Министерство науки и высшего образования Российской Федерации

Томский государственный университет систем
управления и радиоэлектроники

А.М. Голиков

ИССЛЕДОВАНИЕ МЕТОДА ЗАЩИТЫ ТРАФИКА СИСТЕМ МОБИЛЬНОЙ СВЯЗИ AES-128

Методические указания для выполнения лабораторных работ
для студентов направления 11.03.02 "Инфокоммуникационные технологии и системы связи" и специалитета 11.05.01 "Радиоэлектронные системы и комплексы"

Томск
2020

УДК 621.37
ББК 32.884.1
Г 604

Рецензент
Мещеряков А.А. доцент кафедры радиотехнических систем ТУСУР, канд. техн. наук

Голиков А.М.
Г 604 Исследование метода защиты трафика систем мобильной связи AES-128: методические указания для выполнения лабораторных работ для студентов направления 11.03.02 "Инфокоммуникационные технологии и системы связи" и специалитета 11.05.01 "Радиоэлектронные системы и комплексы" / А.М. Голиков -Томск: Томск. гос. ун-т систем упр. и радиоэлектроники, 2020. - 33 с.

В лабораторной работе проводится исследование метода защиты трафика систем мобильной связи AES-128. Стандарт криптографической защиты AES (Advanced Encryption Standart) в поточном режиме используется в системе мобильной связи LTE. Разработанный программный комплекс позволяет изучить и провести исследование стандарта AES-128. Лабораторная работа предназначена для подготовки магистров по направлению 11.04.02 "Инфокоммуникационные технологии и системы связи" по магистерским программам подготовки: "Радиоэлектронные системы передачи информации", "Оптические системы связи и обработки информации", "Инфокоммуникационные системы беспроводного широкополосного доступа", "Защищенные системы связи", для направления подготовки магистров 11.04.01 "Радиотехника" по магистерской программе подготовки: "Радиоэлектронные устройства передачи информации", "Системы и устройства передачи, приема и обработки сигналов", "Видеоинформационные технологии и цифровое телевидение" и специалитета 11.05.01 "Радиоэлектронные системы и комплексы", а также бакалавриата направления 11.03.02 "Инфокоммуникационные технологии и системы связи".

Одобрено на заседании каф. РТС протокол №8 от 20.04.2020 г.

УДК 621.37
ББК 32.884.1

Оглавление

Введение

Одним из эффективных способов защиты трафика в системах мобильной связи является алгоритм шифрования AES (Advanced Encryption Standard). Высочайшую надежность AES подтверждает астрономическими числами - 128 битный ключ обеспечивает 340 андециллиона (340*1036) возможных комбинаций, а 256 битный ключ увеличивает это число до 11*1076. Для сравнения, старый алгоритм DES, дает общее число комбинаций в 72*1015. На их перебор у специально построенной машины "DES Cracker" уходит несколько часов. Но даже если бы она делала это всего за одну секунду, то на перебор 128 битного ключа машина потратила бы 149 триллионов лет. Межу тем, возраст всей Вселенной ученые оценивают в менее, чем 20 миллиардов лет.

Алгоритм AES в поточном режиме используется для защиты системы мобильной связи LTE. Лабораторная работа позволяет провести изучение и исследование модели криптографической защиты стандарта AES-128.

1 Стандарт LTE

Стандарт сетей LTE – стандарт беспроводной высокоскоростной передачи данных для мобильных телефонов и других терминалов, работающих с данными. Он основан на GSM/EDGE и UMTS/HSPA сетевых технологиях, увеличивая пропускную способность и скорость за счёт использования другого радиоинтерфейса вместе с улучшением ядра сети.На рисунке 1.1 представлена структура сети стандарта LTE.

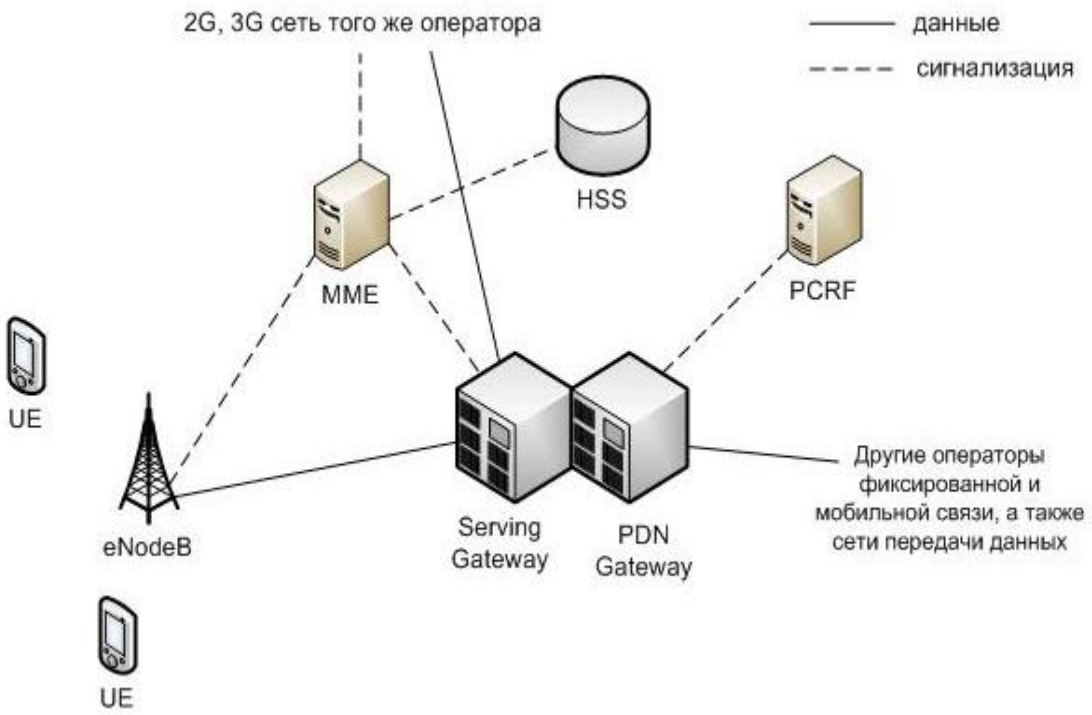


Рисунок 1.1 – Структура сети стандарта LTE

Из этой схемы видно, что структура сети сильно отличается от сетей стандартов 2G и 3G. Существенные изменения претерпела и подсистема базовых станций, и подсистема коммутации. Изменена технология передачи данных между оборудованием пользователя и базовой станцией. Также подверглись изменению и протоколы передачи данных между сетевыми элементами. Вся информация (голос, данные) передается в виде пакетов. Таким образом, уже нет разделения на части обрабатывающие либо только голосовую информацию, либо только пакетные данные.

Можно выделить следующие основные элементы сети стандарта LTE:

• **Serving SAE Gateway** или просто **ServingGateway** (*SGW*) – обслуживающий шлюз сети LTE. Предназначен для обработки и маршрутизации пакетных данных, поступающих из/в подсистему базовых станций. SGW имеет прямое соединение с сетями второго и третьего поколений того же оператора, что упрощает передачу соединения в /из них по причинам ухудшения зоны покрытия, перегрузок и т.п. В SGW нет функции

коммутации каналов для голосовых соединений, т.к. в LTE вся информация, включая голос коммутируется и передается с помощью пакетов.

• **PublicDataNetwork SAE Gateway** или просто **PDN Gateway** (*PGW*) – шлюз к сетям передачи данных других операторов для сети LTE. Основная задача PGW заключается в маршрутизации трафика сети LTE к другим сетям передачи данных, таких как Интернет, а также сетям GSM, UMTS.

• **MobilityManagementEntity** (*MME*) – узел управления мобильностью сети сотовой связи стандарта LTE. Предназначен для обработки сигнализации, преимущественно связанной с управлением мобильностью абонентов в сети.

• **HomeSubscriberServer** (*HSS*) – сервер абонентских данных сети сотовой связи стандарта LTE. Представляет собой большую базу данных и предназначен для хранения данных об абонентах. Кроме того, HSS генерирует данные, необходимые для осуществления процедур шифрования, аутентификации и т.п. Сеть LTE может включать один или несколько HSS. Количество HSS зависит от географической структуры сети и числа абонентов.

• **PolicyandChargingRulesFunction** (*PCRF*) – элемент сети сотовой связи стандарта LTE, отвечающий за управление начислением платы за оказанные услуги связи, а также за качество соединений в соответствии с заданными конкретному абоненту характеристиками.

Для того чтобы данные могли быть транспортированы через интерфейс радио LTE, используются различные «каналы». Они используются для того, чтобы выделять различные типы данных и позволить им транспортироваться через сеть доступа более эффективно. Использование нескольких каналов обеспечивает интерфейс более высокого уровня в рамках протокола LTE и включают более чёткую и определенную сегрегацию данных.

Есть три категории, в которые могут быть сгруппированы различные каналы передачи данных:

• **Логические каналы** – предоставляет услуги среднего уровня управления доступом MAC (MediumAccessControl) в пределах структуры протокола LTE. Логические каналы по типу передаваемой информации делятся на логические каналы управления и логические каналы трафика. Логические каналы управления используются для передачи различных сигнальных и информационных сообщений. По логическим каналам трафика передают пользовательские данные.

• **Транспортные каналы** — транспортные каналы физического уровня предлагают передачу информации в MAC и выше. Информацию логических каналов после обработки на RLC/MAC уровнях размещают в транспортных каналах для дальнейшей передачи по радиоинтерфейсу в физических каналах. Транспортный канал определяет, как и с какими характеристиками происходит передача информации по радиоинтерфейсу. Информационные сообщения на транспортном уровне разбивают на транспортные блоки. В каждом временном интервале передачи (Transmission Time Interval – TTI) по радиоинтерфейсу передают хотя бы один транспортный блок. При

использовании технологии MIMO (Multiple Input Multiple Output) возможна передача до четырех блоков в одном TTI.

• **Физические каналы** – это каналы передачи, которые переносят пользовательские данные и управляющие сообщения. Они изменяются между восходящим и нисходящим потоками, поскольку каждый из них имеет различные требования и действует по-своему.

Методы защиты беспроводных сетей LTE

Безопасность в сетях LTE заключается в нескольких видах:

• Защита абонентов;

• Защита передаваемых сообщений;

• Шифрование сообщений;

• Аутентификация абонента и сети;

Защита абонента заключается в том, что в процессе обслуживания его скрывают временными идентификаторами.

Для закрытия данных в сетях LTE используется потоковое шифрование методом наложения на открытую информацию псевдослучайной последовательности (ПСП) с помощью оператора XOR (исключающее или). В этих сетях для обеспечения безопасности внутри сети применяется принцип туннелирования соединений. Шифрации можно подвергать пакеты S1 и X2 при помощи IPsec ESP, а также подвергаются шифрации сигнальные сообщения этих интерфейсов.

В момент подключения или активизации абонентского оборудования (UE) в сети, сеть запускает процедуру аутентификации и соглашения о ключах AKA (Authentication and Key Agreement). Целью этой процедуры является взаимная аутентификация абонента и сети и выработка промежуточного ключа KASME. Работа механизма AKA занимает доли секунды, которые необходимы для выработки ключа в приложении USIM и для установления соединения с Центром регистрации (HSS). Вследствие этого, для достижения скорости передачи данных сетей LTE необходимо добавить функцию обновления ключевой информации без инициализации механизма AKA. Для решения этой проблемы в сетях LTE предлагается использовать иерархическую ключевую инфраструктуру. Здесь также, как и в сетях 3G, приложение USIM и Центр аутентификации (AuC) осуществляет предварительное распределение ключей. Когда механизм AKA инициализируется для осуществления двусторонней аутентификации пользователя и сети, генерируются ключ шифрования CK и ключ общей защиты, которые затем передаются из ПО USIM в Мобильное оборудование (ME) и из Центра аутентификации в Центр регистрации (HSS). ME и HSS, используя ключевую пару (CK; IK) и ID используемой сети, вырабатывает ключ KASME. Установив зависимость ключа от ID сети, Центр регистрации гарантирует возможность использования ключа только в рамках этой сети. Далее KASME передается из Центра регистрации в устройство мобильного управления (MME) текущей сети, где он используется в качестве мастер-ключа. На основании KASME вырабатывается ключ Knas-enc, который

необходим для шифрования данных протокола NAS между мобильным устройством (UE) и MME, и Knas-int, необходимый для защиты целостности. Когда UE подключается к сети, MME генерирует ключ KeNB и передает его базовым станциям. В свою очередь, из ключа KeNB вырабатывается ключ Kup-enc, используемый для шифрования пользовательских данных протокола U-Plane, ключ Krrc-enc для протокола RRC (RadioResourceControl - протокол взаимодействия между Мобильными устройствами и базовыми станциями) и ключ Krrc-int, предназначенный для защиты целостности.

