DCAD206		EC9BDBA057D4F4D77C5D70619DCAD206
/eeee_tb/DUT/lxx5	1357FD11DE9257290C2A1473EB6BCDE1		1357FD11DE9257290C2A1473EB6BCDE1
/eeee_tb/DUT/lxx6	28AE31E7D4C2354261027EF0832897DF		28AE31E7D4C2354261027EF0832897DF
/eeee_tb/DUT/lxx7	07E223D56002C013D3F5E6F714886D2D		07E223D56002C013D3F5E6F714886D2D
/eeee_tb/DUT/lxx8	CD8EF6CD97E0E092A8E4CCA61B38BF65		CD8EF6CD97E0E092A8E4CCA61B38BF65
/eeee_tb/DUT/lxx9	0D8E40E4A800D06B2F1B37EA379EAD8E		0D8E40E4A800D06B2F1B37EA379EAD8E
/eeee_tb/DUT/x10	7F679D90BEC24305A468D42B9D4EDCD		7F679D90BEC24305A468D42B9D4EDCD

Рисунок 21 – Временные диаграммы сигналов для проверки алгоритма зашифрования

При сравнении результатов тестирования со значениями справочной информации ГОСТ Р 34.12 - 2015 был сделан вывод, что алгоритм зашифрования работает правильно.

4.4 Тестирование алгоритма расшифрования

На рисунке 22 представлены временные диаграммы для проверки алгоритма зашифрования. Сигналы представлены в виде шестнадцатеричной системы счисления.

Описание сигналов:

- data_in. Шифртекст длиной 128 бит;
- x10. Результат X преобразования между сигналами data_in и key10;
- lx. Результат обратного L преобразования сигнала x;
- slx10. Результат обратного S преобразования сигнала lx;
- slx9. Результат обратного SLX преобразования сигнала slx10;
- slx8. Результат обратного SLX преобразования сигнала slx9;
- slx7. Результат обратного SLX преобразования сигнала slx8;

- slx6. Результат обратного SLX преобразования сигнала slx7;
- slx5. Результат обратного SLX преобразования сигнала slx6;
- slx4. Результат обратного SLX преобразования сигнала slx5;
- slx3. Результат обратного SLX преобразования сигнала slx4;
- slx2. Результат обратного SLX преобразования сигнала slx3;
- x. Результат X преобразования между сигналами lxx2 и key1.

		Msgs
/eeee_tb/DUT/data_in	112233445566778899AABBCCDEFF0A00	7F679D908EBC24305A468D4289D4EDCD
/eeee_tb/DUT/x10	63C8EE3043DA83D3ECF7016460B54A43	0D8E40E4A800D06B2F1B37EA379EAD8E
/eeee_tb/DUT/lx	34649359934406A7A7C5D982BCE7B747	8A6B930A52211B45C5BAA43FF8B91319
/eeee_tb/DUT/slx10	41AC14F014A738A0A00DCDA897CE5D5B	76CA149EEF27D1B10D17E3D5D68E5A72
/eeee_tb/DUT/slx9	82B178FA42B616FD6078AC5938886F9A	5D9B06D41B9D1D2D04DF7755363E94A9
/eeee_tb/DUT/slx8	2341A88E43D7DF8A191634AE2AD6FF7C	79487192AA45709C115559D6E9280F6E
/eeee_tb/DUT/slx7	4CE3760AACA496A2B54EB22BD1D93342	AE506924C8CE331BB918FC5BDFB195FA
/eeee_tb/DUT/slx6	BBFCF354F0B94F0CBC08FEB1426E561	BBFFBFC8939EAAFFAFB8E22769E323AA
/eeee_tb/DUT/slx5	A90FFF852EE62057EEFCB567F847827E	3CC2F07CC07A8BEC0F3EA0ED2AE33E4A
/eeee_tb/DUT/slx4	9C7BF5E29379D5CA4DAFC79D488EF5B1	F36F01291D0B96D591E228872D011C36
/eeee_tb/DUT/slx3	4E7FB8B81CBEE4FB2FE66F99DF4A8BA1	1C4B0C1E950182B1CE696AF5C0BFC5DF
/eeee_tb/DUT/slx2	2ED62C0E006F5EEA8642011B6216594F	998B99FF998B99FFFFFFFFFFFFFFFFFFFF
/eeee_tb/DUT/x	A64F86B5CCB280158653232826433F38	1122334455667700FFEEDDCC88AA9988

Рисунок 22 – Временные диаграммы сигналов для проверки алгоритма расшифрования

При сравнении результатов тестирования со значениями справочной информации ГОСТ Р 34.12 - 2015 был сделан вывод, что алгоритм расшифрования работает правильно.

4.5 Тестирование процесса программы

На рисунке 23 представлены временные диаграммы для проверки модели аппаратного модуля. Сигналы представлены в виде шестнадцатеричной системы счисления.

Описание сигналов:

- data_in. Входной сигнал для шифрования строки;
- key. Входной сигнал ключевой информации;
- mode. Входной сигнал режима шифрования. Если mode = 1, то входной сигнал data_in шифруется. Если mode = 0, то входной сигнал data_in расшифровывается;
- clk. Входной тактовый сигнал;

- key_download. Входной сигнал загрузки ключа;
- key_delete. Входной сигнал сброса ключа;
- data_out. Выходной сигнал шифрованного сигнала data_in.
- key1 - 10. Сгенерированные итерационные ключи;

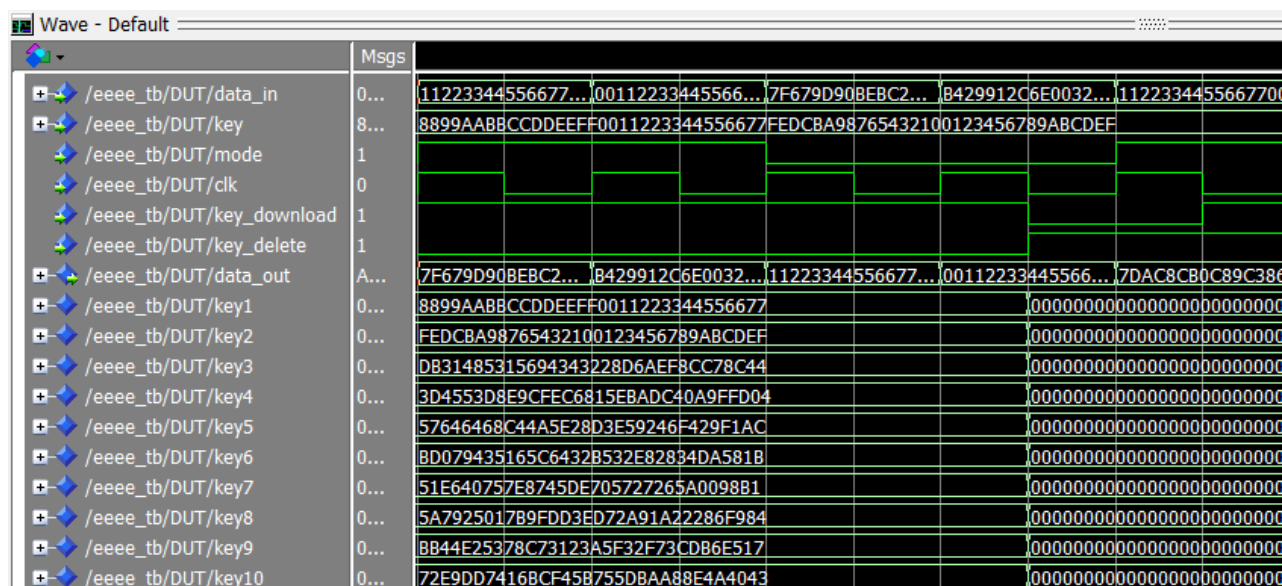


Рисунок 23 – Временные диаграммы сигналов для проверки модели аппаратного модуля

При сравнении результатов тестирования можно сделать вывод, что модель аппаратного модуля работает правильно.

5 Вредные психофизиологические факторы, влияющие на нервную систему специалиста по защите информации. Методы и средства физической культуры, снижающие их воздействие

Работники, связанные с периодической или непрерывной работой за ПК, подвергаются воздействию условий производственной опасности, несмотря на то, что безопасность самих пользователей электронно-вычисляемых машин и видеодисплейных терминалов регулярно улучшаются. Одним из главных критериев оказывающих значимое влияние на состояние самочувствия пользователя ЭВМ и ВДТ являются психофизиологические факторы:

- напряжённость зрения;
- напряжённость памяти;
- нерациональная организация рабочего места;
- напряжённость нервно-эмоциональные перегрузки;
- сравнительно большой объем данных, обрабатываемый в единицу времени;
- продолжительное статическое напряжение;
- однообразие работы в отдельных вариантах.

К основным вредным факторам, воздействующим на человека во время долгой работы за персональным компьютером, относятся: продолжительное сидячее состояние, электромагнитное излучение, нагрузка на зрение, перегруженность кистевых суставов, вероятность заболеваний органов дыхания, аллергии, нарушение нормального течения беременности и др.

Длительная работа за компьютером в сидячем положении оказывает статическую нагрузку, которая вызывает костно-мышечные заболевания, стрессы, легкую возбудимость и депрессии, нарушение сна. Так же есть вероятность возникновения воспалительных процессов в сухожилиях и мышечных тканях в результате частого нажатия на клавиши. Всему этому способствует длительное нахождение в положении сидя в одной и той же позе. В мышцах, подверженных долгой статической нагрузке, ухудшается кровоток,

накапливаются продукты распада (молочная кислота), что способствует появлению болевых ощущений.

При работе за компьютером человек часто испытывает напряжение глазных мышц, что является следствием задержки моргания в ожидании информации. Надобность фильтрации нужной информации за счет его чрезмерного потока в виде мерцания экрана монитора, по которому двигаются глаза, может привести к заболеванию, которое характеризуется быстрым уставанием глаз. При чрезмерном потоке информации, наступает синдром информационного утомления.

Не последнее по важности значение имеет организация рабочего места. Эргономика кресла должна повторять форму спины, так же необходимо присутствие регулировок наклона и высоты для уменьшения давления на копчик (при низкой посадке спинки) и бедра (при высокой посадке).

При длительном сидячем положении устают, и в следствии этого напрягаются шейные мышцы, руки и плечи, что приводит к остеохондрозу. Долгое нахождение в сидячей позе ведет к застою крови в тазовых органах, что может привести к простатиту и геморрою. Причиной остеохондроза является нарушение положения межпозвоночных дисков, которое приводит к грыже межпозвоночного диска. Симптомами остеохондроза могут быть разнообразные боли в спине и конечностях. А последствиями являются различные проблемы с организмом вплоть до паралича конечностей и смерти. Наиболее частой причиной остеохондроза является ослабление мышц спины.

Человеческий глаз очень чувствителен к частоте мерцания монитора компьютера. Долгое пребывание за монитором введет к тому, что мышцы управляющие глазом, находятся в постоянном напряжении, что приводит к ухудшению зрения. Важное значение для комфортной работы за монитором его правильная калибровка контраста и яркости, немаловажное значение имеет выбор шрифтов, шрифтов, компоновки окон в используемых приложениях, и положение монитора относительно зрителя. Изображение на мониторе складывается из набора пикселей, которые постоянно мерцают и меняют цвет,

что оказывает большое влияние на глаза. У пользователя неизбежно ухудшается зрение, глаза начинают слезиться, появляется головная боль, утомление, изображение двоится и искажается.

Во время работы нужен постоянный отдых для глаз. Нужно стараться часто моргать для того чтобы поверхность глаза оставалась влажной.

Не лишними будут 15-минутные перерывы, во время которых нужно выполнять упражнения для глаз такие как, повороты по часовой и против часовой стрелки, вверх и вниз. Затем просто закрыть глаза и расслабиться. Данные упражнения являются наиболее простыми и эффективными.

Так же в перерывах можно выполнить ряд других упражнений.

Жонглирование помогает снять напряжение и тренировать глаза, так же это небольшая физическая нагрузка на спину руки и ноги, в начале обучения жонглированию придется часто подбирать упавшие шары. Для жонглирования не необходима специальная спортивная форма и достаточно повседневной одежды.

К физическим упражнениям, тренирующим мелкую моторику и глаза можно отнести карточные фокусы.

Предпочтительно выполнять растяжку для улучшения кровообращения не только по всему телу, но и в головном мозге, что улучшает самочувствие и общее состояние организма. Так же физические упражнения способствуют снятию сонливости и слабости. Упражнения можно выполнять в течении достаточно короткого времени от 3 до 10 минут за каждый час работы за компьютером.

Физические упражнения помогают снять утомление нервной системы и всего тела, повышают работоспособность, способствуют укреплению здоровья.

Для напоминания отдыха существуют различные программы одной из них является «RestTime». Работает оно по следующему принципу, при истечении определенного времени выводится на экран монитора «стикер» с напоминанием об необходимости выполнения физических упражнений. Есть более

функциональные программы, например, «EyeDefender» помимо обычного напоминания предлагает специальную тренировочную программу для глаз».

Так же к физическим упражнениям можно отнести и более время затратное и в следствии более полезное - это плавание. Данный вид спорта гармонично развивает все группы мышц.

Недостаток движения можно компенсировать так же ходьбой в разных темпах что улучшит кровоснабжение всех органов человеческого.

Заключение

В результате работы были рассмотрены общие принципы шифрования, режимы блочных шифров и был рассмотрен ГОСТ Р 34.12-2015.

После этого был реализован алгоритм блочного шифрования с длиной блока $n=128$ бит, на базисных элементах И, ИЛИ, НЕ. Были проведены результаты тестирования. Выполнено сравнение с помощью набора тестовых последовательностей и установлено, что модель аппаратного модуля работает должным образом.

