А.М. Голиков

ЗАЩИТА ИНФОРМАЦИИ В ЦИФРОВЫХ СИСТЕМАХ СВЯЗИ. КРИПТОГРАФИЧЕСКИЕ ПРОТОКОЛЫ

Учебное пособие

для студентов инженерно-технических специальностей

Курс лекций, компьютерные лабораторные и практические занятия, задание на самостоятельную работу

Томск 2023

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
Томский государственный университет систем управления и радиоэлектроники

А.М. ГОЛИКОВ

ЗАЩИТА ИНФОРМАЦИИ В ЦИФРОВЫХ СИСТЕМАХ СВЯЗИ. КРИПТОГРАФИЧЕСКИЕ ПРОТОКОЛЫ

Учебное пособие
для студентов инженерно-технических специальностей
Курс лекций, компьютерные лабораторные и практические занятия, задание на самостоятельную работу

2023

УДК 621.39(075.8)

ББК 32.973(я73)

Г 60

Автор:

Голиков А.М. — канд. техн. наук, доц. кафедры радиотехнических систем Томского государственного университета систем управления и радиоэлектроники

Рецензент:

Красненко Н.П. — доктор физ-мат. наук, проф. кафедры радиотехнических систем, Томского государственного университета систем управления и радиоэлектроники

Голиков Александр Михайлович

Г60

Защита информации в цифровых системах связи. Криптографические протоколы. Учебное пособие для студентов инженерно-технических специальностей. Курс лекций, компьютерные лабораторные и практические занятия, задание на самостоятельную работу / А.М. Голиков. – Томск: Томск. гос. ун-т систем упр. и радиоэлектроники, 2023. – 144 с.: ил. — (Учебная литература для вузов)

Учебное пособие предназначено для подготовки студентов инженерно-технических специальностей. Представляет собой курс лекций, компьютерные лабораторные и практические занятия, задание на самостоятельную работу. Рассматриваются криптографические протоколы в сетях передачи данных, теория шифров с открытым ключом криптографические протоколы в сетях передачи данных, компьютерный практикум для шифров с открытым ключом и здание на самостоятельную работу по криптографическим протоколам в сетях передачи данных.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Разные люди понимают под шифрованием разные вещи. Дети играют в игрушечные шифры и секретные языки. Это, однако, не имеет ничего общегос настоящей криптографией. Настоящая криптография (strong cryptography) должна обеспечивать такой уровень секретности, чтобы вы имели возможность надежно защитить критическую информацию от расшифровки крупными организациями — такими как мафия, транснациональные корпорации и крупные государства. Настоящая криптография в прошломиспользовалась лишь в военных целях. Однако сейчас, со становлением информационного общества, она становится центральным инструментом для обеспечения конфиденциальности.

По мере образования информационного общества, крупным государствам становятся доступны Технологические средства тотального надзора за миллионами людей. Поэтому криптография становится одним из основных инструментов, обеспечивающих конфиденциальность, доверие, авторизацию, электронные платежи, корпоративную безопасность и бесчисленное множество других важных вещей.

Криптография не является более придумкой военных, с которой не стоит связываться. Настала пора снять с криптографии покровы таинственности и использовать все ее возможности на пользу современному обществу. Широкое распространение криптографии является одним из немногих способов защитить человека от ситуации, когда он вдруг обнаруживает, что живет в тоталитарном государстве, которое может контролировать каждый его шаг.

1. КРИПТОГРАФИЧЕСКИЕ ПРОТОКОЛЫ В СЕТЯХ ПЕРЕДАЧИ ДАННЫХ

1.1 ТЕОРИЯ ШИФРОВ С ОТКРЫТЫМ КЛЮЧОМ. КРИПТОГРАФИЧЕСКИЕ ПРОТОКОЛЫ В СЕТЯХ ПЕРЕДАЧИ ДАННЫХ

1.1.1 БЕЗОПАСНОСТЬ НА ТРАНСПОРТНОМ УРОВНЕ: SSL И TLS

Безопасность транспортного уровня обеспечивает услуги безопасности "из конца в конец" для приложений, которые используют протоколы транспортного уровня, такие как TCP. Основные идеи предназначены для того, чтобы обеспечить услуги безопасности на сети Интернет. Например, когда в сети имеются интерактивно работающие онлайн-магазины, то желательны следующие услуги безопасности:

1. Клиент должен убедиться, что сервер принадлежит фактическому продавцу, а не самозванцу. Клиент не хочет сообщать самозванцу номер кредитной карточки (установление подлинности объекта).

2. Клиент и продавец должны быть убеждены, что содержание сообщения не изменено в течение передачи (целостность сообщения).