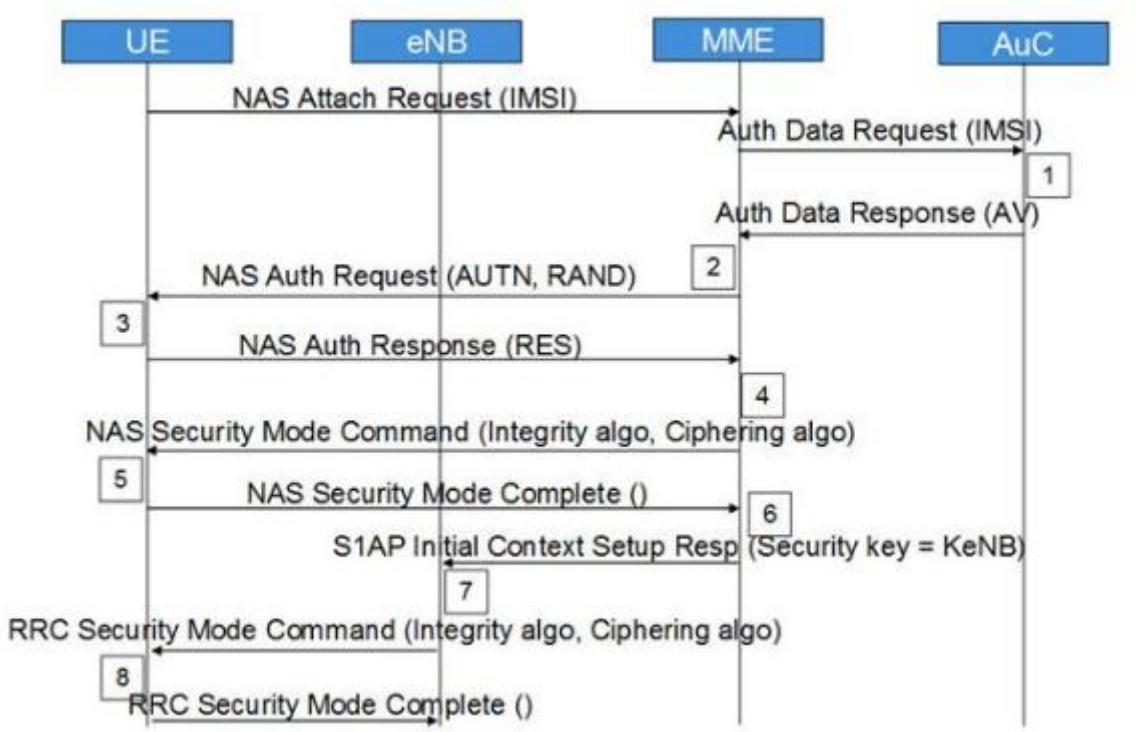


Рисунок 1.2 – Алгоритм аутентификации и генерации ключа

В этом алгоритме такая последовательность:

Шаг 1. Запрос о подключении к сети от мобильной станции (UE). MME запрашивает аутентификационные данные, относящиеся к конкретному IMSI, отправляя Authentication Data Request. AuC/HSS выбирает PSK, относящийся к конкретному IMSI и вычисляет аутентификационные данные по PSK. AuC/HSS отправляет обратно AV с Authentication Data Response.

Шаг 2. MME получает IK, CK, XRES, RAND и AUTH из AV. MME отправляет AUTH и RAND при помощи Authentication Request к UE

Шаг 3. UE аутентифицирует NW, проверяя полученный AUTH. После чего вычисляет IK, CK, RES, XMAC из своего ключа защиты, AMF, (OP), AUTH и RAND. Она отправляет RES с Authentication response.

Шаг 4. После получения RES, MME сравнивает его с XRES и если они совпадают, то аутентификация прошла успешно, в противном случае, MME отправляет сбой аутентификации (Authentication failure) к UE. MME сбрасывает счетчик DL NAS. Рассчитывает KASME, KeNB, Knas-int, Knas-

enc. Отправляет NAS (Non Access Stratum) команду режима безопасности (алгоритм целостности, алгоритм шифрования, NAS набор ключей ID, функцию безопасности UE) с целостностью охраняемых, но не зашифрованных, используя Knas-inc.

Шаг 5. После получения NAS команды режима безопасности, UE вычисляет KASME, KeNB, Knas-int, Knas-enc. UE отправляет NAS режима безопасности выполнен с целостностью, защищенных и зашифрованных.

Шаг 6. После получения NAS команды режима безопасности от UE, MME отправляет KeNB в eNB с S1AP первоначальная установка начального контекста (ключ защиты).

Шаг 7. После получения KeNB, eNB вычисляет Krrc-int, Krrc-enc, Kup-enc. Затем оно отправляет RRC ключ защиты команду с AS (Access Stratum) целостностью алгоритма и AS шифрующий алгоритм.

Шаг 8. После получения RRC команды ключа защиты UE вычисляет Krrc-int, Krrc-enc, Kup-enc. UE отправляет RRC выполненный ключ шифрования на eNB.

После всех описанных действий, все NAS и AS сообщения будут надежно защищены и зашифрованы, в отличие от пользовательских данных, которые будут только шифроваться.

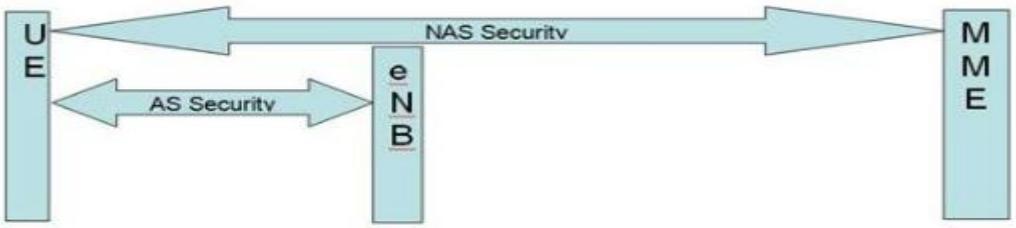


Рисунок 1.3 – Слои безопасности

Архитектура безопасности LTE определяет механизм безопасности и для уровня NAS и для уровня AS.

Безопасность NAS (слоя без доступа): выполнена для NAS сообщений и принадлежит области UE и MME. В этом случае необходима при передаче сообщений NAS между UE и MME – целостность, защищенная и зашифрованная с дополнительным заголовком безопасности NAS.

Безопасность AS (слоя с доступом): выполнена для RRC и плоскости пользовательских данных, принадлежащих области UE и eNB. Уровень PDCP на сторонах UE и eNB отвечает за шифрование и защиту целостности.

RRC сообщения защищены целостностью и зашифрованы, однако данные U-Plane только зашифрованы.

Для генерации векторов аутентификации используется криптографический алгоритм с помощью однонаправленных функций (f1, f2, f3, f4, f5) когда прямой результат получается путем простых вычислений, а обратный результат не может быть получен обратным путем, то есть не существует эффективного алгоритма получения обратного результата. Для

этого алгоритма используется случайное 128 битное случайное число RAND, мастер-ключ K абонента, также 128 бит и порядковый номер процедуры SQN (SequenceNumber). Счетчик SQN меняет свое значение при каждой генерации вектора аутентификации. Похожий счетчик SQN работает и в USIM. Такой метод позволяет генерировать каждый раз новый вектор аутентификации, не повторяя предыдущий уже использованный вектор аутентификации.

Помимо этих трех исходных величин: SQN, RAND и K в алгоритме f1 участвует поле управления аутентификацией AuthenticationManagementField (AMF), а в алгоритмах f2 – f5 исходные параметры – RAND и K, что и продемонстрировано на рис. 2.3, 2.4. На выходах соответствующих функций получают MessageAuthenticationCode (MAC) - 64 бита; XRES – eXpectedResponse, результат работы алгоритма аутентификации <32 – 128 бит>; ключ шифрации CK, генерируемый с использованием входящих (K,RAND)->f3->CK; ключ целостности IK, сгенерированный с использованием входящего (K,RAND)->f4->IK; и промежуточный ключ AnonymityKey (AK), генерируемый с помощью (K,RAND)->f5->AK - 64 бита.

При обслуживании абонента сетью LTE ключи CK и IK в открытом виде в ядро сети не передают. В этом случае HSS генерирует KASME с помощью алгоритма KDF (KeyDerivationFunction), для которого исходными параметрами являются CK и IK, а также идентификатор обслуживающей сети и SQNÅAK. Вектор аутентификации содержит RAND, XRES, AUTN и KASME, на основе которого происходит генерация ключей шифрации и целостности, используемых в соответствующих алгоритмах.

Когда мобильная станция получает из ядра сети три параметра (RAND, AUTN и KSIASME, где KSI – KeySetIdentifier, индикатор установленного ключа, однозначно связанный с KASME в мобильной станции).

После чего используя RAND и AUTN, USIM на основе алгоритмов безопасности, тождественных хранящимся в HSS, производит вычисление XMAC, RES, CK и IK.

Затем в ответе RES UE передает в MME вычисленное RES, которое должно совпасть с XRES, полученным из HSS. Так сеть аутентифицирует абонента. Вычислив XMAC, UE сравнивает его с MAC, полученным ею в AUTN. При успешной аутентификации абонентом сети (MAC = XMAC) UE сообщает об этом в ответе RES. Если аутентификация сети не удалась (MAC ≠ XMAC), то UE направляет в MME ответ CAUSE, где указывает причину неудачи аутентификации.

При успешном завершении предыдущего этапа MME, eNB и UE производят генерацию ключей, используемых для шифрации и проверки целостности получаемых сообщений. В LTE имеется иерархия ключей, которая приведена на рисунке 1.4.

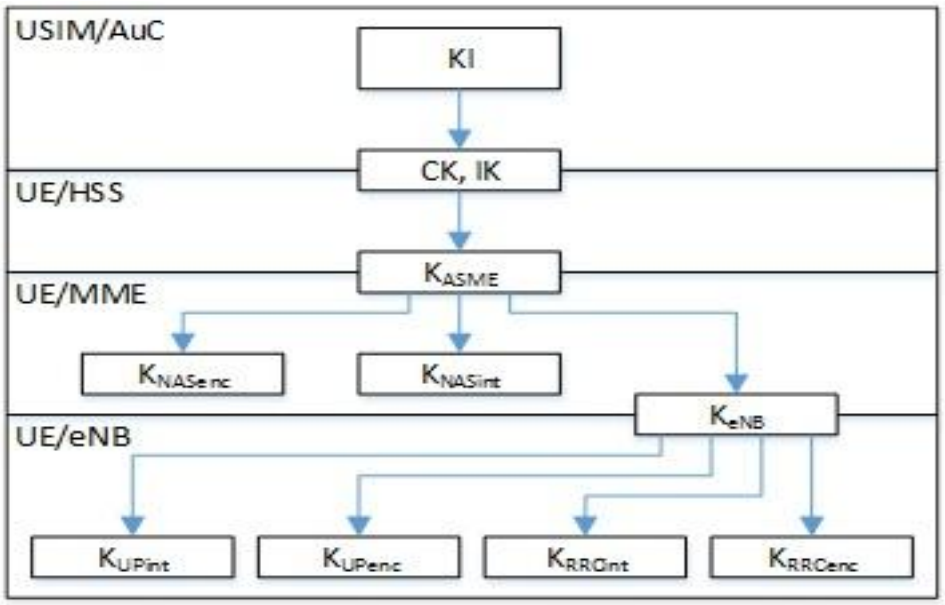


Рисунок 1.4 – Иерархия ключей в LTE

Векторы аутентификации представлены на рисунках 1.5 и 1.6:

- Ключи IK и CK генерируются и в центре аутентификации, и в USIM;
- Ключ AK генерируется только в центре аутентификации;
- Ответ XRES генерируется только в центре аутентификации, а RES генерируется в USIM;
- Код MAC генерируется только в центре аутентификации, а соответствующий ему параметр XMAC генерируется в USIM;
- Маркер AUTH генерируется только в центре аутентификации.