Далее были рассмотрены вредные психофизиологические факторы, влияющие на нервную систему специалиста по защите информации и методы, снижающие их воздействие.

Результатом выполнения ВКР является полностью протестированная модель аппаратного модуля, который реализует ГОСТ Р 34.12 – 2015 для возможного дальнейшего построения аппаратного модуля или для дальнейшего построения криптопроцессора.

Модель аппаратного модуля подлежит дальнейшему усовершенствованию, но уже на данном этапе разработки установлено, что полученный модель удовлетворяет всем техническим требованиям наложенные на него в техническом задании на данный дипломный проект.

Таким образом, задание на выпускную квалификационную работу выполнено в полном объеме.

Список использованных источников

1. Шифрование[электронный ресурс].- Режим доступа: <https://it-sektor.ru/shifrovanie.html>, свободный - Загл.с экрана (Дата обращения: 18.01.2018).
2. Криптографическая стойкость [электронный ресурс].- Режим доступа: http://www.gpedia.com/ru/gpedia/Криптографическая_стойкость, свободный - Загл.с экрана (Дата обращения: 18.01.2018).
3. ГОСТ Р 34.13 – 2015. Информационная технология. Криптографическая защита информации. Режимы работы блочных шифров. Введ. 01.01.2016. Стандартиформ – 42л.
4. Аппаратное шифрование[электронный ресурс].- Режим доступа: http://wiki-org.ru/wiki/Аппаратное_шифрование, свободный - Загл.с экрана (Дата обращения: 18.01.2018).
5. ГОСТ Р 34.12 - 2015. Информационная технология. Криптографическая защита информации. Блочные шифры. Введ. 01.01.2016. Стандартиформ – 25л.
6. VHDL vs СХЕМОТЕХНИКА [электронный ресурс].- Режим доступа: <https://logways.ru/2011/09/12/vhdl-vs-sxemotexnika/>, свободный - Загл.с экрана (Дата обращения: 18.01.2018).
7. ModelSim [электронный ресурс].- Режим доступа: http://www.compitech.ru/html.cgi/arhiv/02_06/stat_122.htm, свободный - Загл.с экрана (Дата обращения: 18.01.2018).
8. Опасные и вредные факторы при работе с ПЭВМ [электронный ресурс].- Режим доступа: <http://rgrtu-640.ru/bezopasnost-zhiznedeyatelnosti/45.html>, свободный - Загл.с экрана (Дата обращения: 18.01.2018).

Приложение А

(обязательное)

Листинг программы, реализующий алгоритм блочного шифрования с
длиной блока $n=128$ бит

```

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_signed;
use IEEE.std_logic_unsigned.all;
use IEEE.numeric_std.all;
entity eeee is
port (
    data_in : in std_logic_vector (127 downto 0);
    key : in std_logic_vector (255 downto 0);
    mode : in bit;
    clk : in bit;
    key_download : in bit;
    key_delete : in bit;
    data_out : out std_logic_vector (127 downto 0)
);
end eeee;
architecture kuznechik of eeee is
    constant c1 : std_logic_vector :=x"6EA276726C487AB85D27BD10DD849401";
    constant c2 : std_logic_vector :=x"DC87ECE4D890F4B3BA4EB92079CBEB02";
    constant c3 : std_logic_vector :=x"B2259A96B4D88E0BE7690430A44F7F03";
    constant c4 : std_logic_vector :=x"7BCD1B0B73E32BA5B79CB140F2551504";
    constant c5 : std_logic_vector :=x"156F6D791FAB511DEABB0C502FD18105";
    constant c6 : std_logic_vector :=x"A74AF7EFAB73DF160DD208608B9EFE06";
    constant c7 : std_logic_vector :=x"C9E8819DC73BA5AE50F5B570561A6A07";
    constant c8 : std_logic_vector :=x"F6593616E6055689ADFBA18027AA2A08";
    constant c9 : std_logic_vector :=x"98FB40648A4D2C31F0DC1C90FA2EBE09";
    constant c10 : std_logic_vector :=x"2ADEDADF23E95A23A17B518A05E61C10A";
    constant c11 : std_logic_vector :=x"447CAC8052DDD8824A92A5B083E5550B";
    constant c12 : std_logic_vector :=x"8D942D1D95E67D2C1A6710C0D5FF3F0C";
    constant c13 : std_logic_vector :=x"E3365B6FF9AE07944740ADD0087BAB0D";
    constant c14 : std_logic_vector :=x"5113C1F94D76899FA029A9E0AC34D40E";
    constant c15 : std_logic_vector :=x"3FB1B78B213EF327FD0E14F071B0400F";
    constant c16 : std_logic_vector :=x"2FB26C2C0F0AACD1993581C34E975410";
    constant c17 : std_logic_vector :=x"41101A5E6342D669C4123CD39313C011";
    constant c18 : std_logic_vector :=x"F33580C8D79A5862237B38E3375CBF12";
    constant c19 : std_logic_vector :=x"9D97F6BABBD222DA7E5C85F3EAD82B13";
    constant c20 : std_logic_vector :=x"547F77277CE987742EA93083BCC24114";
    constant c21 : std_logic_vector :=x"3ADD015510A1FDCC738E8D936146D515";
    constant c22 : std_logic_vector :=x"88F89BC3A47973C794E789A3C509AA16";
    constant c23 : std_logic_vector :=x"E65AEDB1C831097FC9C034B3188D3E17";
    constant c24 : std_logic_vector :=x"D9EB5A3AE90FFA5834CE2043693D7E18";
    constant c25 : std_logic_vector :=x"B7492C48854780E069E99D53B4B9EA19";
    constant c26 : std_logic_vector :=x"056CB6DE319F0EEB8E80996310F6951A";
    constant c27 : std_logic_vector :=x"6BCEC0AC5DD77453D3A72473CD72011B";
    constant c28 : std_logic_vector :=x"A22641319AECDD1FD835291039B686B1C";
    constant c29 : std_logic_vector :=x"CC843743F6A4AB45DE752C1346ECFF1D";
    constant c30 : std_logic_vector :=x"7EA1ADD5427C254E391C2823E2A3801E";
    constant c31 : std_logic_vector :=x"1003DBA72E345FF6643B95333F27141F";
    constant c32 : std_logic_vector :=x"5EA7D8581E149B61F16AC1459CEDA820";
    -----
    signal key1 : std_logic_vector (127 downto 0);
    signal key2 : std_logic_vector (127 downto 0);
    signal key3 : std_logic_vector (127 downto 0);
    signal key4 : std_logic_vector (127 downto 0);
    signal key5 : std_logic_vector (127 downto 0);
    signal key6 : std_logic_vector (127 downto 0);
    signal key7 : std_logic_vector (127 downto 0);
    signal key8 : std_logic_vector (127 downto 0);
    signal key9 : std_logic_vector (127 downto 0);
    signal key10 : std_logic_vector (127 downto 0);
    signal key34 : std_logic_vector (255 downto 0);
    signal key56 : std_logic_vector (255 downto 0);
    signal key78 : std_logic_vector (255 downto 0);

```

```

        signal key910 : std_logic_vector (255 downto 0);
-----
---function c=148 -
function func_c148(z: std_logic_vector (7 downto 0)) return std_logic_vector is
variable result_s : std_logic_vector(7 downto 0) := (others => '0');
variable uj, mj, vj, lj, jj, kj, sj, pj : std_logic := '0';
begin
pj:= z(0); sj:= z(1); kj:= z(2); jj:= z(3); lj:= z(4); vj:= z(5); m:= z(6);
uj:= z(7);
result_s (7) := (Uj and mj and vj and sj and pj ) or ( not uj and not mj and vj
and sj and pj ) or ( not uj and mj and not vj and sj and pj ) or ( uj and not
mj and not vj and sj and pj ) or ( not uj and mj and vj and not sj and pj ) or
( uj and not mj and vj and not sj and pj ) or ( uj and mj and not vj and not sj
and pj ) or ( not uj and not mj and not vj and not sj and pj ) or ( not uj and
mj and vj and sj and not pj ) or ( uj and not mj and vj and sj and not pj ) or
( uj and mj and not vj and sj and not pj ) or ( not uj and not mj and not vj
and sj and not pj ) or ( uj and mj and vj and not sj and not pj ) or ( not uj
and not mj and vj and not sj and not pj ) or ( not uj and mj and not vj and not
sj and not pj ) or ( uj and not mj and not vj and not sj and not P);
result_s (6) := (U and lj and sj ) or ( not uj and not lj and sj ) or ( not uj
and lj and not sj ) or ( uj and not lj and not S);
result_s (5) := (not uj and vj and jj and sj ) or ( uj and not vj and jj and sj
) or ( uj and vj and not jj and sj ) or ( not uj and not vj and not jj and sj )
or ( uj and vj and jj and not sj ) or ( not uj and not vj and jj and not sj )
or ( not uj and vj and not jj and not sj ) or ( uj and not vj and not jj and
not S);
result_s (4) := (not mj and lj and kj and pj ) or ( mj and not lj and kj and pj
) or ( mj and lj and not kj and pj ) or ( not mj and not lj and not kj and pj )
or ( mj and lj and kj and not pj ) or ( not mj and not lj and kj and not pj )
or ( not mj and lj and not kj and not pj ) or ( mj and not lj and not kj and
not P);
result_s (3) := (V and jj and sj ) or ( not vj and not jj and sj ) or ( not vj
and jj and not sj ) or ( vj and not jj and not S);
result_s (2) := (not uj and lj and kj and pj ) or ( uj and not lj and kj and pj
) or ( uj and lj and not kj and pj ) or ( not uj and not lj and not kj and pj )
or ( uj and lj and kj and not pj ) or ( not uj and not lj and kj and not pj )
or ( not uj and lj and not kj and not pj ) or ( uj and not lj and not kj and
not P);
result_s (1) := (M and jj and sj ) or ( not mj and not jj and sj ) or ( not mj
and jj and not sj ) or ( mj and not jj and not S);
result_s (0) := (not uj and mj and kj and sj ) or ( uj and not mj and kj and sj
) or ( uj and mj and not kj and sj ) or ( not uj and not mj and not kj and sj )
or ( uj and mj and kj and not sj ) or ( not uj and not mj and kj and not sj )
or ( not uj and mj and not kj and not sj ) or ( uj and not mj and not kj and
not S);
return result_s;
end func_c148;
-----
---function c=32 -
function func_c32(z: std_logic_vector (7 downto 0)) return std_logic_vector is
variable result_s : std_logic_vector(7 downto 0) := (others => '0');
variable uj, mj, vj, lj, jj, kj, sj, pj : std_logic := '0';
begin
pj:= z(0); sj:= z(1); kj:= z(2); jj:= z(3); lj:= z(4); vj:= z(5); m:= z(6);
uj:= z(7);
result_s (7) := (not mj and vj and jj and kj ) or ( mj and not vj and jj and kj
) or ( mj and vj and not jj and kj ) or ( not mj and not vj and not jj and kj )
or ( mj and vj and jj and not kj ) or ( not mj and not vj and jj and not kj )
or ( not mj and vj and not jj and not kj ) or ( mj and not vj and not jj and
not K);
result_s (6) := (U and mj and lj and jj and sj ) or ( not uj and not mj and lj
and jj and sj ) or ( not uj and mj and not lj and jj and sj ) or ( uj and not
mj and not lj and jj and sj ) or ( not uj and mj and lj and not jj and sj ) or