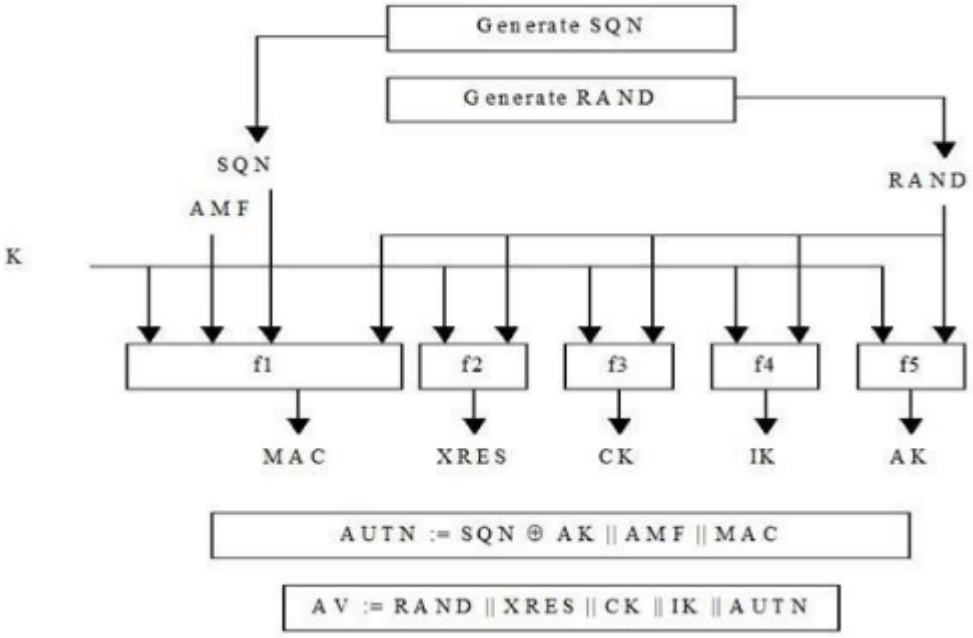


Рисунок 1.5 – Создание векторов на передающей стороне

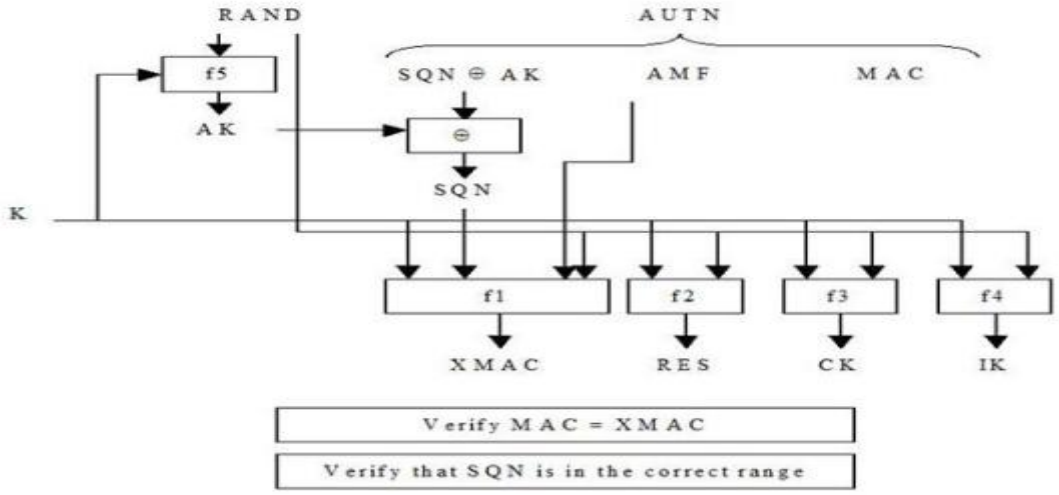


Рисунок 1.6 – Преобразование векторов на приемной стороне

Исходным ключом для всей цепочки является KASME (256 бит). При передаче в радиоканале защиту обеспечивают для сигнального трафика (ControlPlane) и для пользовательских пакетов (UserPlane). При этом все сообщения сигнализации разделяют на сквозные сигнальные сообщения между UE и MME протоколов MM и SM (NAS) и сигнальные сообщения между eNB протокола RRC (AS).

Базовые алгоритмы шифрования

Для шифрации и защиты целостности можно использовать разные базовые алгоритмы:

• UEA2 (UMTSEncryptionAlgorithm 2) иUIA2 (UMTSIntegrityAlgorithm 2);

• AES.

На рисунке 1.7 представлена генерация ключей шифрации и целостности для NAS сигнализации.

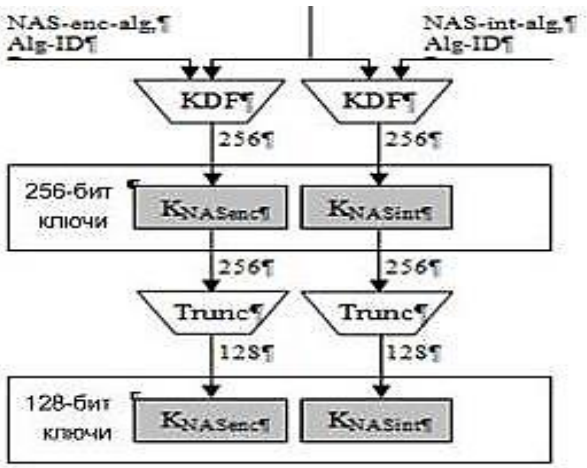


Рисунок 1.7 – Генерирование ключей шифрации и целостности для NAS сигнализации

Сигнальные сообщения протокола RRC (AS) также шифруют и обеспечивают их целостность. Пакеты трафика только шифруют. Эти операции производят в обслуживающей eNB и UE. Схема получения ключей шифрации и целостности для AS и UP трафика отличается от предыдущего случая тем, что исходным параметром здесь служит вторичный промежуточный ключ KeNB (256 бит). Этот ключ генерируют, также используя KDF, где входными параметрами являются: KASME, счетчик сигнальных сообщений NAS вверх, прежнее значение KeNB, идентификатор соты и номер частотного канала в направлении вверх. Следовательно, при каждой периодической локализации UE происходит изменение KeNB.

Также KeNB меняется и при хэндовере; при этом в алгоритме генерации нового KeNB можно использовать дополнительный параметр NH (NextHop), фактически счетчик числа базовых станций, по цепочке обслуживающих абонента. Все реализуемые процедуры безопасности в сети LTE продемонстрированы на рисунке. 1.8.

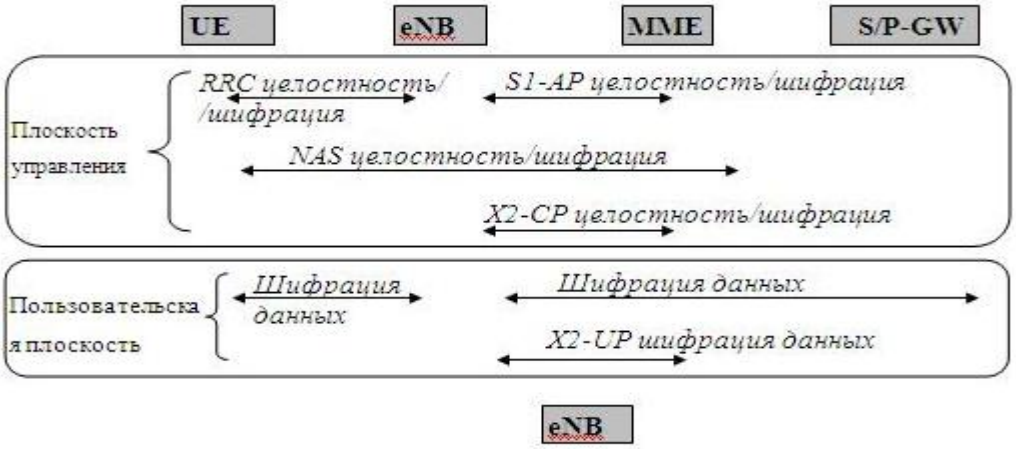


Рисунок 1.8 – Процедуры безопасности в сети LTE

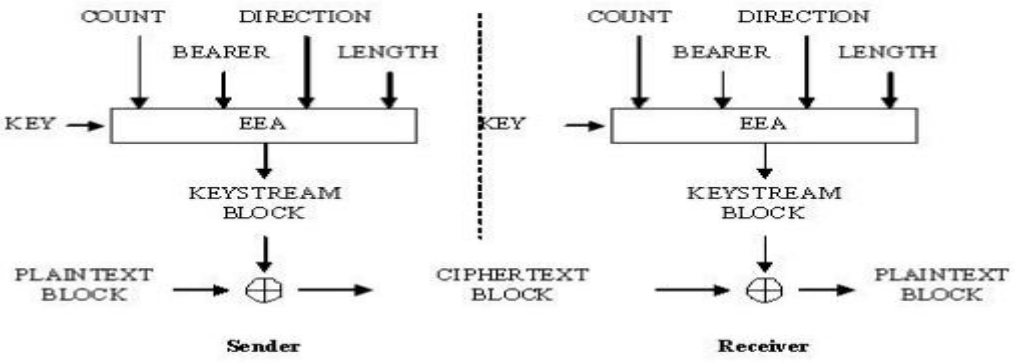


Рисунок 1.9 – Алгоритм шифрации и дешифрации в LTE

Исходными параметрами в этом алгоритме являются шифрующий ключ KEY (128 бит), счетчик пакетов (блоков) COUNT (32 бита), идентификатор сквозного канала BEARER (5 бит), указатель направления передачи

DIRECTION (1 бит) и длина шифрующего ключа LENGTH. В соответствии с выбранным алгоритмом шифрации EEA (EPS Encryption Algorithm) вырабатывается шифрующее число KEYSTREAM BLOCK, которое при передаче складывают по модулю два с шифруемым исходным текстом блока PLAINTEXT BLOCK. При дешифрации на приемном конце повторно совершают эту же операцию. Процедура защиты целостности сообщения состоит в генерации “хвоста“ MAC (MessageAuthenticationCode) (32 бита), присоединяемого к передаваемому пакету. Алгоритм генерации MAC и проверки целостности полученного пакета путем сравнения XMAC с MAC (они должны совпасть) отображен на рисунке 1.10.

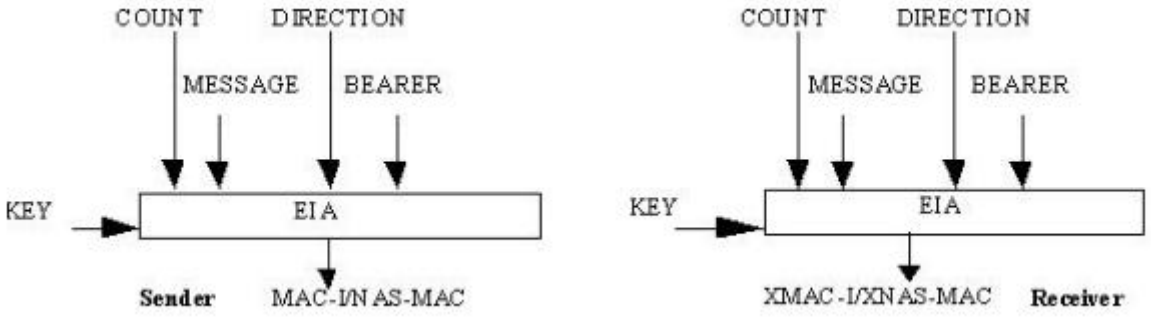


Рисунок 1.10 – Алгоритм проверки целостности

В алгоритме EIA (EPS IntegrityAlgorithm) использован ключ целостности KEY (128 бит), счетчик сообщений COUNT (32 бита), идентификатор сквозного канала BEARER (5 бит), указатель направления передачи DIRECTION (1 бит) и само сообщение MESSAGE.

2 Алгоритм шифрования AES (Advanced Encryption Standard)

AES алгоритм шифрует блок информации длиной 128 бит. Для преобразования применяется расширенный ключ W, формируемый из основного ключа шифрования. 128-битный блок в AES представляется в виде матрицы байтов *4x4*. Длина ключа равна $4 * N_k$ байт и может составлять 128, 192, 256 бит. Алгоритм шифрования состоит из N_r раундов. В таблице 2.1 представлены параметры стандарта AES для разной длины ключа.

Таблица 2.1 – Параметры стандарта AES

	N_k	N_r
AES-128	4	10
AES-192	6	12
AES-256	8	14

На рисунке 2.1 представлен общий алгоритм построения AES шифра.

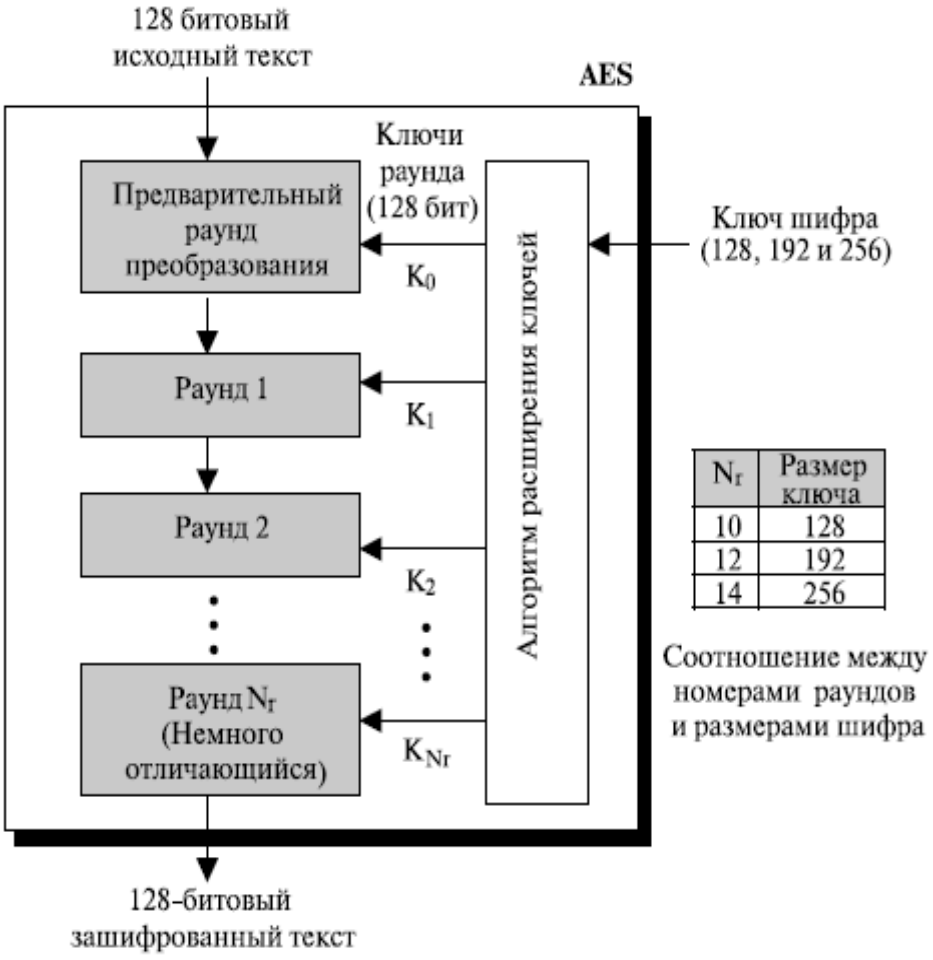


Рисунок 2.1 – Алгоритм AES шифра

AES использует несколько раундов, каждый раунд состоит из нескольких каскадов [5]. Блок данных преобразовывается от одного каскада к другому. В начале и в конце шифра AES применяется термин блок данных; до и после каждого каскада блок данных называется матрицей состояний. Матрицы состояний, подобно блокам, состоят из 16 байтов, но обычно обрабатываются как матрицы *4x4* байтов. В этом случае каждый элемент матрицы состояний обозначается как $S_{r,c}$, где r (от 0 до 3) определяет строку и c (от 0 до 3) определяет столбец. На рисунке 2.2 представлено преобразование из блока данных в матрицу состояния и обратно.