```



```

result_s (2) := (V and lj and pj ) or ( not vj and not lj and pj ) or ( not vj
and lj and not pj ) or ( vj and not lj and not P);
result_s (1) := (U and lj and jj ) or ( not uj and not lj and jj ) or ( not uj
and lj and not jj ) or ( uj and not lj and not J);
result_s (0) := (not mj and vj and lj and kj and sj and pj ) or ( mj and not vj
and lj and kj and sj and pj ) or ( mj and vj and not lj and kj and sj and pj )
or ( not mj and not vj and not lj and kj and sj and pj ) or ( mj and vj and lj
and not kj and sj and pj ) or ( not mj and not vj and lj and not kj and sj and
pj ) or ( not mj and vj and not lj and not kj and sj and pj ) or ( mj and not
vj and not lj and not kj and sj and pj ) or ( mj and vj and lj and kj and not
sj and pj ) or ( not mj and not vj and lj and kj and not sj and pj ) or ( not
mj and vj and not lj and kj and not sj and pj ) or ( mj and not vj and not lj
and kj and not sj and pj ) or ( not mj and vj and lj and not kj and not sj and
pj ) or ( mj and not vj and lj and not kj and not sj and pj ) or ( mj and vj
and not lj and not kj and not sj and pj ) or ( not mj and not vj and not lj and
not kj and not sj and pj ) or ( mj and vj and lj and kj and sj and not pj ) or
( not mj and not vj and lj and kj and sj and not pj ) or ( not mj and vj and
not lj and kj and sj and not pj ) or ( mj and not vj and not lj and kj and sj
and not pj ) or ( not mj and vj and lj and not kj and sj and not pj ) or ( mj
and not vj and lj and not kj and sj and not pj ) or ( mj and vj and not lj and
not kj and sj and not pj ) or ( not mj and not vj and not lj and not kj and sj
and not pj ) or ( not mj and vj and lj and kj and not sj and not pj ) or ( mj
and not vj and lj and kj and not sj and not pj ) or ( mj and vj and not lj and
kj and not sj and not pj ) or ( not mj and not vj and not lj and kj and not sj
and not pj ) or ( mj and vj and lj and not kj and not sj and not pj ) or ( not
mj and not vj and lj and not kj and not sj and not pj ) or ( not mj and vj and
not lj and not kj and not sj and not pj ) or ( mj and not vj and not lj and not
kj and not sj and not P);
    return result_s;
end func_c133;
-----
---function c=16 -
function func_c16(z: std_logic_vector (7 downto 0)) return std_logic_vector is
variable result_s : std_logic_vector(7 downto 0) := (others => '0');
variable uj, mj, vj, lj, jj, kj, sj, pj : std_logic := '0';
begin
pj:= z(0); sj:= z(1); kj:= z(2); jj:= z(3); lj:= z(4); vj:= z(5); m:= z(6);
uj:= z(7);
result_s (7) := (not uj and mj and lj and jj ) or ( uj and not mj and lj and jj
) or ( uj and mj and not lj and jj ) or ( not uj and not mj and not lj and jj )
or ( uj and mj and lj and not jj ) or ( not uj and not mj and lj and not jj )
or ( not uj and mj and not lj and not jj ) or ( uj and not mj and not lj and
not J);
result_s (6) := (not uj and vj and lj and kj ) or ( uj and not vj and lj and kj
) or ( uj and vj and not lj and kj ) or ( not uj and not vj and not lj and kj )
or ( uj and vj and lj and not kj ) or ( not uj and not vj and lj and not kj )
or ( not uj and vj and not lj and not kj ) or ( uj and not vj and not lj and
not K);
result_s (5) := G;
result_s (4) := (not uj and pj ) or ( uj and not P);
result_s (3) := B;
result_s (2) := (not uj and vj ) or ( uj and not V);
result_s (1) := (U and mj and lj ) or ( not uj and not mj and lj ) or ( not uj
and mj and not lj ) or ( uj and not mj and not L);
result_s (0) := (U and vj and lj ) or ( not uj and not vj and lj ) or ( not uj
and vj and not lj ) or ( uj and not vj and not L);
    return result_s;
end func_c16;
-----
---function c=194 -
function func_c194(z: std_logic_vector (7 downto 0)) return std_logic_vector is
variable result_s : std_logic_vector(7 downto 0) := (others => '0');
variable uj, mj, vj, lj, jj, kj, sj, pj : std_logic := '0';

```

```

begin
pj:= z(0); sj:= z(1); kj:= z(2); jj:= z(3); lj:= z(4); vj:= z(5); m:j= z(6);
uj:= z(7);
result_s (7) := (not vj and jj and kj and pj ) or ( vj and not jj and kj and pj )
) or ( vj and jj and not kj and pj ) or ( not vj and not jj and not kj and pj )
or ( vj and jj and kj and not pj ) or ( not vj and not jj and kj and not pj )
or ( not vj and jj and not kj and not pj ) or ( vj and not jj and not kj and
not P);
result_s (6) := (not uj and vj and lj and jj and sj and pj ) or ( uj and not vj
and lj and jj and sj and pj ) or ( uj and vj and not lj and jj and sj and pj )
or ( not uj and not vj and not lj and jj and sj and pj ) or ( uj and vj and lj
and not jj and sj and pj ) or ( not uj and not vj and lj and not jj and sj and
pj ) or ( not uj and vj and not lj and not jj and sj and pj ) or ( uj and not
vj and not lj and not jj and sj and pj ) or ( uj and vj and lj and jj and not
sj and pj ) or ( not uj and not vj and lj and jj and not sj and pj ) or ( not
uj and vj and not lj and jj and not sj and pj ) or ( uj and not vj and not lj
and jj and not sj and pj ) or ( not uj and vj and lj and not jj and not sj and
pj ) or ( uj and not vj and lj and not jj and not sj and pj ) or ( uj and vj
and not lj and not jj and not sj and pj ) or ( not uj and not vj and not lj and
not jj and not sj and pj ) or ( uj and vj and lj and jj and sj and not pj ) or
( not uj and not vj and lj and jj and sj and not pj ) or ( not uj and vj and
not lj and jj and sj and not pj ) or ( uj and not vj and not lj and jj and sj
and not pj ) or ( not uj and vj and lj and not jj and sj and not pj ) or ( uj
and not vj and lj and not jj and sj and not pj ) or ( not uj and vj and not lj
and not jj and sj and not pj ) or ( not uj and vj and not lj and not jj and sj
and not pj ) or ( not uj and vj and lj and jj and not sj and not pj ) or ( uj
and not vj and lj and jj and not sj and not pj ) or ( uj and vj and not lj and
jj and not sj and not pj ) or ( not uj and not vj and not lj and jj and not sj
and not pj ) or ( uj and vj and lj and not jj and not sj and not pj ) or ( not
uj and not vj and lj and not jj and not sj and not pj ) or ( not uj and vj and
not lj and not jj and not sj and not pj ) or ( uj and not vj and not lj and not
jj and not sj and not P);
result_s (5) := (not uj and mj and vj and lj ) or ( uj and not mj and vj and lj
) or ( uj and mj and not vj and lj ) or ( not uj and not mj and not vj and lj )
or ( uj and mj and vj and not lj ) or ( not uj and not mj and vj and not lj )
or ( not uj and mj and not vj and not lj ) or ( uj and not mj and not vj and
not L);
result_s (4) := (not mj and vj and lj and jj ) or ( mj and not vj and lj and jj
) or ( mj and vj and not lj and jj ) or ( not mj and not vj and not lj and jj )
or ( mj and vj and lj and not jj ) or ( not mj and not vj and lj and not jj )
or ( not mj and vj and not lj and not jj ) or ( mj and not vj and not lj and
not J);
result_s (3) := (U and vj and lj and jj and kj ) or ( not uj and not vj and lj
and jj and kj ) or ( not uj and vj and not lj and jj and kj ) or ( uj and not
vj and not lj and jj and kj ) or ( not uj and vj and lj and not jj and kj ) or
( uj and not vj and lj and not jj and kj ) or ( uj and vj and not lj and not jj
and kj ) or ( not uj and not vj and not lj and not jj and kj ) or ( not uj and
vj and lj and jj and not kj ) or ( uj and not vj and lj and jj and not kj ) or
( uj and vj and not lj and jj and not kj ) or ( not uj and not vj and not lj
and jj and not kj ) or ( uj and vj and lj and not jj and not kj ) or ( not uj
and not vj and lj and not jj and not kj ) or ( not uj and vj and not lj and not
jj and not kj ) or ( uj and not vj and not lj and not jj and not K);
result_s (2) := (not uj and mj and lj and jj and kj and sj ) or ( uj and not mj
and lj and jj and kj and sj ) or ( uj and mj and not lj and jj and kj and sj )
or ( not uj and not mj and not lj and jj and kj and sj ) or ( uj and mj and lj
and not jj and kj and sj ) or ( not uj and not mj and lj and not jj and kj and
sj ) or ( not uj and mj and not lj and not jj and kj and sj ) or ( uj and not
mj and not lj and not jj and kj and sj ) or ( uj and mj and lj and jj and not
kj and sj ) or ( not uj and not mj and lj and jj and not kj and sj ) or ( not
uj and mj and not lj and not kj and sj ) or ( uj and not mj and not lj
and jj and not kj and sj ) or ( not uj and mj and lj and not jj and not kj and
sj ) or ( uj and not mj and lj and not jj and not kj and sj ) or ( uj and mj
and not lj and not jj and not kj and sj ) or ( not uj and not mj and not lj and

```



```

pj ) or ( uj and mj and vj and not jj and not kj and not sj and not pj ) or (
not uj and not mj and vj and not jj and not kj and not sj and not pj ) or ( not
uj and mj and not vj and not jj and not kj and not sj and not pj ) or ( uj and
not mj and not vj and not jj and not kj and not sj and not P);
result_s (0) := (not mj and lj and jj and sj ) or ( mj and not lj and jj and sj
) or ( mj and lj and not jj and sj ) or ( not mj and not lj and not jj and sj )
or ( mj and lj and jj and not sj ) or ( not mj and not lj and jj and not sj )
or ( not mj and lj and not jj and not sj ) or ( mj and not lj and not jj and
not S);
    return result_s;
end func_c194;
-----
--function c=192 -
function func_c192(z: std_logic_vector (7 downto 0)) return std_logic_vector is
    variable result_s : std_logic_vector(7 downto 0) := (others => '0');
    variable uj, mj, vj, lj, jj, kj, sj, pj : std_logic := '0';
begin
pj:= z(0); sj:= z(1); kj:= z(2); jj:= z(3); lj:= z(4); vj:= z(5); m:= z(6);
uj:= z(7);
result_s (7) := (not uj and mj and vj and jj and kj and pj ) or ( uj and not mj
and vj and jj and kj and pj ) or ( uj and mj and not vj and jj and kj and pj )
or ( not uj and not mj and not vj and jj and kj and pj ) or ( uj and mj and vj
and not jj and kj and pj ) or ( not uj and not mj and vj and not jj and kj and
pj ) or ( not uj and mj and not vj and not jj and kj and pj ) or ( uj and not
mj and not vj and not jj and kj and pj ) or ( uj and mj and vj and not jj and
not kj and pj ) or ( not uj and not mj and vj and jj and not kj and pj ) or ( not
uj and mj and not vj and jj and not kj and pj ) or ( uj and not mj and not vj
and jj and not kj and pj ) or ( not uj and mj and vj and not jj and not kj and
pj ) or ( uj and not mj and vj and not jj and not kj and pj ) or ( uj and mj
and not vj and not jj and not kj and pj ) or ( not uj and not mj and not vj and
not jj and not kj and pj ) or ( uj and mj and vj and jj and kj and not pj ) or
( not uj and not mj and vj and jj and kj and not pj ) or ( not uj and mj and
not vj and jj and kj and not pj ) or ( uj and not mj and not vj and jj and kj
and not pj ) or ( not uj and mj and vj and not jj and kj and not pj ) or ( uj
and not mj and vj and not jj and kj and not pj ) or ( uj and mj and not vj and
not jj and kj and not pj ) or ( not uj and not mj and not vj and not jj and kj
and not pj ) or ( not uj and mj and vj and jj and not kj and not pj ) or ( uj
and not mj and vj and not jj and kj and not pj ) or ( uj and mj and not vj and
not jj and kj and not pj ) or ( not uj and not mj and not vj and jj and not kj
and not pj ) or ( uj and mj and vj and not jj and not kj and not pj ) or ( not
uj and not mj and vj and not jj and not kj and not pj ) or ( not uj and mj and
not vj and not jj and not kj and not pj ) or ( uj and not mj and not vj and not
jj and not kj and not P);
result_s (6) := (not lj and jj and sj and pj ) or ( lj and not jj and sj and pj
) or ( lj and jj and not sj and pj ) or ( not lj and not jj and not sj and pj )
or ( lj and jj and sj and not pj ) or ( not lj and not jj and sj and not pj )
or ( not lj and jj and not sj and not pj ) or ( lj and not jj and not sj and
not P);
result_s (5) := (U and mj and vj ) or ( not uj and not mj and vj ) or ( not uj
and mj and not vj ) or ( uj and not mj and not V);
result_s (4) := (M and vj and lj ) or ( not mj and not vj and lj ) or ( not mj
and vj and not lj ) or ( mj and not vj and not L);
result_s (3) := (not uj and vj and lj and jj ) or ( uj and not vj and lj and jj
) or ( uj and vj and not lj and jj ) or ( not uj and not vj and not lj and jj )
or ( uj and vj and lj and not jj ) or ( not uj and not vj and lj and not jj )
or ( not uj and vj and not lj and not jj ) or ( uj and not vj and not lj and
not J);
result_s (2) := (U and mj and lj and jj and kj ) or ( not uj and not mj and lj
and jj and kj ) or ( not uj and mj and not lj and jj and kj ) or ( uj and not
mj and not lj and jj and kj ) or ( not uj and mj and lj and not jj and kj ) or
( uj and not mj and lj and not jj and kj ) or ( uj and mj and not lj and not jj
and kj ) or ( not uj and not mj and not lj and not jj and kj ) or ( not uj and
mj and lj and jj and not kj ) or ( uj and not mj and lj and jj and not kj ) or

```

```

( uj and mj and not lj and jj and not kj ) or ( not uj and not mj and not lj
and jj and not kj ) or ( uj and mj and lj and not jj and not kj ) or ( not uj
and not mj and lj and not jj and not kj ) or ( not uj and mj and not lj and not
jj and not kj ) or ( uj and not mj and not lj and not jj and not K);
result_s (1) := (M and vj and jj and kj and sj ) or ( not mj and not vj and jj
and kj and sj ) or ( not mj and vj and not jj and kj and sj ) or ( mj and not
vj and not jj and kj and sj ) or ( not mj and vj and jj and not kj and sj ) or
( mj and not vj and jj and not kj and sj ) or ( mj and vj and not jj and not kj
and sj ) or ( not mj and not vj and not jj and not kj and sj ) or ( not mj and
vj and jj and kj and not sj ) or ( mj and not vj and jj and kj and not sj ) or
( mj and vj and not jj and kj and not sj ) or ( not mj and not vj and not jj
and kj and not sj ) or ( mj and vj and jj and not kj and not sj ) or ( not mj
and not vj and jj and not kj and not sj ) or ( not mj and vj and not jj and not
kj and not sj ) or ( mj and not vj and not jj and not kj and not S);
result_s (0) := (U and mj and lj and jj and sj ) or ( not uj and not mj and lj
and jj and sj ) or ( not uj and mj and not lj and jj and sj ) or ( uj and not
mj and not lj and jj and sj ) or ( not uj and mj and lj and not jj and sj ) or
( uj and not mj and lj and not jj and sj ) or ( uj and mj and not lj and not jj
and sj ) or ( not uj and not mj and not lj and not jj and sj ) or ( not uj and
mj and lj and jj and not sj ) or ( uj and not mj and lj and jj and not sj ) or
( uj and mj and not lj and jj and not sj ) or ( not uj and not mj and not lj
and jj and not sj ) or ( uj and mj and lj and not jj and not sj ) or ( not uj
and not mj and lj and not jj and not sj ) or ( not uj and mj and not lj and not
jj and not sj ) or ( uj and not mj and not lj and not jj and not S);
    return result_s;
end func_c192;
-----
---function c=251 -
function func_c251(z: std_logic_vector (7 downto 0)) return std_logic_vector is
    variable result_s : std_logic_vector(7 downto 0) := (others => '0');
    variable uj, mj, vj, lj, jj, kj, sj, pj : std_logic := '0';
begin
    pj:= z(0); sj:= z(1); kj:= z(2); jj:= z(3); lj:= z(4); vj:= z(5); m:= z(6);
    uj:= z(7);
    result_s (7) := (not uj and vj and jj and pj ) or ( uj and not vj and jj and pj
) or ( uj and vj and not jj and pj ) or ( not uj and not vj and not jj and pj )
or ( uj and vj and jj and not pj ) or ( not uj and not vj and jj and not pj )
or ( not uj and vj and not jj and not pj ) or ( uj and not vj and not jj and
not P);
    result_s (6) := (U and mj and vj and lj and jj and kj and pj ) or ( not uj and
not mj and vj and lj and jj and kj and pj ) or ( not uj and mj and not vj and
lj and jj and kj and pj ) or ( uj and not mj and not vj and lj and jj and kj
and pj ) or ( not uj and mj and vj and not lj and jj and kj and pj ) or ( uj
and not mj and vj and not lj and jj and kj and pj ) or ( uj and mj and not vj
and not lj and jj and kj and pj ) or ( not uj and not mj and not vj and not lj
and jj and kj and pj ) or ( not uj and mj and vj and lj and not jj and kj and
pj ) or ( uj and not mj and vj and lj and not jj and kj and pj ) or ( uj and mj
and not vj and lj and not jj and kj and pj ) or ( not uj and not mj and not vj
and lj and not jj and kj and pj ) or ( uj and mj and vj and not lj and not jj
and kj and pj ) or ( not uj and not mj and vj and not lj and not jj and kj and
pj ) or ( not uj and mj and not vj and not lj and not jj and kj and pj ) or (
uj and not mj and not vj and not lj and not jj and kj and pj ) or ( not uj and
mj and vj and lj and jj and not kj and pj ) or ( uj and not mj and vj and lj
and jj and not kj and pj ) or ( not uj and not mj and not vj and lj and jj and
not kj and pj ) or ( uj and mj and vj and not lj and jj and not kj and pj ) or
( not uj and not mj and not vj and not lj and jj and not kj and pj ) or ( not
uj and not mj and not vj and not lj and not jj and not kj and pj ) or ( not
uj and mj and not vj and lj and not jj and not kj and pj ) or ( uj and not
mj and not vj and lj and not jj and not kj and pj ) or ( not uj and mj and vj
and not lj and not jj and not kj and pj ) or ( uj and not mj and vj and not lj

```



```

( uj and not mj and vj and not jj and sj ) or ( uj and mj and not vj and not jj
and sj ) or ( not uj and not mj and not vj and not jj and sj ) or ( not uj and
mj and vj and jj and not sj ) or ( uj and not mj and vj and jj and not sj ) or
( uj and mj and not vj and jj and not sj ) or ( not uj and not mj and not vj
and jj and not sj ) or ( uj and mj and vj and not jj and not sj ) or ( not uj
and not mj and vj and not jj and not sj ) or ( not uj and mj and not vj and not
jj and not sj ) or ( uj and not mj and not vj and not jj and not sj );
result_s (1) := (not uj and mj and vj and lj and kj and pj ) or ( uj and not mj
and vj and lj and kj and pj ) or ( uj and mj and not vj and lj and kj and pj )
or ( not uj and not mj and not vj and lj and kj and pj ) or ( uj and mj and vj
and not lj and kj and pj ) or ( not uj and not mj and vj and not lj and kj and
pj ) or ( not uj and mj and not vj and not lj and kj and pj ) or ( uj and not
mj and not vj and not lj and kj and pj ) or ( uj and mj and vj and lj and not
kj and pj ) or ( not uj and not mj and vj and lj and not kj and pj ) or ( not
uj and mj and not vj and lj and not kj and pj ) or ( uj and not mj and not vj
and lj and not kj and pj ) or ( not uj and mj and vj and not lj and not kj and
pj ) or ( uj and not mj and vj and not lj and not kj and pj ) or ( uj and mj
and not vj and not lj and not kj and pj ) or ( not uj and not mj and not vj and
not lj and not kj and pj ) or ( uj and mj and vj and lj and kj and not pj ) or
( not uj and not mj and vj and lj and kj and not pj ) or ( not uj and mj and
not vj and lj and kj and not pj ) or ( uj and not mj and not vj and lj and kj
and not pj ) or ( not uj and mj and vj and not lj and kj and not pj ) or ( uj
and not mj and vj and not lj and kj and not pj ) or ( not uj and not mj and vj
and not lj and kj and not pj ) or ( not uj and mj and not vj and not lj and kj
and not pj ) or ( not uj and mj and vj and lj and not kj and not pj ) or ( uj
and not mj and vj and lj and not kj and not pj ) or ( uj and mj and not vj and
lj and not kj and not pj ) or ( not uj and not mj and not vj and lj and not kj
and not pj ) or ( uj and mj and vj and not lj and not kj and not pj ) or ( not
uj and not mj and vj and not lj and not kj and not pj ) or ( not uj and mj and
not vj and not lj and not kj and not pj ) or ( uj and not mj and not vj and not
lj and not kj and not pj );
result_s (0) := (not mj and lj and sj and pj ) or ( mj and not lj and sj and pj
) or ( mj and lj and not sj and pj ) or ( not mj and not lj and not sj and pj )
or ( mj and lj and sj and not pj ) or ( not mj and not lj and sj and not pj )
or ( not mj and lj and not sj and not pj ) or ( mj and not lj and not sj and
not pj );
return result_s;
end func_c251;
-----

---function x
function func_x(x,y : std_logic_vector (127 downto 0)) return std_logic_vector
is
variable result : std_logic_vector(127 downto 0) := (others => '0');
begin
result := ((not x) and y) or ( x and (not y));
return result;
end func_x;
-----

---function S--
function func_s(z: std_logic_vector (127 downto 0)) return std_logic_vector is
variable result_s : std_logic_vector(127 downto 0) := (others => '0');
variable res : std_logic_vector(7 downto 0) := (others => '0');
variable uj, mj, vj, lj, jj, kj, sj, pj, y0, y1, y2, y3, y4, y5, y6, y7 :
std_logic := '0';
variable x : integer := 0;
variable r : integer := 0;
begin
for qqg in 0 to 15 loop
uj:= z(x); x:= x +1;
mj:= z(x); x:= x +1;
vj:= z(x); x:= x +1;
lg:= z(x); x:= x +1;
jj:= z(x); x:= x +1;

```

```

kj:= z(x); x:= x +1;
sj:= z(x); x:= x +1;
pj:= z(x); x:= x +1;
y0 := (not uj and not vj and lj and jj and sj ) or ( uj and not vj and not lj
and kj and sj ) or ( uj and mj and vj and not lj and not jj and kj ) or ( uj
and mj and not vj and lj and not jj and not kj ) or ( uj and not mj and vj and
lj and kj and sj ) or ( uj and not mj and not lj and not jj and not kj and pj )
or ( not uj and not lj and jj and kj and not sj and not pj ) or ( not uj and mj
and vj and lj and not jj and not kj and sj ) or ( uj and mj and not vj and lj
and not jj and not sj and pj ) or ( not mj and vj and lj and jj and not kj and
not sj and pj ) or ( not mj and vj and lj and kj and not pj ) or ( vj and lj
and jj and kj and not pj ) or ( mj and not vj and lj and jj and kj and sj ) or
( uj and mj and not vj and lj and not kj and not sj ) or ( uj and not mj and
not vj and not lj and not kj and not sj ) or ( uj and not mj and vj and lj and
jj and pj ) or ( not uj and mj and lj and jj and kj and pj ) or ( not uj and
not mj and not lj and jj and kj and pj ) or ( not uj and mj and vj and not lj
and sj and pj ) or ( not uj and mj and not lj and not jj and sj and pj ) or (
not mj and vj and jj and kj and sj and pj ) or ( uj and mj and not lj and jj
and not sj and pj ) or ( not uj and mj and vj and not jj and not sj and pj ) or
( uj and vj and jj and kj and not sj and pj ) or ( mj and vj and not lj and not
kj and not sj and pj ) or ( uj and not mj and not jj and not kj and not sj and
pj ) or ( not uj and not mj and not vj and not jj and kj and not pj ) or ( uj
and mj and vj and jj and not kj and not pj ) or ( not mj and vj and lj and jj
and sj and not pj ) or ( not uj and not mj and not lj and not jj and sj and not
pj ) or ( uj and mj and not lj and not kj and sj and not pj ) or ( not uj and
vj and lj and kj and not sj and not pj ) or ( not uj and vj and not lj and not
kj and not sj and not pj ) or ( mj and lj and jj and not kj and not sj and not
pj ) or ( not mj and vj and not jj and not kj and not sj and not pj ) or ( mj
and not vj and not jj and not kj and not sj and not pj ) or ( not uj and not mj
and vj and not lj and not jj and kj and sj ) or ( not uj and not mj and not vj
and lj and not jj and not kj and not sj ) or ( not uj and not mj and not vj and
not jj and not kj and sj and pj ) or ( not mj and not vj and lj and jj and kj
and not sj and pj ) or ( not uj and not mj and not vj and jj and not kj and not
sj and not pj ) or ( uj and vj and lj and not jj and not kj and not sj and not
P);
y1 := (U and mj and vj and lj and jj and sj ) or ( not mj and vj and lj and not
jj and not kj and sj ) or ( uj and mj and vj and lj and jj and pj ) or ( mj and
vj and not lj and not jj and kj and pj ) or ( not uj and not lj and jj and not
kj and sj and pj ) or ( not uj and not mj and not vj and not jj and not sj and
pj ) or ( uj and not mj and not lj and not jj and kj and not pj ) or ( not mj
and vj and not lj and kj and sj and not pj ) or ( mj and lj and jj and not kj
and sj and not pj ) or ( not uj and vj and lj and kj and not sj and not pj ) or
( vj and lj and not jj and kj and not sj and not pj ) or ( uj and not mj and
not lj and jj and kj and not sj and pj ) or ( uj and mj and vj and not lj and
jj and not sj and not pj ) or ( not uj and mj and vj and lj and not jj and not
sj and not pj ) or ( uj and not vj and lj and jj and kj and not sj and not pj )
or ( not mj and not jj and kj and not sj and not pj ) or ( not uj and mj and
not vj and lj and not jj and sj ) or ( not uj and not mj and vj and jj and kj
and sj ) or ( not mj and not vj and not lj and jj and not kj and sj ) or ( not
mj and vj and lj and not jj and kj and not sj ) or ( not uj and not mj and lj
and jj and not kj and not sj ) or ( uj and not mj and not vj and jj and kj and
pj ) or ( uj and mj and vj and lj and not kj and pj ) or ( not mj and not vj
and lj and not jj and not kj and pj ) or ( not uj and mj and lj and kj and sj
and pj ) or ( not uj and not vj and not jj and kj and sj and pj ) or ( mj and
not vj and not jj and kj and sj and pj ) or ( not uj and not vj and lj and not
kj and sj and pj ) or ( not mj and vj and not lj and not kj and sj and pj ) or
( not uj and mj and not vj and jj and not sj and pj ) or ( uj and mj and jj and
not kj and not sj and pj ) or ( not uj and vj and not jj and kj and sj and not
pj ) or ( uj and mj and not vj and not kj and sj and not pj ) or ( mj and not
vj and not lj and not jj and not sj and not pj ) or ( not uj and not mj and not
lj and kj and not sj and not pj ) or ( uj and not mj and not lj and not kj and
not sj and not pj ) or ( not uj and not vj and jj and not kj and not sj and not
pj ) or ( uj and not mj and vj and not lj and not jj and not kj and not sj ) or