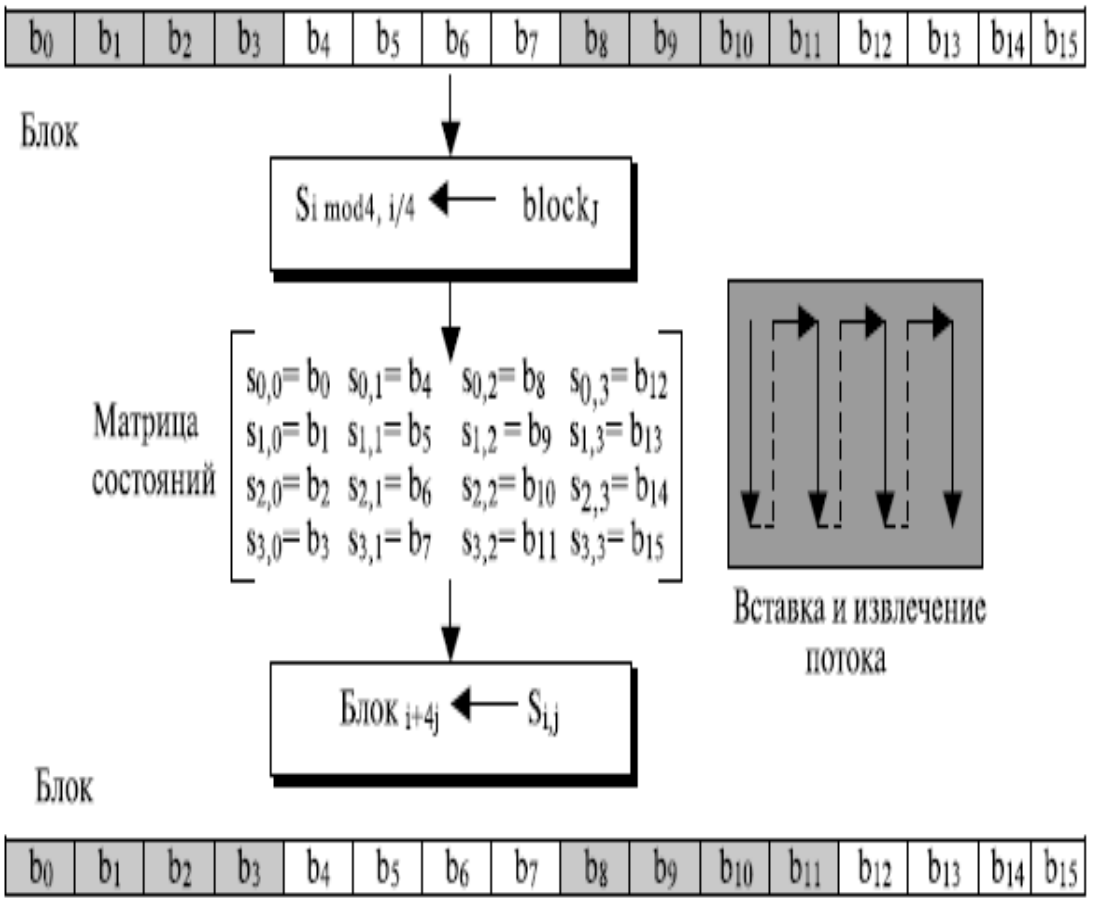


Рисунок 2.2 – Преобразование блок–матрица состояний и наоборот

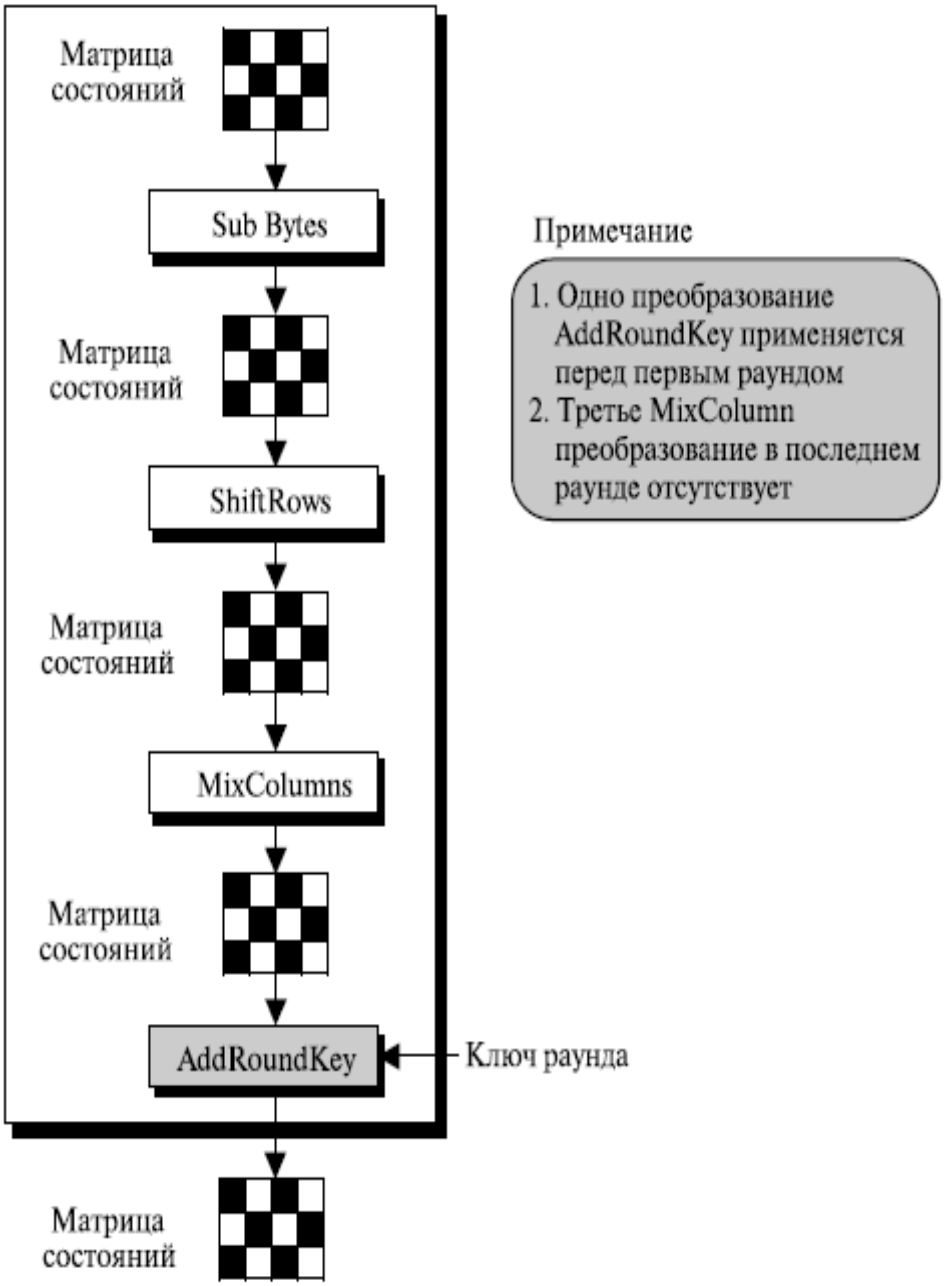


Рисунок 2.3 – Структура раунда шифрования

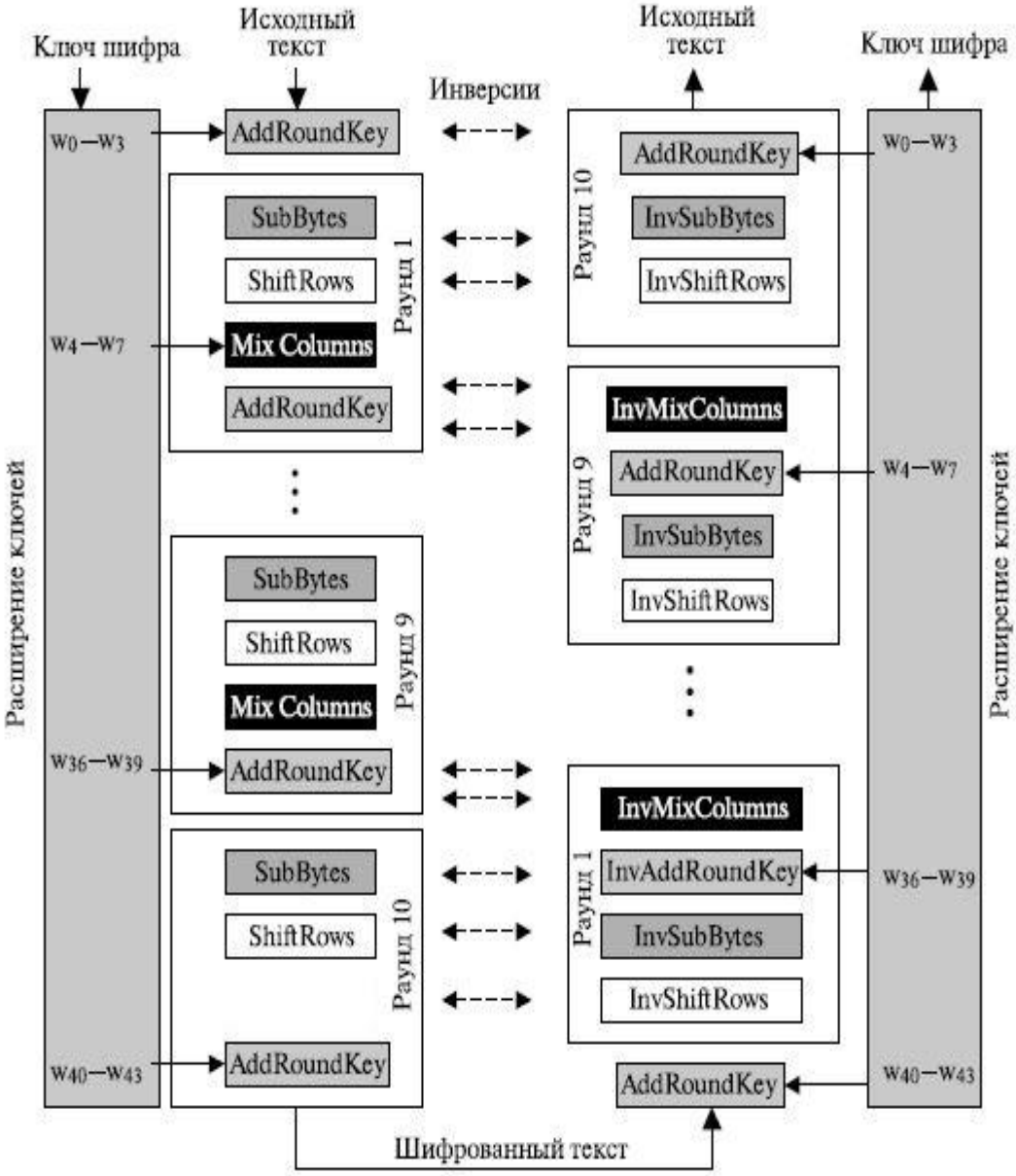


Рисунок 2.4 – Шифрование/дешифрование по алгоритму AES

Для симметричных блочных алгоритмов шифрования определено четыре режима выполнения:

- ECB – Electronic Codebook;
- CBC – Cipher Block Chaining;
- CFB – Cipher Feedback;
- OFB – Output Feedback.

Режим ECB является самым простым режимом, при котором незашифрованное сообщение обрабатывается последовательно, блок за блоком. Каждый блок шифруется, используя один и тот же ключ. Если сообщение длиннее, чем длина блока соответствующего алгоритма, то оно разбивается на блоки соответствующей длины, причем последний блок дополняется при необходимости фиксированными значениями. При

использовании данного режима одинаковые незашифрованные блоки будут преобразованы в одинаковые зашифрованные блоки.

Режим ECB идеален для небольшого количества данных, например, для шифрования ключа сессии.

Существенным недостатком ECB является то, что один и тот же блок незашифрованного сообщения, появляющийся более одного раза в сообщении, всегда имеет один и тот же зашифрованный вид. Вследствие этого для больших сообщений режим ECB считается небезопасным. Если сообщение имеет много одинаковых блоков, то при криптоанализе данная особенность может быть использована.

Для преодоления недостатков ECB используют режим CBC – способ, при котором одинаковые незашифрованные блоки преобразуются в различные зашифрованные. Для этого в качестве входа алгоритма используется результат применения операции XOR к текущему незашифрованному блоку и предыдущему зашифрованному блоку.

Для получения первого блока зашифрованного сообщения используется инициализационный вектор (IV), для которого выполняется операция XOR с первым блоком незашифрованного сообщения. При расшифровании выполняется операция XOR IV с выходом алгоритма расшифрования для получения первого блока незашифрованного текста.

IV должен быть известен как отправителю, так и получателю. Для максимальной безопасности IV должен быть защищен так же, как ключ.

На рисунке 2.5 представлен принцип работы CBC режима.

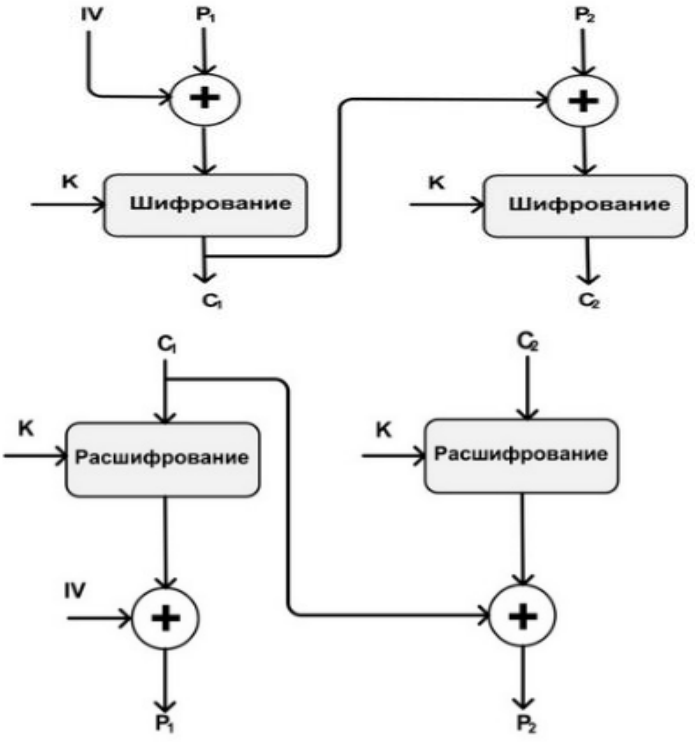


Рисунок 2.5 – Шифрование/дешифрование в режиме CBC

Блочный алгоритм предназначен для шифрования блоков определенной длины. Однако можно преобразовать блочный алгоритм в поточный алгоритм шифрования, используя последние два режима. Поточный алгоритм шифрования устраняет необходимость разбивать сообщение на целое число блоков достаточно большой длины. Таким образом, если передается поток символов, каждый символ может шифроваться и передаваться сразу, с использованием символьно ориентированного режима блочного алгоритма шифрования.

Одним из преимуществ такого режима блочного алгоритма шифрования является то, что зашифрованное сообщение будет той же длины, что и исходное.

Будем считать, что блок данных, используемый для передачи, состоит из J бит; обычным значением является J = 8. Как и в режиме CBC, здесь используется операция XOR для предыдущего блока зашифрованного текста и следующего блока незашифрованного текста. Таким образом, любой блок зашифрованного текста является функцией от всего предыдущего незашифрованного текста.

Входом функции шифрования является регистр сдвига, который первоначально устанавливается в инициализационный вектор IV. Для левых J битов выхода алгоритма выполняется операция XOR с первыми J битами незашифрованного сообщения P1 для получения первого блока зашифрованного сообщения C1. Кроме того, содержимое регистра сдвигается влево на J битов, и C1 помещается в правые J битов этого регистра. Этот процесс продолжается до тех пор, пока не будет зашифровано все сообщение.

При расшифровании используется аналогичная схема, за исключением того, что для блока получаемого зашифрованного сообщения выполняется операция XOR с выходом алгоритма для получения незашифрованного блока.

На рисунке 2.6 представлен принцип режима CFB.

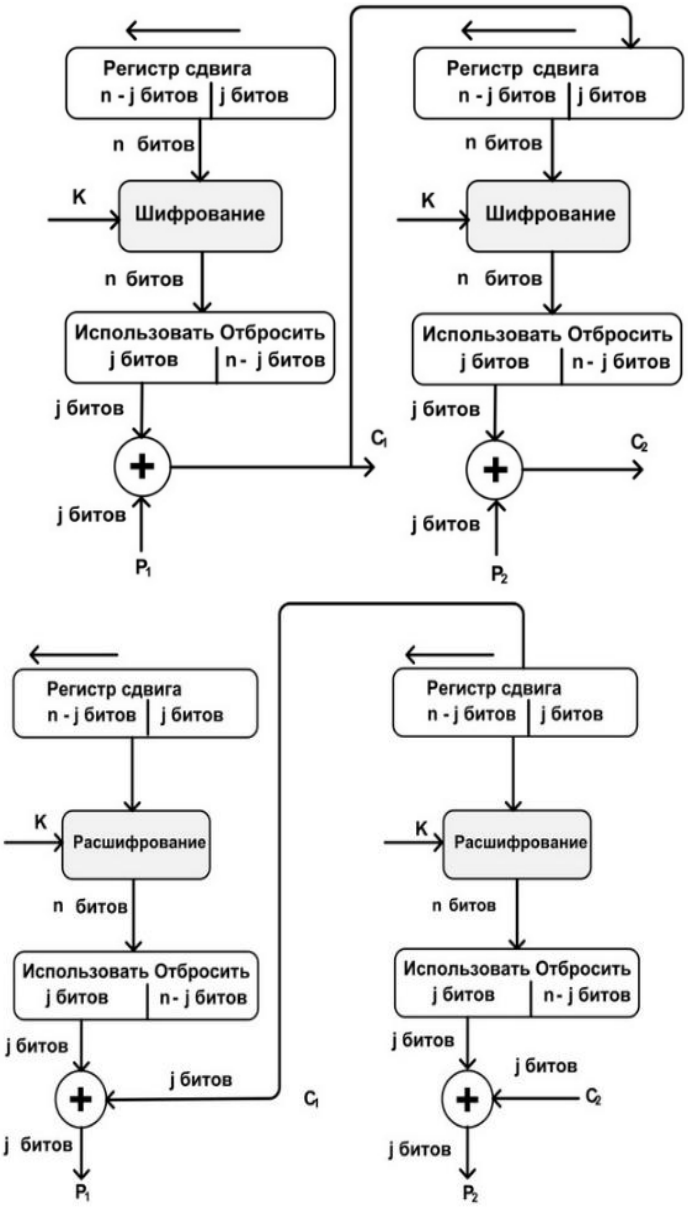


Рисунок 2.6 – Шифрование/дешифрование в режиме CFB

Режим OFB аналогичен режиму CFB. Разница заключается в том, что выход алгоритма в режиме OFB подается обратно в регистр, тогда как в режиме CFB в регистр подается результат применения операции XOR к незашифрованному блоку и результату алгоритма.