```


not jj and sj and pj) or (uj and not mj and not vj and jj and not sj and pj)
 or (uj and not mj and jj and not kj and not sj and pj) or (not uj and mj and
 not vj and not lj and not sj and not pj) or (not mj and vj and lj and kj and
 not sj and not pj) or (not vj and lj and not jj and not kj and not sj and not
 pj) or (mj and not vj and not lj and not jj and kj and sj and pj) or (mj
 and not vj and lj and not jj and kj and sj and not pj) or (not uj and not mj
 and not lj and not jj and kj and sj and not pj) or (not uj and not vj and not
 jj and not kj and not sj) or (uj and not vj and lj and not jj and kj and sj)
 or (uj and not mj and not lj and not jj and not kj and sj) or (not uj and mj
 and vj and lj and not jj and not sj) or (uj and not mj and not vj and not lj
 and kj and not sj) or (uj and mj and not vj and not jj and kj and pj) or (not
 uj and not mj and lj and not jj and kj and pj) or (uj and not mj and not
 vj and not lj and not kj and pj) or (mj and vj and not lj and not jj and not
 kj and pj) or (not uj and mj and not vj and jj and not sj and pj) or (not
 mj and not vj and not lj and not jj and not sj and pj) or (not uj and not mj
 and vj and not lj and not jj and not pj) or (uj and mj and vj and lj and kj
 and not pj) or (uj and not mj and not lj and jj and kj and not pj) or (uj
 and vj and not lj and not jj and not kj and not pj) or (uj and not mj and vj
 and not lj and sj and not pj) or (uj and not lj and jj and kj and sj and not
 pj) or (uj and mj and vj and not kj and sj and not pj) or (not mj and not
 vj and lj and not kj and sj and not pj) or (mj and not vj and not lj and not
 kj and sj and not pj) or (not uj and not mj and jj and not kj and sj and not
 pj) or (uj and lj and jj and not kj and sj and not pj) or (not uj and not
 vj and not lj and jj and not sj and not pj) or (uj and not mj and lj and not
 jj and not sj and not pj) or (not uj and mj and jj and not kj and not sj and
 not pj) or (not uj and not mj and not vj and lj and jj and not kj and pj) or
 (uj and not mj and vj and lj and jj and sj and pj) or (uj and not mj and vj
 and lj and kj and sj and pj) or (uj and mj and lj and jj and kj and sj and pj
) or (not uj and not mj and vj and not jj and kj and sj and pj) or (not uj
 and mj and lj and jj and kj and not sj and pj) or (uj and vj and lj and jj
 and not kj and not sj and P);

y6 := (U and not mj and not lj and not jj and kj and sj) or (not uj and mj
 and not vj and lj and not kj and pj) or (mj and not vj and not jj and not kj
 and sj and pj) or (uj and mj and jj and kj and not sj and pj) or (not uj
 and mj and lj and not kj and not sj and pj) or (uj and not mj and vj and not
 jj and sj and not pj) or (uj and vj and not jj and not kj and sj and not pj)
 or (uj and not mj and jj and kj and not sj and not pj) or (mj and vj and jj
 and not kj and not sj and not pj) or (not uj and mj and vj and lj and kj and
 sj and pj) or (uj and not mj and vj and lj and not jj and not sj and pj) or
 (uj and mj and not lj and not jj and not kj and not sj and pj) or (uj and mj
 and vj and jj and kj and sj and not pj) or (not mj and not vj and not lj and
 not kj and not sj) or (not uj and not mj and not vj and not lj and not jj and not
 kj) or (uj and not vj and not lj and not jj and not kj and sj) or (not mj and
 not vj and lj and not jj and not kj and sj) or (not mj and vj and not lj and
 not jj and not kj and sj) or (not uj and not vj and not lj and not jj and kj
 and not sj) or (uj and not mj and not vj and jj and not kj and not sj) or (not
 uj and not mj and not vj and jj and kj and pj) or (not uj and not mj and
 not lj and jj and sj and pj) or (uj and vj and not lj and kj and sj and pj)
 or (uj and mj and not jj and kj and sj and pj) or (uj and vj and lj and kj
 and not sj and pj) or (not mj and vj and lj and kj and not sj and pj) or (not
 uj and not lj and not jj and kj and not sj and pj) or (mj and lj and jj
 and not kj and not sj and pj) or (not uj and not mj and vj and not lj and not
 jj and not pj) or (not uj and mj and vj and jj and not kj and not pj) or (not
 uj and not mj and lj and jj and sj and not pj) or (mj and not vj and not
 lj and jj and sj and not pj) or (not uj and mj and not vj and kj and sj and
 not pj) or (not uj and not mj and not vj and not kj and sj and not pj) or (not
 uj and mj and not vj and not lj and not sj and not pj) or (uj and mj and
 lj and not jj and not sj and not pj) or (not uj and vj and lj and kj and not
 sj and not pj) or (uj and not vj and lj and not sj and not pj) or (not uj and
 mj and vj and not lj and not jj and kj and pj) or (uj and not vj and lj and jj
 and kj and sj and pj) or (not uj and not vj and not lj and jj and kj and sj
 and pj) or (uj and mj and vj and lj and not kj and sj and pj) or (uj and

```

not mj and lj and jj and not kj and sj and pj ) or ( mj and vj and not lj and
jj and not kj and sj and pj ) or ( mj and not vj and lj and not jj and kj and
sj and not P);
y7 := (U and not mj and lj and jj and not kj and not sj ) or ( uj and mj and vj
and lj and not jj and pj ) or ( uj and not mj and not vj and not jj and kj and
pj ) or ( uj and mj and vj and lj and sj and pj ) or ( uj and mj and vj and not
kj and sj and pj ) or ( not mj and vj and lj and jj and sj and not pj ) or (
not uj and not mj and vj and lj and not jj and not kj and pj ) or ( not uj and
mj and vj and not lj and not kj and not sj and pj ) or ( not uj and mj and vj
and not lj and not jj and sj and not pj ) or ( uj and mj and not vj and not jj
and not kj and sj and not pj ) or ( mj and vj and not lj and jj and not sj ) or
( not uj and not mj and lj and not jj and kj and sj ) or ( not uj and not mj
and not vj and jj and not kj and sj ) or ( uj and mj and lj and jj and not kj
and sj ) or ( not uj and vj and lj and jj and not kj and sj ) or ( uj and mj
and not vj and jj and kj and not sj ) or ( not uj and mj and lj and jj and kj
and not sj ) or ( mj and not vj and not lj and not jj and kj and not sj ) or (
not uj and not mj and not vj and not lj and not kj and not sj ) or ( uj and not
mj and not lj and not jj and not kj and not sj ) or ( not uj and not vj and lj
and jj and kj and pj ) or ( uj and vj and not lj and jj and not kj and pj ) or
( mj and not vj and not lj and jj and sj and pj ) or ( not uj and mj and not lj
and kj and sj and pj ) or ( uj and not vj and not jj and kj and sj and pj ) or
( not mj and lj and not jj and not kj and sj and pj ) or ( uj and not mj and vj
and jj and not sj and pj ) or ( uj and mj and not vj and lj and jj and not pj )
or ( mj and not vj and not lj and jj and kj and not pj ) or ( mj and vj and lj
and not jj and kj and not pj ) or ( not mj and not vj and not lj and jj and not
kj and not pj ) or ( not uj and not mj and not vj and not jj and not kj and not
pj ) or ( uj and not vj and lj and not jj and not kj and not pj ) or ( uj and
not vj and lj and not jj and sj and not pj ) or ( not uj and vj and not lj and
kj and sj and not pj ) or ( not uj and not mj and vj and not lj and not sj and
not pj ) or ( not uj and not mj and not jj and kj and not sj and not pj ) or (
not uj and mj and not vj and not kj and not sj and not pj ) or ( uj and not mj
and not jj and not kj and not sj and not pj ) or ( uj and not mj and not vj and
not lj and not kj and sj and pj ) or ( not uj and not mj and not vj and jj and
kj and not sj and pj ) or ( not mj and vj and lj and jj and kj and not sj and
pj ) or ( uj and not vj and not lj and jj and kj and sj and not pj ) or ( uj
and mj and vj and jj and not kj and not sj and not P);
result_s(x-8):= y0; result_s(x-7):= y1; result_s(x-6):= y2; result_s(x-5):= y3;
result_s(x-4):= y4; result_s(x-3):= y5; result_s(x-2):= y6; result_s(x-1):= y7;
end loop;
return result_s;
end func_s;
-----
---function S obrat--
function func_so(z: std_logic_vector (127 downto 0)) return std_logic_vector is
variable result_s : std_logic_vector(127 downto 0) := (others => '0');
variable res : std_logic_vector(7 downto 0) := (others => '0');
variable uj, mj, vj, lj, jj, kj, sj, pj, y0, y1, y2, y3, y4, y5, y6, y7 :
std_logic := '0';
variable x : integer := 0;
variable r : integer := 0;
begin
for qqg in 0 to 15 loop
uj:= z(x); x:= x +1;
mj:= z(x); x:= x +1;
vj:= z(x); x:= x +1;
lg:= z(x); x:= x +1;
jj:= z(x); x:= x +1;
kj:= z(x); x:= x +1;
sj:= z(x); x:= x +1;
pj:= z(x); x:= x +1;
y7 := (U and not mj and not lj and jj and not sj ) or ( not mj and vj and not
lj and kj and pj ) or ( not uj and vj and not lj and not jj and not pj ) or (
not uj and not mj and not lj and kj and sj and pj ) or ( not vj and not lj and

```

[illegible]

and sj) or (not uj and not mj and vj and not lj and not jj and sj) or (not
 uj and not mj and vj and jj and kj and sj) or (mj and vj and lj and jj and kj
 and sj) or (not uj and not mj and not vj and not lj and jj and not sj) or (
 uj and not mj and lj and jj and kj and not sj) or (not mj and not vj and lj
 and jj and kj and pj) or (not uj and mj and not vj and jj and sj and pj) or
 (uj and mj and vj and kj and sj and pj) or (not uj and not mj and vj and not
 kj and sj and pj) or (uj and not vj and not sj and kj and not sj and pj) or
 (not mj and vj and lj and not kj and not sj and pj) or (uj and not vj and jj
 and not kj and not sj and pj) or (uj and not mj and vj and lj and not jj and
 not pj) or (uj and not mj and lj and jj and sj and not pj) or (mj and vj
 and not lj and not kj and sj and not pj) or (not uj and not mj and not vj and
 not jj and not sj and not pj) or (uj and vj and lj and not jj and kj and sj
 and pj) or (uj and not mj and not lj and jj and not kj and sj and pj) or (
 uj and mj and not vj and not jj and not kj and sj and pj) or (uj and vj and
 lj and jj and kj and not sj and pj) or (not uj and not mj and not lj and jj
 and kj and not sj and pj) or (not uj and not mj and not vj and lj and kj and
 sj and not pj) or (uj and not mj and vj and jj and kj and not sj and not pj)
 or (uj and not mj and vj and not lj and not kj and not sj and not P);
 y1 := (not uj and mj and not lj and not jj and not kj) or (not vj and lj and
 not jj and sj and not pj) or (uj and mj and not vj and lj and not kj and sj)
 or (uj and vj and lj and not jj and not kj and not sj) or (not uj and not vj
 and lj and jj and kj and pj) or (not uj and mj and vj and not lj and sj and
 pj) or (vj and not lj and jj and kj and sj and pj) or (not uj and lj and jj
 and kj and not sj and pj) or (not mj and not vj and not lj and jj and sj and
 not pj) or (mj and vj and lj and kj and sj and not pj) or (not mj and vj
 and lj and jj and not kj and sj and pj) or (not uj and mj and vj and not jj
 and not kj and sj and pj) or (mj and not vj and lj and jj and kj and not sj
 and pj) or (uj and not mj and not vj and not jj and kj and not sj and pj) or
 (not uj and mj and vj and lj and not jj and kj and not pj) or (not uj and
 not mj and vj and not lj and not jj and kj and not pj) or (uj and not mj and
 not jj and not kj and not pj) or (mj and not vj and not jj and not kj and not
 pj) or (uj and not mj and not lj and jj and kj and sj) or (mj and not vj
 and not lj and not jj and kj and sj) or (uj and not mj and vj and jj and not
 kj and sj) or (mj and not vj and lj and jj and not kj and sj) or (not uj
 and not mj and not vj and not jj and not kj and sj) or (not mj and vj and not
 lj and not jj and kj and not sj) or (uj and not mj and vj and lj and not kj
 and pj) or (not uj and mj and not vj and kj and sj and pj) or (not uj and
 not vj and not lj and jj and not sj and pj) or (not uj and vj and not lj and
 not jj and not sj and pj) or (mj and not vj and not lj and not kj and not sj
 and pj) or (not mj and not vj and jj and not kj and not sj and pj) or (not
 mj and not lj and jj and not kj and not sj and pj) or (uj and not mj and vj
 and lj and not jj and not pj) or (uj and not mj and not vj and not lj and not
 kj and not pj) or (not uj and mj and not vj and not lj and not kj and not pj)
 or (uj and mj and not vj and kj and sj and not pj) or (not uj and vj and not
 lj and jj and not sj and not pj) or (not uj and vj and jj and not kj and not
 sj and not pj) or (uj and mj and vj and not lj and jj and kj and not sj) or
 (uj and not mj and not vj and not lj and not jj and sj and pj) or (uj and mj
 and vj and not jj and kj and sj and pj) or (not uj and mj and not vj and lj
 and not jj and not sj and pj) or (not uj and not mj and lj and not jj and not
 kj and not sj and pj) or (uj and mj and not vj and not lj and jj and kj and
 not pj) or (uj and mj and vj and lj and jj and not kj and not pj) or (not
 mj and vj and lj and jj and kj and not sj and not pj) or (uj and vj and not
 lj and not jj and kj and not sj and not pj) or (not uj and not vj and not lj
 and not jj and kj and not sj and not P);
 y0 := (not uj and mj and lj and not jj and not kj) or (uj and mj and not vj
 and not lj and not sj) or (uj and mj and not jj and kj and not sj) or (uj
 and not vj and not jj and not sj and not pj) or (not uj and not mj and vj and
 lj and not jj and kj) or (not uj and mj and vj and not lj and kj and sj and
 not pj) or (not uj and not mj and not vj and not lj and not pj) or (uj and not vj and not
 jj and not kj and not pj) or (mj and not vj and not jj and not sj and not pj
) or (uj and mj and not vj and lj and jj and not kj) or (uj and not mj and
 vj and not lj and not jj and not kj) or (uj and vj and lj and not jj and kj

```

and sj ) or ( not mj and vj and lj and jj and not kj and sj ) or ( not uj and
not vj and not lj and not jj and not kj and not sj ) or ( not uj and mj and not
vj and not lj and jj and pj ) or ( mj and not vj and not lj and not jj and kj
and pj ) or ( not mj and vj and not lj and jj and not kj and pj ) or ( not uj
and not mj and vj and lj and sj and pj ) or ( not mj and not vj and lj and jj
and sj and pj ) or ( not uj and mj and lj and not jj and sj and pj ) or ( not
uj and not lj and jj and kj and sj and pj ) or ( not uj and lj and jj and not
kj and sj and pj ) or ( not uj and mj and vj and lj and not sj and pj ) or (
not uj and not lj and not jj and not kj and not sj and pj ) or ( uj and not mj
and vj and not lj and kj and not pj ) or ( not uj and vj and not lj and jj and
sj and not pj ) or ( not uj and not vj and lj and not jj and sj and not pj ) or
( mj and vj and jj and kj and sj and not pj ) or ( not uj and mj and not vj and
not kj and sj and not pj ) or ( uj and not mj and vj and jj and not sj and not
pj ) or ( not vj and lj and jj and not kj and not sj and not pj ) or ( uj and
not mj and not vj and not jj and kj and sj and pj ) or ( not uj and not mj and
not vj and not lj and not kj and sj and pj ) or ( not mj and vj and lj and jj
and kj and not sj and pj ) or ( uj and not mj and not lj and jj and not kj and
not sj and pj ) or ( uj and not mj and lj and not jj and not kj and not sj and
pj ) or ( not mj and not vj and lj and not jj and not kj and not sj and pj ) or
( uj and not vj and lj and jj and kj and sj and not pj ) or ( not uj and not mj
and vj and not lj and not kj and sj and not pj ) or ( uj and vj and not lj and
jj and not kj and not sj and not pj);
result_s(x-8):= y0; result_s(x-7):= y1; result_s(x-6):= y2; result_s(x-5):=
y3; result_s(x-4):= y4; result_s(x-3):= y5; result_s(x-2):= y6; result_s(x-1):=
y7;
end loop;
return result_s;
end func_so;
-----
---function r--
function func_r(z: std_logic_vector (127 downto 0)) return std_logic_vector is
variable result_s : std_logic_vector(127 downto 0) := (others => '0');
variable nb, nj, nu, ni, y, jq, jw, je, jr, jb, jt, jy, ju, ji, jo, jp, ja, js,
jd, jf, jg: std_logic_vector(7 downto 0) := (others => '0'); begin
yq:= z(7 downto 0);
yw:= func_c148(z(15 downto 8));
ye:= func_c32(z(23 downto 16));
yr:= func_c133(z(31 downto 24));
yb:= func_c16(z(39 downto 32));
yt:= func_c194(z(47 downto 40));
yy:= func_c192(z(55 downto 48));
yu:= z(63 downto 56);
yi:= func_c251(z(71 downto 64));
yo:= z(79 downto 72);
yp:= func_c192(z(87 downto 80));
ya:= func_c194(z(95 downto 88));
ys:= func_c16(z(103 downto 96));
yd:= func_c133(z(111 downto 104));
yf:= func_c32(z(119 downto 112));
yg:= func_c148(z(127 downto 120));
nb := (not jq and jw and je and jr ) or ( jq and not jw and je and jr ) or ( jq
and jw and not je and jr ) or ( not jq and not jw and not je and jr ) or ( jq
and jw and je and not jr ) or ( not jq and not jw and je and not jr ) or ( not
jq and jw and not je and not jr ) or ( jq and not jw and not je and not jr);
nj := (not jb and jt and jy and ju ) or ( jb and not jt and jy and ju ) or (
jb and jt and not jy and ju ) or ( not jb and not jt and not jy and ju ) or (
jb and jt and jy and not ju ) or ( not jb and not jt and jy and not ju ) or (
not jb and jt and not jy and not ju ) or ( jb and not jt and not jy and not
ju);
nu := (not ji and jo and jp and ja ) or ( ji and not jo and jp and ja ) or (
ji and jo and not jp and ja ) or ( not ji and not jo and not jp and ja ) or (
ji and jo and jp and not ja ) or ( not ji and not jo and jp and not ja ) or (