Основное преимущество режима OFB состоит в том, что если при передаче произошла ошибка, то она не распространяется на следующие зашифрованные блоки, и тем самым сохраняется возможность расшифрования последующих блоков. Например, если появляется ошибочный бит в C_i, то это приведет только к невозможности расшифрования этого блока и получения P_i. Дальнейшая последовательность блоков будет расшифрована корректно. При использовании режима CFB C_i, подается в качестве входа в регистр и, следовательно, является причиной последующего искажения потока.

Недостаток OFB в том, что он более уязвим к атакам модификации потока сообщений, чем CFB. На рисунке 2.7 представлен принцип режима OFB.

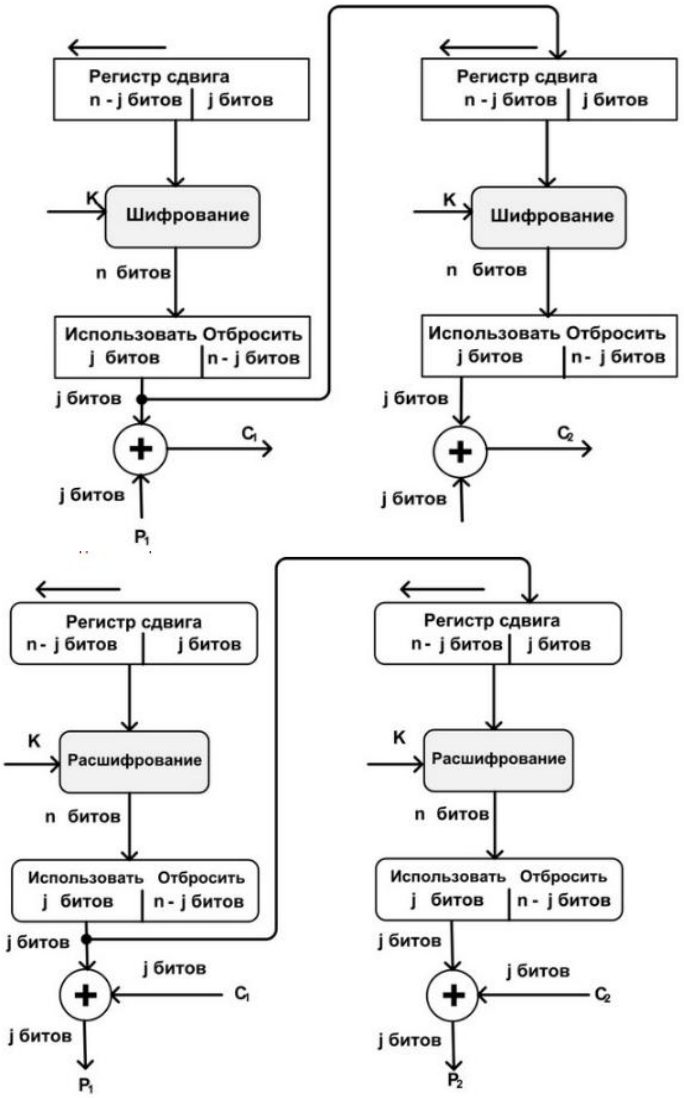


Рисунок 2.7 – Шифрование/дешифрование в режиме OFB

В данной работе будет реализован простой режим ECB, остальные режимы возможно реализовать на его основе.

Основные операции раундов будут рассмотрены ниже.

Расширение ключа KeyExpansion

Алгоритм AES берет ключ шифрования *KEY* выполняет операцию расширения ключа для создания раундовых ключей. Данная операция формирует слова раундового ключа, представляемого в виде вектор-столбца 4 байт. Итоговый расширенный ключ W содержит $4 * (N_r + 1)$ слов, обозначаемых как $w_0, w_1, \dots, w_{4*(N_r+1)-1}$.

На рисунке 2.8 представлена схема формирования расширенного ключа.

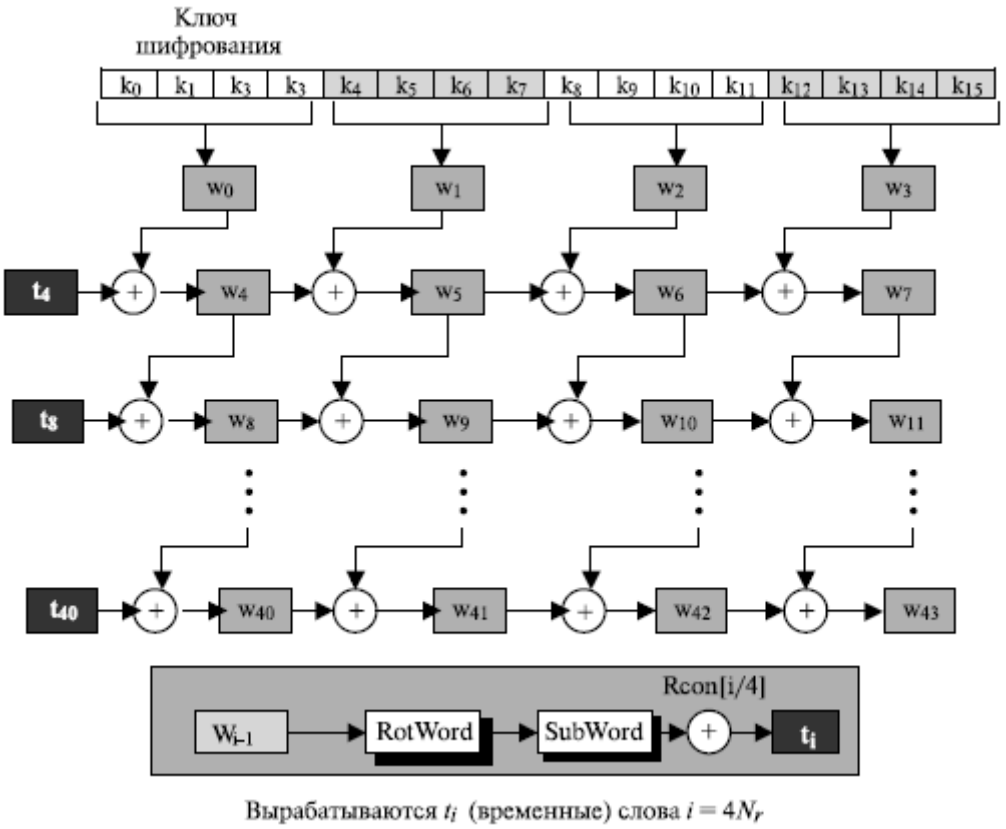


Рисунок 2.8 – Расширение ключа

Алгоритм формирования расширенного ключа для AES-128:

1. Первые четыре слова (матрица *K*, полученная из представления *KEY* в матрицу) равны матрице *K*.

2. Оставшиеся слова w_i, *где* $i = 4 \dots 43$

- $t = w_{i-1}$;
- Если i кратно 4, то $t = SubWord(RotWord(t)) \oplus RCon_{i/4}$;
- $w_i = w_{i-1} \oplus t$;

Здесь $RotWord()$ – функция, которая циклически сдвигает слово из 4 байт $[a_0\ a_1\ a_2\ a_3]$ на 1 байт влево и возвращает $[a_1\ a_2\ a_3\ a_0]$. Массив констант $RCon$ содержит слова, которые для AES-128 имеет значения, представленные на рисунке 2.9.

Раунд	Константа (RCon)	Раунд	Константа (RCon)
1	$(01\ 00\ 00\ 00)_{16}$	6	$(20\ 00\ 00\ 00)_{16}$
2	$(02\ 00\ 00\ 00)_{16}$	7	$(40\ 00\ 00\ 00)_{16}$
3	$(04\ 00\ 00\ 00)_{16}$	8	$(80\ 00\ 00\ 00)_{16}$
4	$(08\ 00\ 00\ 00)_{16}$	9	$(1B\ 00\ 00\ 00)_{16}$
5	$(10\ 00\ 00\ 00)_{16}$	10	$(36\ 00\ 00\ 00)_{16}$

Рисунок 2.9 – Таблица данных $RCon$

Функция *SubWord()* аналогична *SubBytes()*, которая будет рассмотрена ниже. Листинг функции *KeyExpand* представлен в приложении А.

Преобразование SubBytes/InvSubBytes

Функция *SubByte* производит нелинейную замену байт. Для этого значение байта представляется в шестнадцатеричной форме. Левая цифра определяет строку, а вторая – столбец в таблице замен. Для AES-128 эта таблица замен *S-box* фиксирована и представлена в таблице 2.2.

Таблица 2.2 – Таблица *S-box*

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	FO	AD	D4	A2	AF	9C	A4	72	CO
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	AO	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	DO	EF	AA	FB	43	4D	33	85	45	F9	02	F7	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DE
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	CB	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	IF	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	OE	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	OD	BF	E6	42	68	41	99	2D	OF	BO	54	BB	16

Принцип работы функции изображен на рисунке 2.10.

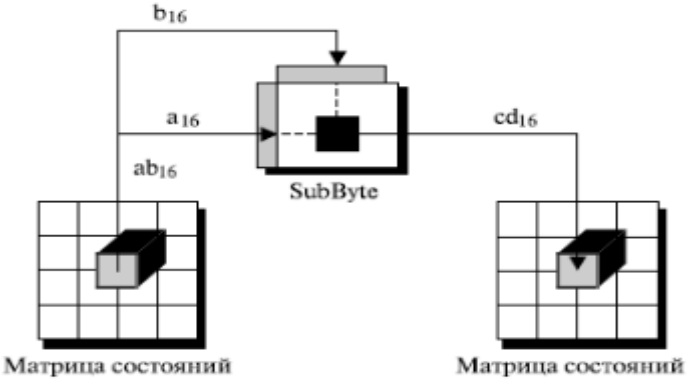


Рисунок 2.10 – Функция *SubBytes*

Инверсным (обратным) преобразованием к функции *SubBytes* является *InvSubBytes*, принцип которой полностью аналогичен, только лишь используется другая таблица замен, представленная в таблице 2.3.

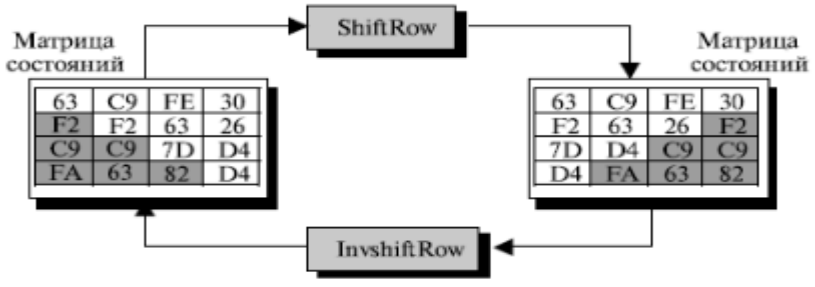
Таблица 2.3 – Таблица *InvS-box*

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
8	3A	91	11	41	4F	67	DC	*E4*	97	F2	CF	CE	F0	B4	E6	73
9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DE	6E
A	47	E1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
B	*FC*	56	3E	4B	C6	D2	79	20	9A	*DB*	C0	*FE*	78	CD	*54*	F4
C	1F	D	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
D	60	51	7E	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
E	A0	E0	3B	4D	AE	2A	F5	B0	CB	EB	BB	3C	83	53	99	61
F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

Листинги функций *SubBytes* и *InvSubBytes* представлены в приложении Преобразование ShiftRows/InvShiftRows

В преобразовании *ShiftRows* байты в последних трех строках матрицы состояния (*state*) циклически смещаются влево на различное число байт. Строка 1 (нумерация строк начинается с нуля) смещается на один байт, строка 2 – на два байта, строка 3 – на три байта.