```

```

not ji and jo and not jp and not ja ) or ( ji and not jo and not jp and not
ja);
ni := (not js and jd and jf and jg ) or ( js and not jd and jf and jg ) or (
js and jd and not jf and jg ) or ( not js and not jd and not jf and jg ) or (
js and jd and jf and not jg ) or ( not js and not jd and jf and not jg ) or (
not js and jd and not jf and not jg ) or ( js and not jd and not jf and not
jg);
y:= (not nb and nj and nu and ni ) or ( nb and not nj and nu and ni ) or ( nb
and nj and not nu and ni ) or ( not nb and not nj and not nu and ni ) or ( nb
and nj and nu and not ni ) or ( not nb and not nj and nu and not ni ) or ( not
nb and nj and not nu and not ni ) or ( nb and not nj and not nu and not ni);

result_s(119 downto 0):= z(127 downto 8); --sdvig
result_s(127 downto 120) := y;
return result_s;
end func_r;
-----
---function r obratonoe--
function func_ro(z: std_logic_vector (127 downto 0)) return std_logic_vector is
variable result_s : std_logic_vector(127 downto 0) := (others => '0');
variable nb, nj, nu, ni, y, jq, jw, je, jr, jb, jt, jy, ju, ji, jo, jp, ja,
js, jd, jf, jg: std_logic_vector(7 downto 0) := (others => '0');
begin
jq:= z(127 downto 120);
yw:= func_c148(z(7 downto 0));
ye:= func_c32(z(15 downto 8));
yr:= func_c133(z(23 downto 16));
yb:= func_c16(z(31 downto 24));
yt:= func_c194(z(39 downto 32));
yy:= func_c192(z(47 downto 40));
yu:= z(55 downto 48);
yi:= func_c251(z(63 downto 56));
yo:= z(71 downto 64);
yp:= func_c192(z(79 downto 72));
ya:= func_c194(z(87 downto 80));
ys:= func_c16(z(95 downto 88));
yd:= func_c133(z(103 downto 96));
yf:= func_c32(z(111 downto 104));
yg:= func_c148(z(119 downto 112));
nb := (not jq and jw and je and jr ) or ( jq and not jw and je and jr ) or ( jq
and jw and not je and jr ) or ( not jq and not jw and not je and jr ) or ( jq
and jw and je and not jr ) or ( not jq and not jw and je and not jr ) or ( not
jq and jw and not je and not jr ) or ( jq and not jw and not je and not jr);
nj := (not jb and jt and jy and ju ) or ( jb and not jt and jy and ju ) or (
jb and jt and not jy and ju ) or ( not jb and not jt and not jy and ju ) or (
jb and jt and jy and not ju ) or ( not jb and not jt and jy and not ju ) or (
not jb and jt and not jy and not ju ) or ( jb and not jt and not jy and not
ju);
nu := (not ji and jo and jp and ja ) or ( ji and not jo and jp and ja ) or (
ji and jo and not jp and ja ) or ( not ji and not jo and not jp and ja ) or (
ji and jo and jp and not ja ) or ( not ji and not jo and jp and not ja ) or (
not ji and jo and not jp and not ja ) or ( ji and not jo and not jp and not
ja);
ni := (not js and jd and jf and jg ) or ( js and not jd and jf and jg ) or (
js and jd and not jf and jg ) or ( not js and not jd and not jf and jg ) or (
js and jd and jf and not jg ) or ( not js and not jd and jf and not jg ) or (
not js and jd and not jf and not jg ) or ( js and not jd and not jf and not
jg);
y:= (not nb and nj and nu and ni ) or ( nb and not nj and nu and ni ) or ( nb
and nj and not nu and ni ) or ( not nb and not nj and not nu and ni ) or ( nb
and nj and nu and not ni ) or ( not nb and not nj and nu and not ni ) or ( not
nb and nj and not nu and not ni ) or ( nb and not nj and not nu and not ni);

```

```

    result_s(127 downto 8) := z(119 downto 0); --sdvig
    result_s(7 downto 0) := y;
    return result_s;
end func_ro;
-----
---function l
function func_l(x : std_logic_vector (127 downto 0)) return std_logic_vector is
    variable result : std_logic_vector(127 downto 0) := (others => '0');
begin
    result:= func_r(x);
    result:= func_r(result);
    result:= func_r(result);
    result:= func_r(result);
    result:= func_r(result);
    result:= func_r(result);
    result:= func_r(result);
    result:= func_r(result);
    result:= func_r(result);
    result:= func_r(result);
    result:= func_r(result);
    result:= func_r(result);
    result:= func_r(result);
    result:= func_r(result);
    result:= func_r(result);
    return result;
end func_l;
-----
---function lsx
    function func_lsx(x, y : std_logic_vector (127 downto 0)) return
std_logic_vector is
    variable result : std_logic_vector(127 downto 0) := (others => '0');
begin
    result:=func_x(x,y);
    result:=func_s(result);
    result :=func_l (result);
    return result;
end func_lsx;
-----
---function f
    function func_f(x : std_logic_vector (255 downto 0); c : std_logic_vector (127
downto 0)) return std_logic_vector is
    variable result : std_logic_vector(255 downto 0) := (others => '0');
    variable key1, key2, lsx: std_logic_vector(127 downto 0) := (others => '0');
begin
    key1(127 downto 0):=x(255 downto 128);
    key2(127 downto 0):=x(127 downto 0);
    lsx:=func_lsx(key1, c);
    result(255 downto 128):= ((not lsx) and key2) or ( lsx and (not key2));
    result(127 downto 0):=key1;
    return result;
end func_f;
-----
---function key
    function func_key(x : std_logic_vector (255 downto 0); c1, c2, c3, c4, c5, c6,
c7, c8 : std_logic_vector (127 downto 0)) return std_logic_vector is
    variable result : std_logic_vector(255 downto 0) := (others => '0');
    variable p1,p2,p3,p4,p5,p6,p7,p8 : std_logic_vector(255 downto 0) := (others
=> '0');
begin
    p1:= func_f(x,c1);
    p2:= func_f(p1,c2);
    p3:= func_f(p2,c3);

```

```

p4:= func_f(p3,c4);
p5:= func_f(p4,c5);
p6:= func_f(p5,c6);
p7:= func_f(p6,c7);
p8:= func_f(p7,c8);
result:=p8;
return result;
end func_key;
-----
---function zashfrovka
function func_za(a : std_logic_vector (127 downto 0); k1, k2, k3, k4, k5, k6,
k7, k8, k9, k10 : std_logic_vector (127 downto 0)) return std_logic_vector is
variable result : std_logic_vector(127 downto 0) := (others => '0');
variable p1,p2,p3,p4,p5,p6,p7,p8,p9,p10: std_logic_vector(127 downto 0) :=
(others => '0');
begin
p1:= func_lsx(k1, a); p2:= func_lsx(p1,k2);
p3:= func_lsx(p2,k3); p4:= func_lsx(p3,k4);
p5:= func_lsx(p4,k5); p6:= func_lsx(p5,k6);
p7:= func_lsx(p6,k7); p8:= func_lsx(p7,k8);
p9:= func_lsx(p8,k9); p10:= func_x(p9,k10);
result:=p10;
return result;
end func_za;
-----
---function l obratnoe
function func_lo(x : std_logic_vector (127 downto 0)) return std_logic_vector
is
variable result : std_logic_vector(127 downto 0) := (others => '0');
begin
result:= func_ro(x);
result:= func_ro(result); result:= func_ro(result);
result:= func_ro(result); result:= func_ro(result);
result:= func_ro(result); result:= func_ro(result);
result:= func_ro(result); result:= func_ro(result);
result:= func_ro(result); result:= func_ro(result);
result:= func_ro(result); result:= func_ro(result);
result:= func_ro(result);
return result;
end func_lo;
-----
---function lsx obratnoe
function func_lsxo(x, y : std_logic_vector (127 downto 0)) return
std_logic_vector is
variable result : std_logic_vector(127 downto 0) := (others => '0');
begin
result:=func_x(x,y);
result:=func_lo(result);
result :=func_so (result);
return result;
end func_lsxo;
-----
---function zashfrovka
function func_ras(a : std_logic_vector (127 downto 0); k1, k2, k3, k4, k5, k6,
k7, k8, k9, k10 : std_logic_vector (127 downto 0)) return std_logic_vector is
variable result : std_logic_vector(127 downto 0) := (others => '0');
variable p1,p2,p3,p4,p5,p6,p7,p8,p9,p10: std_logic_vector(127 downto 0) :=
(others => '0');
begin
p1:= func_lsxo(a, k10);
p2:= func_lsxo(p1,k9);
p3:= func_lsxo(p2,k8);

```

```

p4:= func_lsxo(p3,k7);
p5:= func_lsxo(p4,k6);
p6:= func_lsxo(p5,k5);
p7:= func_lsxo(p6,k4);
p8:= func_lsxo(p7,k3);
p9:= func_lsxo(p8,k2);
p10:= func_x(p9,k1);
result:=p10;
return result;
end func_ras;

-----
begin
Process (clk, key_download, key_delete, mode)
begin
if (key_delete = '1')then
key1<= (others => '0');
key2<= (others => '0');
key34<= (others => '0');
key3<= (others => '0');
key4<= (others => '0');
key56<= (others => '0');
key5<= (others => '0');
key6<= (others => '0');
key78<= (others => '0');
key7<= (others => '0');
key8<= (others => '0');
key910<= (others => '0');
key9<= (others => '0');
key10<= (others => '0');
-----
elsif key_download= '1' then
key1 <=key(255 downto 128);
key2 <=key(127 downto 0);
key34<=func_key(key,c1, c2, c3, c4, c5, c6, c7, c8);
key3 <=key34(255 downto 128);
key4 <=key34(127 downto 0);
key56<=func_key(key34,c9, c10, c11, c12, c13, c14, c15, c16);
key5 <=key56(255 downto 128);
key6 <=key56(127 downto 0);
key78<=func_key(key56,c17, c18, c19, c20, c21, c22, c23, c24);
key7 <=key78(255 downto 128);
key8 <=key78(127 downto 0);
key910<=func_key(key78,c25, c26, c27, c28, c29, c30, c31, c32);
key9 <=key910(255 downto 128);
key10 <=key910(127 downto 0);
end if;
-----
if (clk' event and clk = '1') then
if (mode= '1') then
data_out<= func_zs(data_in,
key1,key2,key3,key4,key5,key6,key7,key8,key9,key10);
elsif (mode= '0') then
data_out <= func_ras(data_in,
key1,key2,key3,key4,key5,key6,key7,key8,key9,key10);
end if;
end if;
end process;
end architecture kuznechik;

```

Приложение Б

(обязательное)

Таблица истинности для S преобразования

Таблица Б.1 -Таблица истинности для S преобразования

x7	x6	x5	x4	x3	x2	x1	x0	y7	y6	y5	y4	y3	y2	y1	y0
0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0
0	0	0	0	0	0	0	1	1	1	1	0	1	1	1	0
0	0	0	0	0	0	1	0	1	1	0	1	1	1	0	1
0	0	0	0	0	0	1	1	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	1	1	0	0	1	1	1	1
0	0	0	0	0	1	0	1	0	1	1	0	1	1	1	0
0	0	0	0	0	1	1	0	0	0	1	1	0	0	0	1
0	0	0	0	0	1	1	1	0	0	0	1	0	1	1	0
0	0	0	0	1	0	0	0	1	1	1	1	1	0	1	1
0	0	0	0	1	0	0	1	1	1	0	0	0	1	0	0
0	0	0	0	1	0	1	0	1	1	1	1	1	0	1	0
0	0	0	0	1	0	1	1	1	1	0	1	1	0	1	0
0	0	0	0	1	1	0	0	0	0	1	0	0	0	1	1
0	0	0	0	1	1	0	1	1	1	0	0	0	1	0	1
0	0	0	0	1	1	1	0	0	0	0	0	0	1	0	0
0	0	0	0	1	1	1	1	0	1	0	0	1	1	0	1
0	0	0	1	0	0	0	0	1	1	1	0	1	0	0	1
0	0	0	1	0	0	0	1	0	1	1	1	0	1	1	1
0	0	0	1	0	0	1	0	1	1	1	1	0	0	0	0
0	0	0	1	0	0	1	1	1	1	0	1	1	0	1	1
0	0	0	1	0	1	0	0	1	1	0	1	1	0	1	1
0	0	0	1	0	1	0	1	0	0	1	0	1	1	1	0
0	0	0	1	0	1	1	0	1	0	0	1	1	0	0	1
0	0	0	1	0	1	1	1	1	0	1	1	1	0	1	0
0	0	0	1	1	0	0	0	0	0	0	1	0	1	1	1
0	0	0	1	1	0	0	1	0	0	1	1	0	1	1	0
0	0	0	1	1	0	1	0	1	1	1	1	0	0	0	1
0	0	0	1	1	0	1	1	1	0	1	1	1	0	1	1
0	0	0	1	1	1	0	0	0	0	0	1	0	1	0	0
0	0	0	1	1	1	0	1	1	1	0	0	1	1	0	1
0	0	0	1	1	1	1	0	0	1	0	1	1	1	1	1
0	0	0	1	1	1	1	1	1	1	0	0	0	0	0	1
0	0	1	0	0	0	0	0	1	1	1	1	1	0	0	1
0	0	1	0	0	0	0	1	0	0	0	1	1	0	0	0
0	0	1	0	0	0	1	0	0	1	1	0	0	1	0	1
0	0	1	0	0	0	1	1	0	1	0	1	1	0	1	0
0	0	1	0	0	1	0	0	1	1	1	0	0	0	1	0
0	0	1	0	0	1	0	1	0	1	0	1	1	1	0	0
0	0	1	0	0	1	1	0	1	1	1	0	1	1	1	1
0	0	1	0	0	1	1	1	0	0	1	0	0	0	0	1

Продолжение таблицы Б.1

x7	x6	x5	x4	x3	x2	x1	x0	y7	y6	y5	y4	y3	y2	y1	y0
0	0	1	0	1	0	0	0	1	0	0	0	0	0	0	1
0	0	1	0	1	0	0	1	0	0	0	1	1	1	0	0
0	0	1	0	1	0	1	0	0	0	1	1	1	1	0	0
0	0	1	0	1	0	1	1	0	1	0	0	0	0	1	0
0	0	1	0	1	1	0	0	1	0	0	0	1	0	1	1
0	0	1	0	1	1	0	1	0	0	0	0	0	0	0	1
0	0	1	0	1	1	1	0	1	0	0	0	1	1	1	0
0	0	1	0	1	1	1	1	0	1	0	0	1	1	1	1
0	0	1	1	0	0	0	0	0	0	0	0	0	1	0	1
0	0	1	1	0	0	0	1	1	0	0	0	0	1	0	0
0	0	1	1	0	0	1	0	0	0	0	0	0	0	1	0
0	0	1	1	0	0	1	1	1	0	1	0	1	1	1	0
0	0	1	1	0	1	0	0	1	1	1	0	0	0	1	1
0	0	1	1	0	1	0	1	0	1	1	0	1	0	1	0
0	0	1	1	0	1	1	0	1	0	0	0	1	1	1	1
0	0	1	1	0	1	1	1	1	0	1	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0	0	0	0	1	1	0
0	0	1	1	1	0	0	1	0	0	0	0	1	0	1	1
0	0	1	1	1	0	1	0	1	1	1	0	1	1	0	1
0	0	1	1	1	0	1	1	1	0	0	1	1	0	0	0
0	0	1	1	1	1	0	0	0	1	1	1	1	1	1	1
0	0	1	1	1	1	0	1	1	1	0	1	0	1	0	0
0	0	1	1	1	1	1	0	1	1	0	1	0	0	1	1
0	0	1	1	1	1	1	1	0	0	0	1	1	1	1	1
0	1	0	0	0	0	0	0	1	1	1	0	1	0	1	1
0	1	0	0	0	0	0	1	0	0	1	1	0	1	0	0
0	1	0	0	0	0	1	0	0	0	1	0	1	1	0	0
0	1	0	0	0	1	0	0	1	1	1	0	1	0	1	0
0	1	0	0	0	1	0	1	1	1	0	0	1	0	0	0
0	1	0	0	0	1	1	0	0	1	0	0	1	0	0	0
0	1	0	0	0	1	1	1	1	0	1	0	1	0	1	1
0	1	0	0	1	0	0	0	1	1	1	1	0	0	1	0
0	1	0	0	1	0	0	1	0	0	1	0	1	0	1	0
0	1	0	0	1	0	1	0	0	1	1	0	1	0	0	0
0	1	0	0	1	0	1	1	1	0	1	0	0	0	1	0
0	1	0	0	1	1	0	0	1	1	1	1	1	1	0	1
0	1	0	0	1	1	0	1	0	0	1	1	1	0	1	0
0	1	0	0	1	1	1	0	1	1	0	0	1	1	1	0
0	1	0	0	1	1	1	1	1	1	0	0	1	1	0	0
0	1	0	1	0	0	0	0	1	0	1	1	0	1	0	1
0	1	0	1	0	0	0	1	0	1	1	1	0	0	0	0

Продолжение таблицы Б.1

x7	x6	x5	x4	x3	x2	x1	x0	y7	y6	y5	y4	y3	y2	y1	y0
0	1	0	1	0	0	1	0	0	0	0	0	1	1	1	0
0	1	0	1	0	0	1	1	0	1	0	1	0	1	1	0
0	1	0	1	0	1	0	0	0	0	0	0	1	0	0	0
0	1	0	1	0	1	0	1	0	0	0	0	1	1	0	0
0	1	0	1	0	1	1	0	0	1	1	1	0	1	1	0
0	1	0	1	0	1	1	1	0	0	0	1	0	0	1	0
0	1	0	1	1	0	0	0	1	0	1	1	1	1	1	1
0	1	0	1	1	0	0	1	0	1	1	1	0	0	1	0
0	1	0	1	1	0	1	0	0	0	0	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	0	0	0	1	1	1
0	1	0	1	1	1	0	0	1	0	0	1	1	1	0	0
0	1	0	1	1	1	0	1	1	0	1	1	0	1	1	1
0	1	0	1	1	1	1	0	0	1	0	1	1	1	0	1
0	1	0	1	1	1	1	1	1	0	0	0	0	1	1	1
0	1	1	0	0	0	0	0	0	0	0	1	0	1	0	1
0	1	1	0	0	0	0	1	1	0	1	0	0	0	0	1
0	1	1	0	0	0	1	0	1	0	0	1	0	1	1	0
0	1	1	0	0	0	1	1	0	0	1	0	1	0	0	1
0	1	1	0	0	1	0	0	0	0	0	1	0	0	0	1
0	1	1	0	0	1	0	1	0	0	0	1	1	0	1	1
0	1	1	0	0	1	1	0	1	0	0	1	1	0	1	0
0	1	1	0	0	1	1	1	1	1	0	0	0	1	1	1
0	1	1	0	1	0	0	0	1	1	1	1	0	0	1	1
0	1	1	0	1	0	0	1	1	0	0	1	0	0	0	1
0	1	1	0	1	0	1	0	0	1	1	1	1	0	0	0
0	1	1	0	1	0	1	1	0	1	1	0	1	1	1	1
0	1	1	0	1	1	0	0	1	0	0	1	1	1	0	1
0	1	1	0	1	1	0	1	1	0	1	1	0	0	1	0
0	1	1	0	1	1	1	1	1	0	1	1	0	0	0	1
0	1	1	1	0	0	0	0	0	0	1	1	0	0	1	0
0	1	1	1	0	0	0	1	0	1	1	1	0	1	0	1
0	1	1	1	0	0	1	0	0	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	0	0	1	1	1	1	0	1
0	1	1	1	0	1	0	0	1	1	1	1	1	1	1	1
0	1	1	1	0	1	0	1	0	0	1	1	0	1	0	1
0	1	1	1	0	1	1	0	1	0	0	0	1	0	1	0
0	1	1	1	0	1	1	1	0	1	1	1	1	1	1	0
0	1	1	1	1	0	0	0	0	1	1	0	1	1	0	1
0	1	1	1	1	0	0	1	0	1	0	1	0	1	0	0
0	1	1	1	1	0	1	0	1	1	0	0	0	1	1	0
0	1	1	1	1	0	1	1	1	0	0	0	0	0	0	0

Продолжение таблицы Б.1

x7	x6	x5	x4	x3	x2	x1	x0	y7	y6	y5	y4	y3	y2	y1	y0
0	1	1	1	1	1	0	0	1	1	0	0	0	0	1	1
0	1	1	1	1	1	0	1	1	0	1	1	1	1	0	1
0	1	1	1	1	1	1	0	0	0	0	0	1	1	0	1
0	1	1	1	1	1	1	1	0	1	0	1	0	1	1	1
1	0	0	0	0	0	0	0	1	1	0	1	1	1	1	1
1	0	0	0	0	0	0	1	1	1	1	1	0	1	0	1
1	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0
1	0	0	0	0	0	1	1	1	0	1	0	1	0	0	1
1	0	0	0	0	1	0	0	0	0	1	1	1	1	1	0
1	0	0	0	0	1	0	1	1	0	1	0	1	0	0	0
1	0	0	0	0	1	1	0	0	1	0	0	0	0	1	1
1	0	0	0	0	1	1	1	1	1	0	0	1	0	0	1
1	0	0	0	1	0	0	0	1	1	0	1	0	1	1	1
1	0	0	0	1	0	0	1	0	1	1	1	1	0	0	1
1	0	0	0	1	0	1	0	1	1	0	1	0	1	1	0
1	0	0	0	1	0	1	1	1	1	1	1	0	1	1	0
1	0	0	0	1	1	0	0	0	1	1	1	1	1	0	0
1	0	0	0	1	1	0	1	0	0	1	0	0	0	1	0
1	0	0	0	1	1	1	0	1	0	1	1	1	0	0	1
1	0	0	0	1	1	1	1	0	0	0	0	0	0	1	1
1	0	0	1	0	0	0	0	1	1	1	0	0	0	0	0
1	0	0	1	0	0	0	1	0	0	0	0	1	1	1	1
1	0	0	1	0	0	1	0	1	1	1	0	1	1	0	0
1	0	0	1	0	1	0	0	0	1	1	1	1	0	1	0
1	0	0	1	0	1	0	1	1	0	0	1	0	1	0	0
1	0	0	1	0	1	1	0	1	0	1	1	0	0	0	0
1	0	0	1	0	1	1	1	0	1	1	1	1	1	0	0
1	0	0	1	1	0	0	0	1	1	0	1	1	1	0	0
1	0	0	1	1	0	0	1	1	1	1	0	1	0	0	0
1	0	0	1	1	0	1	1	0	0	1	0	1	0	0	0
1	0	0	1	1	1	0	0	0	1	0	0	1	1	1	0
1	0	0	1	1	1	0	1	0	0	1	1	0	0	1	1
1	0	0	1	1	1	1	0	0	1	1	0	0	0	1	1
1	0	0	1	1	1	1	1	0	1	0	0	1	0	1	0
1	0	1	0	0	0	0	0	1	0	1	0	0	1	1	1
1	0	1	0	0	0	0	1	1	0	0	1	0	1	1	1
1	0	1	0	0	0	1	0	0	1	1	0	0	0	0	0
1	0	1	0	0	0	1	1	0	1	1	1	0	0	1	1
1	0	1	0	0	1	0	0	0	0	0	1	1	1	1	0
1	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0