Функция *InvShiftRows* выполняет обратное преобразование. На рисунке 2.11 проиллюстрировано применение преобразования *ShiftRows* и *InvShiftRows* к *state.*



с

Рисунок 2.11 – Функции *ShiftRows* и *InvShiftRows*

Листинг функций *ShiftRows* и *InvShiftRows* представлен в приложении Преобразование MixColumns/InvMixColumns

В преобразовании *MixColumns* – перемешивание столбца – столбцы *state* рассматриваются как полиномы над полем $F(2^8)$ и умножаются по модулю
$x^8 + x^4 + x^3 + x + 1$ на постоянный полином $a(x) = 3x^3 + x^2 + x + 2$. Умножение полиномов эквивалентно умножению на матрицу:

$$\begin{bmatrix} S'_{0i} \\ S'_{1i} \\ S'_{2i} \\ S'_{3i} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \cdot \begin{bmatrix} S_{0i} \\ S_{1i} \\ S_{2i} \\ S_{3i} \end{bmatrix}$$

Функция *InvMixColumns* инверсна *MixColumns*, что можно выразить взаимной обратимостью матриц:

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \quad \text{Инверсия} \quad \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix}$$

C $\qquad$ C^{-1}

Листинг данных функций представлен в приложении Г.

Добавление раундового ключа AddRoundKey

В преобразовании *AddRoundKey* раундовый ключ добавляется к матрице состояния *state*. Раундовый ключ для раунда k выбирается из расширенного ключа как слова с $w_{4k} \dots w_{4(k+1)-1}$. Преобразование *AddRoundKey* инверсно само себе, так как XOR инверсно само себе.

Пример шифрования

Для пояснения принципа шифрования представлен пример с параметрами [6], указанными в таблице 2.4.

Таблица 2.4 – Пример шифра

Исходный текст (шестнадцатеричный)	00 04 12 14 12 04 12 00 0C 00 13 11 08 23 19 19
Исходный текст (десятеричный)	0 4 18 20 18 4 18 0 12 0 19 17 8 35 25 25
Ключ шифрования (шестнадцатеричный)	24 75 A2 B3 34 75 56 88 31 E2 12 00 13 AA 54 87
Ключ шифрования (десятеричный)	36 117 162 179 52 117 86 136 49 226 18 0 19 170 84 135
Зашифрованный текст (шестнадцатеричный)	BC 02 8B D3 E0 E3 B1 95 55 0D 6D FB E6 F1 82 41
Зашифрованный текст (десятеричный)	188 2 139 211 224 227 177 149 85 13 109 248 230 241 130 65

На рисунке 2.12 представлен процесс шифрования каждого раунда алгоритма.

Раунд	Входная матрица состояний	Выходная матрица состояний	Ключ раунда
Предварительный раунд	00 12 *OC* 08	24 26 3D 1B	24 34 31 13
	04 04 00 23	71 71 E2 89	75 75 E2 AA
	12 12 13 19	BO 44 01 4D	A2 56 12 54
	14 00 11 19	A7 88 11 9E	B3 88 00 87
1	24 26 3D 1B	6C 44 13 BD	89 BD 8C 9F
	71 71 E2 89	Bl 9E 46 35	55 20 C2 68
	BO 44 01 4D	C5 B5 F3 02	B5 E3 F1 A5
	A7 88 11 9E	5D 87 *FC* 8C	CE 46 46 C1
2	6C 44 13 BD	1A 90 15 B2	CE 73 FF 60
	Bl 9E 46 35	66 09 ID *FC*	53 73 Bl D9
	C5 B5 F3 02	20 55 *5A* B2	CD 2E DF 7A
	5D 87 PC 8C	2B CB 8C 3C	15 53 15 D4
3	1A 90 15 B2	F6 7D A2 BO	FF 8C 73 13
	66 09 ID *FC*	1B 61 B4 B8	89 FA 4B 92
	20 55 *5A* B2	67 09 C9 45	85 AB 74 OE
	2B CB 8C 3C	4A 5C 51 09	C5 96 83 57
4	F6 7D A2 BO	CA E5 48 BB	B8 34 47 54
	1B 61 B4 B8	D8 42 AF 71	22 D8 93 01
	67 09 C9 45	Dl BA 98 2D	DE 75 01 OF
	4A 5C 51 09	4E 60 9E DF	B8 2E AD FA
5	CA E5 48 BB	90 35 13 60	D4 EO A7 F3
	D8 42 AF 71	2C FB 82 3A	54 8C IF IE
	Dl BA 98 2D	9E *FC* 61 ED	F3 86 87 88
	4E 60 9E DF	49 39 CB 47	98 B6 1B El

Рисунок 2.12 – Пример шифрования

На рисунке 2.13 представлен пример 7-го раунда.

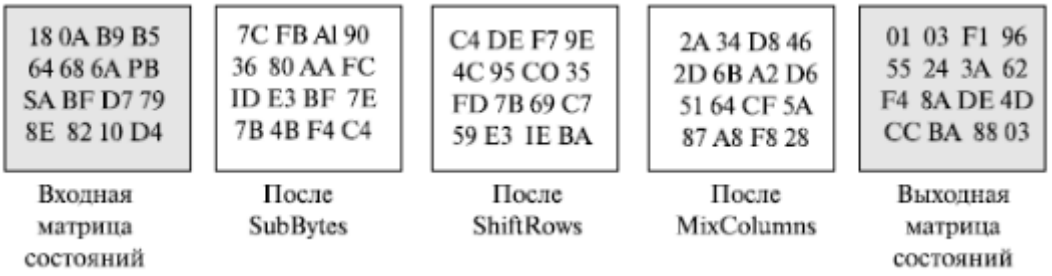


Рисунок 2.13 – 7-й раунд шифрования из примера

3 Моделирование алгоритма AES-128

Согласно разработанному программному коду алгоритма (приложение А), была разработана соответствующая модель в *Simulink*. На рисунке 3.1 представлена модель AES-128 в режиме ECB.

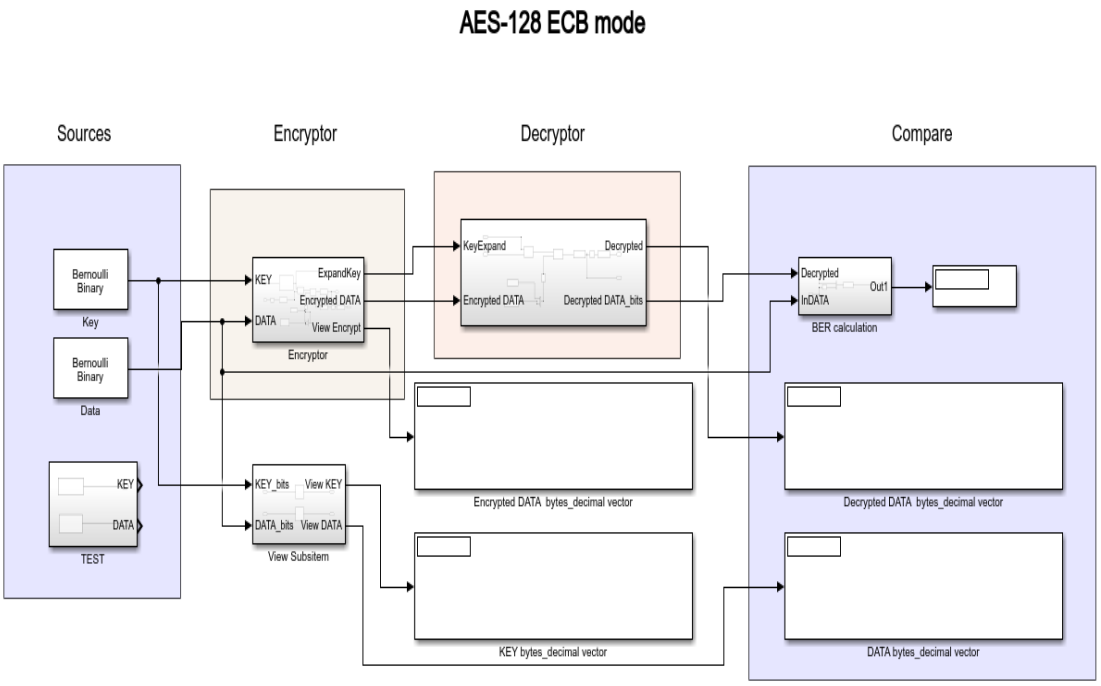


Рисунок 3.1 – Модель AES-128

Основными параметрами являются ключ шифрования (*Key*) и данные (*Data*), которые в данной модели представлены блоками *Bernoulli Binary Generator*, имитирующие 128-битные блоки данных. Помимо генераторов случайных данных в модели имеются тестовые параметры *TEST*, которые соответствуют таблице 2.4. На рисунке 3.2 представлены параметры блока *Bernoulli Binary Generator.*

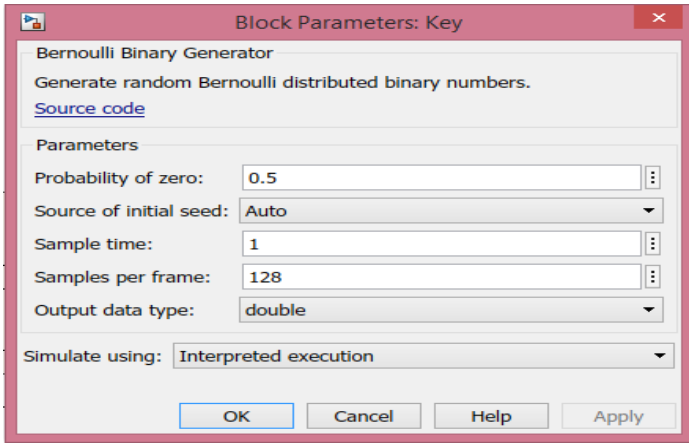


Рисунок 3.2 – Параметры блока Bernoulli Binary Generator

Подсистема шифратора (*Encryptor*) представлена на рисунке 3.3.

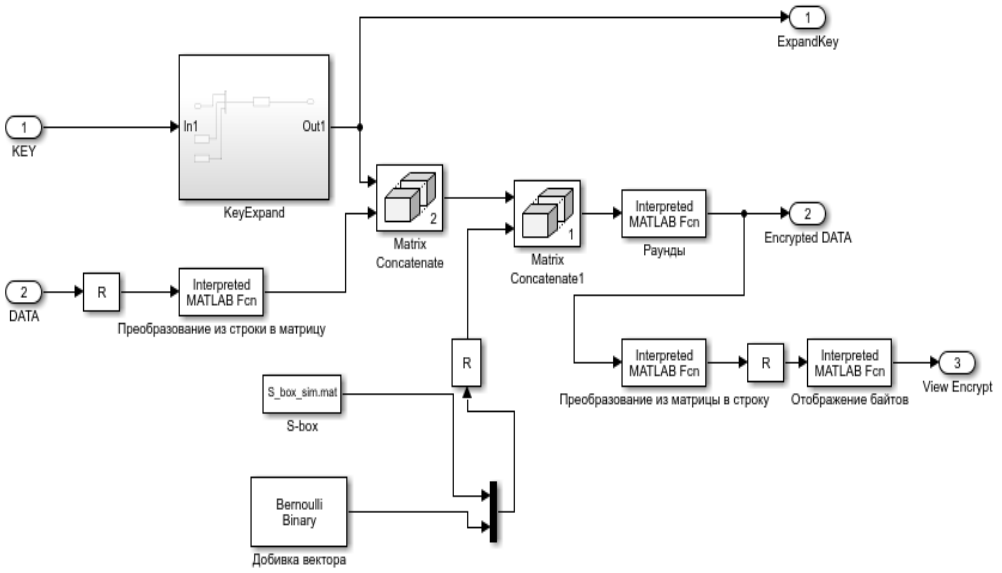


Рисунок 3.3 – Подсистема Encryptor

Подсистема дешифратора (*Decryptor*) представлена на рисунке 3.4.