Продолжение таблицы Б.1

x7	x6	x5	x4	x3	x2	x1	x0	y7	y6	y5	y4	y3	y2	y1	y0
1	0	1	0	0	1	1	0	0	1	1	0	0	0	1	0
1	0	1	0	0	1	1	1	0	1	0	0	0	1	0	0
1	0	1	0	1	0	0	0	0	0	0	1	1	0	1	0
1	0	1	0	1	0	0	1	1	0	1	1	1	0	0	0
1	0	1	0	1	0	1	0	0	0	1	1	1	0	0	0
1	0	1	0	1	0	1	1	1	0	0	0	0	0	1	0
1	0	1	0	1	1	0	0	0	1	1	0	0	1	0	0
1	0	1	0	1	1	0	1	1	0	0	1	1	1	1	1
1	0	1	0	1	1	1	0	0	0	1	0	0	1	1	0
1	0	1	0	1	1	1	1	0	1	0	0	0	0	0	1
1	0	1	1	0	0	0	0	1	0	1	0	1	1	0	1
1	0	1	1	0	0	0	1	0	1	0	0	0	1	0	1
1	0	1	1	0	0	1	0	0	1	0	0	0	1	1	0
1	0	1	1	0	0	1	1	1	0	0	1	0	0	1	0
1	0	1	1	0	1	0	0	0	0	1	0	0	1	1	1
1	0	1	1	0	1	0	1	0	1	0	1	1	1	1	0
1	0	1	1	0	1	1	0	0	1	0	1	0	1	0	1
1	0	1	1	0	1	1	1	0	0	1	0	1	1	1	1
1	0	1	1	1	0	0	0	1	0	0	0	1	1	0	0
1	0	1	1	1	0	0	1	1	0	1	0	0	0	1	1
1	0	1	1	1	0	1	0	1	0	1	0	0	1	0	1
1	0	1	1	1	0	1	1	0	1	1	1	1	1	0	1
1	0	1	1	1	1	0	0	0	1	1	0	1	0	0	1
1	0	1	1	1	1	0	1	1	1	0	1	0	1	0	1
1	0	1	1	1	1	1	0	1	0	0	1	0	1	0	1
1	0	1	1	1	1	1	1	0	0	1	1	1	0	1	1
1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1
1	1	0	0	0	0	0	1	0	1	0	1	1	0	0	0
1	1	0	0	0	0	1	0	1	0	1	1	0	0	1	1
1	1	0	0	0	0	1	1	0	1	0	0	0	0	0	0
1	1	0	0	0	1	0	0	1	0	0	0	0	1	1	0
1	1	0	0	0	1	0	1	1	0	1	0	1	1	0	0
1	1	0	0	0	1	1	0	0	0	0	1	1	1	0	1
1	1	0	0	0	1	1	1	1	1	1	1	0	1	1	1
1	1	0	0	1	0	0	0	0	0	1	1	0	0	0	0
1	1	0	0	1	0	0	1	0	0	1	1	0	1	1	1
1	1	0	0	1	0	1	0	0	1	1	0	1	0	1	1
1	1	0	0	1	0	1	1	1	1	1	0	0	1	0	0
1	1	0	0	1	1	0	0	1	0	0	0	1	0	0	0
1	1	0	0	1	1	0	1	1	1	0	1	1	0	0	1
1	1	0	0	1	1	1	0	1	1	1	0	0	1	1	1
1	1	0	0	1	1	1	1	1	0	0	0	1	0	0	1

Продолжение таблицы Б.1

x7	x6	x5	x4	x3	x2	x1	x0	y7	y6	y5	y4	y3	y2	y1	y0
1	1	0	1	0	0	0	0	1	1	1	0	0	0	0	1
1	1	0	1	0	0	0	1	0	0	0	1	1	0	1	1
1	1	0	1	0	0	1	0	1	0	0	0	0	0	1	1
1	1	0	1	0	0	1	1	0	1	0	0	1	0	0	1
1	1	0	1	0	1	0	0	0	1	0	0	1	1	0	0
1	1	0	1	0	1	0	1	0	0	1	1	1	1	1	1
1	1	0	1	0	1	1	0	1	1	1	1	1	0	0	0
1	1	0	1	0	1	1	1	1	1	1	1	1	1	1	0
1	1	0	1	1	0	0	0	1	0	0	0	1	1	0	1
1	1	0	1	1	0	0	1	0	1	0	1	0	0	1	1
1	1	0	1	1	0	1	0	1	0	1	0	1	0	1	0
1	1	0	1	1	0	1	1	1	0	0	1	0	0	0	0
1	1	0	1	1	1	0	0	1	1	0	0	1	0	1	0
1	1	0	1	1	1	0	1	1	1	0	1	1	0	0	0
1	1	0	1	1	1	1	0	1	0	0	0	0	1	0	1
1	1	0	1	1	1	1	1	0	1	1	0	0	0	0	1
1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0
1	1	1	0	0	0	0	1	0	1	1	1	0	0	0	1
1	1	1	0	0	0	1	0	0	0	1	1	0	1	1	1
1	1	1	0	0	1	0	0	0	0	1	0	1	0	0	1
1	1	1	0	0	1	1	0	0	0	0	0	1	0	0	1
1	1	1	0	0	1	1	1	0	1	0	1	1	0	1	1
1	1	1	0	1	0	0	0	1	1	0	0	1	0	1	1
1	1	1	0	1	0	0	1	1	0	0	1	1	0	1	1
1	1	1	0	1	0	1	0	0	0	1	0	0	1	0	1
1	1	1	0	1	0	1	1	1	1	0	1	0	0	0	0
1	1	1	0	1	1	0	0	1	0	1	1	1	1	1	0
1	1	1	0	1	1	0	1	1	1	1	0	0	1	0	1
1	1	1	0	1	1	1	0	0	1	1	1	0	1	0	0
1	1	1	0	1	1	1	1	0	1	0	1	0	0	1	0
1	1	1	1	0	0	0	0	0	1	0	1	1	0	0	1
1	1	1	1	0	0	0	1	1	0	1	0	0	1	1	0
1	1	1	1	0	0	1	0	0	0	1	1	0	1	1	0
1	1	1	1	0	1	0	0	1	1	1	1	0	1	0	0
1	1	1	1	0	1	1	0	1	0	1	1	0	1	0	0
1	1	1	1	0	1	1	1	1	1	0	0	0	0	0	0
1	1	1	1	0	1	0	0	1	1	0	1	0	0	0	1
1	1	1	1	0	1	0	1	0	1	1	1	0	1	1	0

Продолжение таблицы Б.1

x7	x6	x5	x4	x3	x2	x1	x0	y7	y6	y5	y4	y3	y2	y1	y0
1	1	1	1	1	0	1	0	1	0	1	0	1	1	1	1
1	1	1	1	1	0	1	1	1	1	0	0	0	0	1	0
1	1	1	1	1	1	0	0	0	0	1	1	1	0	0	1
1	1	1	1	1	1	0	1	0	1	0	0	1	0	1	1
1	1	1	1	1	1	1	0	0	1	1	0	0	0	1	1
1	1	1	1	1	1	1	1	1	0	1	1	0	1	1	0

Приложение В

(обязательное)

Листинг программы испытательного стенда

```

LIBRARY ieee ;
USE ieee.NUMERIC_STD.all ;
USE ieee.std_logic_1164.all ;
USE ieee.STD_LOGIC_SIGNED ;
USE ieee.STD_LOGIC_UNSIGNED.all ;
ENTITY eeee_tb IS
END ;

ARCHITECTURE eeee_tb_arch OF eeee_tb IS
    SIGNAL data_in : std_logic_vector (127 downto 0) ;
    SIGNAL data_out : std_logic_vector (127 downto 0) ;
    SIGNAL mode : BIT ;
    SIGNAL key : std_logic_vector (255 downto 0) ;
    SIGNAL key_download : BIT ;
    SIGNAL clk : BIT ;
    SIGNAL key_delete : BIT ;
    COMPONENT eeee
    PORT (
        data_in : in std_logic_vector (127 downto 0) ;
        data_out : out std_logic_vector (127 downto 0) ;
        mode : in BIT ;
        key : in std_logic_vector (255 downto 0) ;
        key_download : in BIT ;
        clk : in BIT ;
        key_delete : in BIT );
    END COMPONENT ;
BEGIN
    DUT : eeee
    PORT MAP (
        data_in => data_in ,
        data_out => data_out ,
        mode => mode ,
        key => key ,
        key_download => key_download ,
        clk => clk ,
        key_delete => key_delete ) ;

    process
    begin

        clk <= '0';
        key_download <= '1';
        key <= x"8899aabbccddeeff0011223344556677fedcba98765432100123456789abcdef";
        wait for 100ns;

        clk <= '0';
        wait for 100ns;
        clk <= '1';
        wait for 100ns;
        clk <= '0';
        wait for 100ns;
        clk <= '1';
        wait for 100ns;
        clk <= '0';
        wait for 100ns;
        clk <= '1';
        wait for 100ns;
        clk <= '0';
        wait for 100ns;

        mode <= '1';
        clk <= '1';
        data_in <= x"1122334455667700ffeeddccbbaa9988";
    end process;

```

```

wait for 100ns;
-----
clk <= '0';
wait for 100ns;
clk <= '1';
data_in <=x"00112233445566778899aabbccceeff0a";
wait for 100ns;

clk <= '0';
wait for 100ns;

clk <= '1';
mode <= '0';
data_in <=x"7f679d90bebc24305a468d42b9d4edcd";
wait for 100ns;

clk <= '0';
wait for 100ns;

clk <= '1';
data_in <=x"b429912c6e0032f9285452d76718d08b";

wait for 100ns;

clk <= '0';
key_download <= '0';
key_delete <= '1';
wait for 100ns;

clk <= '1';
mode <= '1';
data_in <=x"1122334455667700ffeeddccbbaa9988";
wait for 100ns;
end process;
END ;

```

Приложение Г
(обязательное)
Протокол проверки ВКР в системе «Антиплагиат»

Ректору Пензенского
государственного
университета
А.Д. Гулякову
студента 6 курса ФПИТЭ
очной формы обучения
Читаева Расима Равильевича

ЗАЯВЛЕНИЕ

Я, Читаев Расим Равильевич, студент 6 курса очной формы обучения специальности 10.05.02 – «Информационная безопасность телекоммуникационных систем», прошу провести проверку с использованием системы «Антиплагиат» дипломного проекта на тему «Модель аппаратного модуля, реализующего алгоритм шифрования ГОСТ Р 34.12-2015, построенный на базисных логических элементах», выполненной мной самостоятельно, на содержание элементов плагиата.

Все прямые заимствования из печатных и электронных источников, а также из защищенных ранее выпускных квалификационных работ, кандидатских и докторских диссертаций имеют соответствующие ссылки.

Я ознакомлен с действующим Стандартом университета «Выпускная квалификационная работа обучающихся по образовательным программам высшего образования – программам бакалавриата, программам специалитета и программам магистратуры», согласно которому обнаружение плагиата является основанием для недопуска ВКР к защите и отчисления из университета.

_____/Р.Р. Читаев/

ВКР представлена на проверку _____

Руководитель ВКР _____/С.В. Алексеев/

Протокол

проверки на оригинальность в системе «Антиплагиат.ВУЗ» выпускной квалификационной работы на тему «Модель аппаратного модуля, реализующего алгоритм шифрования ГОСТ Р 34.12-2015, построенный на базисных логических элементах» студента 6 курса Р.Р. Читаева специальности 10.05.02 – «Информационная безопасность телекоммуникационных систем».

Руководитель ВКР ведущий инженер НИС-723 АО «ПНИЭИ» С.В. Алексеев.

Отчет о проверке на заимствования

Автор: Читаев Расим Равильевич rasim.95@yandex.ru / ID: 654

Проверяющий: Фатеев Алексей Григорьевич ibst@pnzgu.ru / ID: 393

Организация: Пензенский гос. Университет

Отчет предоставлен сервисом «Антиплагиат» - <http://pnzgu.antiplagiat.ru>

ИНФОРМАЦИЯ О ДОКУМЕНТЕ

№ документа: 41
Начало загрузки: 22.01.2018 23:02:13
Длительность загрузки: 00:00:08
Имя исходного файла: Читаев Расим Равильевич ВКР_12ПТ1_Читаев.docx
Размер текста: 1568 кБ
Символов в тексте: 178387
Слов в тексте: 32383
Число предложений: 510

ИНФОРМАЦИЯ ОБ ОТЧЕТЕ

Последний готовый отчет (ред.)
Начало проверки: 22.01.2018 23:13:02
Длительность проверки: 00:00:14
Комментарии: не указано
Модули поиска: Модуль поиска ЭБС "БиблиоРоссика", Модуль поиска ЭБС "BOOK.ru", Коллекция РГБ, Цитирование, Модуль поиска ЭБС "Университетская библиотека онлайн", Коллекция eLIBRARY.RU, Модуль поиска ЭБС "Айбукс", Модуль поиска Интернет, Модуль поиска ЭБС "Лань", Модуль поиска "ПГУ", Кольцо вузов

ЗАИМСТВОВАНИЯ	ЦИТИРОВАНИЯ	ОРИГИНАЛЬНОСТЬ
25,76%	0%	74,24%



23.01.2018

С.В. Алексеев