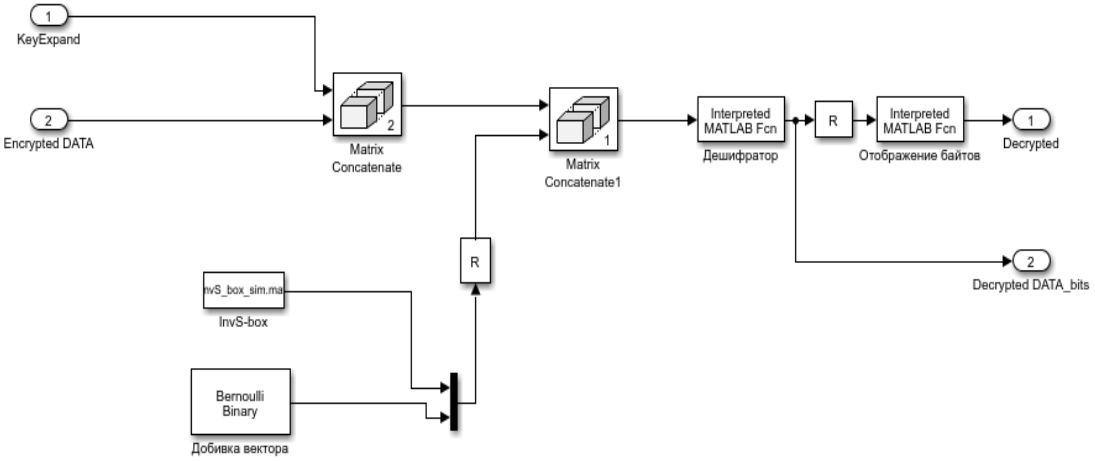


Рисунок 3.4 – Подсистема Decryptor

Для демонстрации правильной работы модели было проведено моделирование с параметрами из таблицы 2.4, в результате полученные данные совпали. Для независимой проверки работы алгоритма AES можно воспользоваться онлайн-шифратором AES. На рисунке 3.5 представлены результаты работы модели.

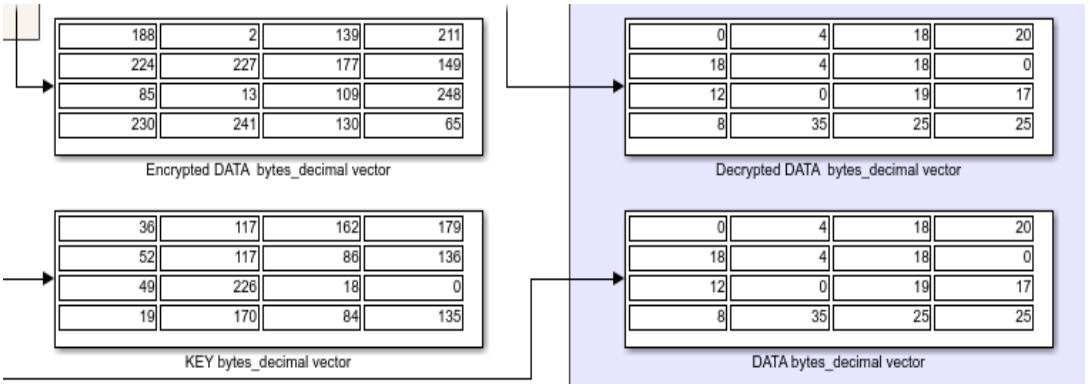


Рисунок 3.5 – Результаты моделирования

Также был разработан алгоритм для шифрования/дешифрования текстового файла формата *.txt*. Программа шифрует и дешифрует данные, после чего в командном окне выводится основная информация (рисунок 3.6) и формируется файл *Log_file.txt*. Поскольку программный пакет MATLAB не поддерживает корректную работу с кириллицей, исходный текстовый файл состоит из латинского алфавита.

С помощью данной программы была исследована зависимость шифрования/дешифрования от объема файла. Результаты представлены в таблице 3.1.

Таблица 3.1 – Результаты исследования

Объем файла, Б	1	2	5	10	65	167	421	1322	2082
Время шифрования, с	0.0612	0.0637	0.0687	0.0721	0.186	0.472	1.06	3.31	5.535
Время дешифрования, с	0.0552	0.0569	0.05582	0.063	0.1867	0.503	1.19	4.58	7.235

На рисунке 3.7 представлены зависимости времени шифрования/дешифрования от объема файла.

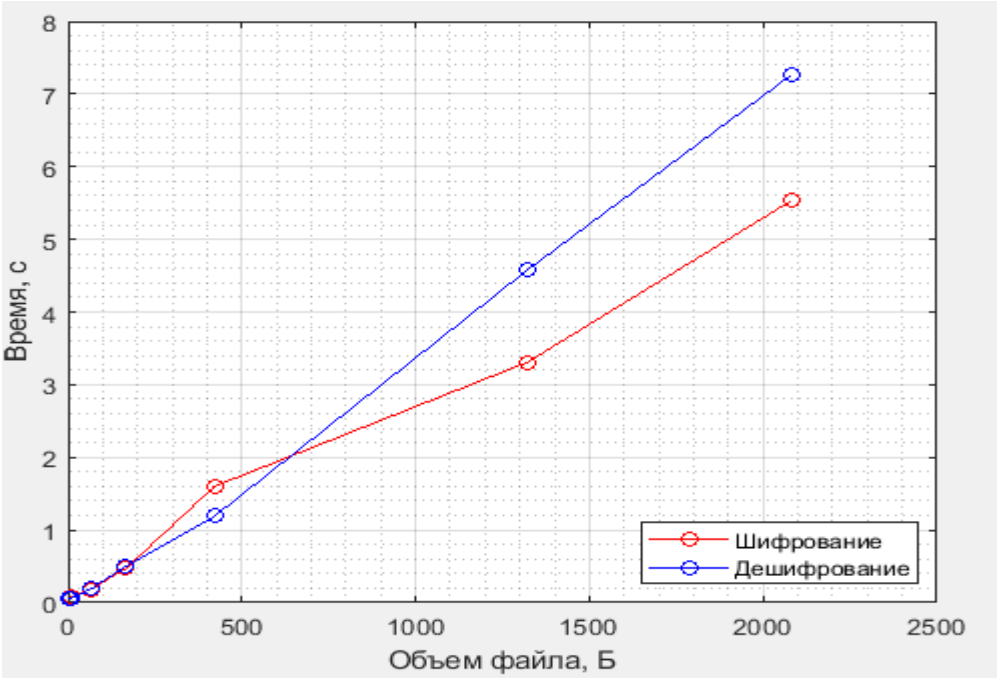


Рисунок 3.7 – Зависимость времени шифрования/дешифрования

Как видно из рисунка 3.7, затраты времени на обработку линейно зависят от объема файла. Однако, полученные значения зависят от аппаратной составляющей компьютера, его загруженности, структуры кода программы.

Методика работы с ПО представлена в приложении.

В результате проделанной работы были получены основные понятия о структуре сетей LTE, рассмотрены существующие методы и стандарты защиты беспроводных сетей LTE. Представлен современный симметричный блочный алгоритм шифрования AES, обеспечивающий надежную защиту информации, и его режимы работы.

Результатом работы является учебный программный комплекс, реализующий алгоритм криптографической защиты AES-128 в режиме ECB. Данный комплекс шифрует и дешифрует информацию согласно указанному алгоритму, также имеется возможность наблюдать исходные данные, ключ шифрования, закодированное и расшифрованное сообщение. Для индикации правильности работы используется подсчет ошибок исходного и расшифрованного сообщения.

Данный учебный программный комплекс может использоваться в учебных целях.

Содержание отчета

1. Теоретическая часть.
2. Моделирование алгоритма AES-128.
3. Результаты исследования.
4. Выводы по работе.

Контрольные вопросы

1. Что такое симметричное шифрование?
2. Что представляет собой стандарт AES (длина ключа, размер входного блока)?
3. Какой алгоритм выбран в качестве стандарта AES?
4. Что собой представляет архитиктура данного стандарта?
5. Из чего состоит один раунд?
6. Сколько раундов шифрования предусмотрено стандартом?
7. Что быстрее шифрование или разшифрование? Почему?
8. Какие режимы шифрования применяются в стандарте AES?
9. Какие режимы быстрее при разшифровании? Почему?
10. Какие режимы лучше восстанавливают зашифрованную информацию при ошибке в одном символе? Почему?

Литература

1. Голиков А.М. Модуляция, кодирование и моделирование в телекоммуникационных системах. Теория и практика: Учебное пособие / А.М. Голиков. - СПб.: Издательство «Лань», 2018. – 452с.

2. AES Encryption: Encrypt and decrypt online [Электронный ресурс]. — Режим доступа: https://cryptii.com/pipes/aes-encryption (дата обращения: 16.03.2020).

3. Зензин О.С. Стандарт криптографической защиты - AES. Конечный поля / О.С. Зензин, М.А. Иванов, под ред. М.А. Иванова - М.: КУДИЦ-ОБРАЗ, 2002. - 176 с.

ПРИЛОЖЕНИЕ
Методика работы с ПО

Для запуска *Simulink* модели необходимо:

1. Открыть **Matlab**;
2. Установить путь к папке **Simulink model** во вкладке *Set Path* (рисунок П.1);

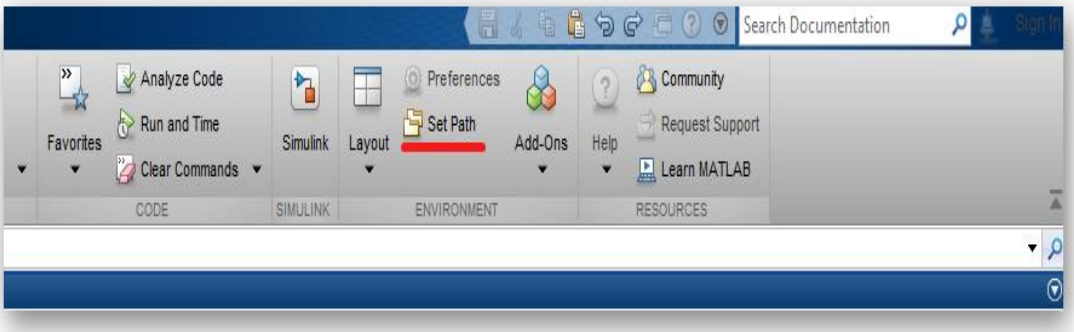


Рисунок П.1 – Меню Matlab

3. Открыть файл **AES128.slx**;
4. Запустить модель.

Для запуска модели в виде программного кода для обработки файла необходимо:

1. Открыть **Matlab**;
2. Установить путь к папке **Code model** во вкладке *Set Path*;
3. Открыть файл **Main_AES.m**;
4. Открыть в директории файл **file.txt** и внести/изменить данные. Вводить только латинские символы, не вводить « – » (тире). Сохранить файл и закрыть его.
5. Запустить файл **Main_AES.m** (рисунок П.2).
6. Для получения сведений по результатам работы программы использовать командное окно либо открыть файл **Log_file.txt**.

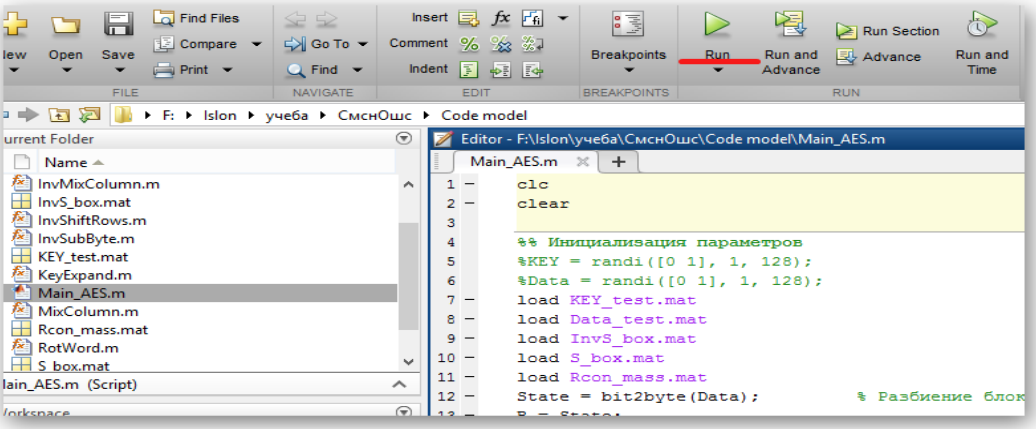


Рисунок П.2 – Запуск кода