3. Клиент и продавец должны быть убеждены, что самозванец не перехватит чувствительную информацию, такую как номер кредитной карточки (конфиденциальность).

Сегодня применяются в основном два протокола обеспечения безопасности на транспортном уровне: Протокол "Уровень безопасных розеток" (SSL - Secure Socket Layer) и Протокол Безопасности Транспортного уровня (TLS - Transport Layer Security).

Протокол SSL разработан компанией Netscape для защиты данных между сервисными и транспортными протоколами. Первая обнародованная версия была выпущена в 1995 году. Широко используется для VoIP-приложений, сервисов обмена мгновенными сообщениями. Протокол SSL использует как симметричный, так и асимметричный ключи.

TLS-протокол основан на спецификации протокола SSL версии 3.0, разработанной компанией Netscape. Сейчас развитием стандарта TLS занимается IETF.

В таблице 1.1 представлены версии сетевых протоколов SSL и TLS.

Таблица 1.1 – Протоколы SSL и TLS

Протокол	Дата публикации	Состояние
SSL 1.0	не публиковался	не публиковался
SSL 2.0	1995	Признан устаревшим в 2011 году
SSL 3.0	1996	Признан устаревшим в 2015 году
TLS 1.0	1999	Признан устаревшим в 2020 году

TLS 1.1	2006	Признан устаревшим в 2020 году
TLS 1.2	2008	
TLS 1.3	2018	

На рисунке 1.1 показано место SSL и TLS в модели интернет TCP/IP:

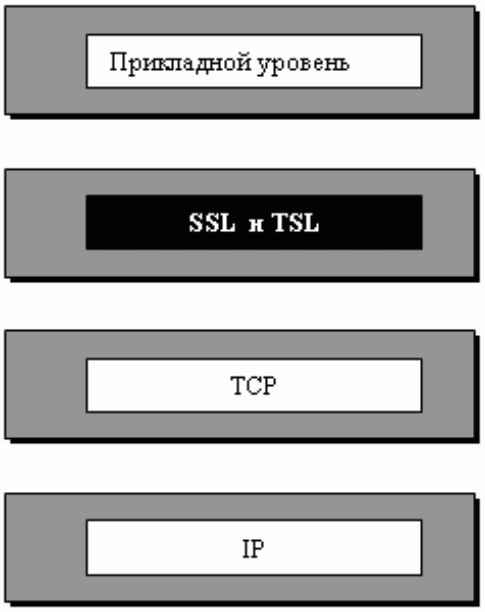


Рисунок 1.1 – Место SSL и TLS в модели Интренет

Одна из целей этих протоколов состоит в том, чтобы обеспечить сервер и клиента услугами установления подлинности, конфиденциальности и целостности данных. Прикладной уровень программ клиент-сервер (client-server), таких как Язык передачи гипертекста (HTTP), который использует услуги TCP, может инкапсулировать свои данные в пакеты SSL.

АРХИТЕКТУРА SSL

SSL разработан, чтобы обеспечить безопасность и услуги сжатия данным, сгенерированным прикладным уровнем. Как правило, SSL может получить данные от любого протокола прикладного уровня, но обычно он получает их от протокола HTTP. Данные, полученные от приложения, сжаты (дополнительно), подписаны и зашифрованы, а затем их передают к протоколу транспортного уровня, такому как TCP.

SSL обеспечивает несколько услуг для данных, полученных от прикладного уровня:

1. Фрагментация. Сначала SSL делит данные на блоки 214 байтов или меньше.

2. Сжатие. Каждый фрагмент данных сжат с использованием одного из методов сжатия без потери методом, согласованным по договору между клиентом и сервером. Эта услуга является дополнительной.

3. Целостность сообщения. Чтобы сохранять целостность данных, SSL использует ключевую хэш-функцию для создания кода проверки подлинности (MAC).

4. Конфиденциальность. Чтобы обеспечить конфиденциальность, первоначальные данные и код проверки подлинности (MAC) зашифрованы, с использованием криптографии с симметричными ключами.

5. Организация кадра. К зашифрованной полезной нагрузке добавляется заголовок. Полезную нагрузку затем передают достоверному протоколу транспортного уровня.

Алгоритмы смены ключа

Для обмена подлинными и конфиденциальными сообщениями клиенту и серверу нужны шесть криптографических объектов секретности (четыре ключа и два вектора инициализации). Однако, чтобы создать их, между этими двумя сторонами должен быть установлен один предварительный главный секретный код (pre-master secret). SSL определяет шесть методов обмена ключами, чтобы установить этот предварительный объект секретности: NULL, RSA, анонимный Диффи-Хеллман (Diffie-Hellman), кратковременный Диффи-Хеллман, фиксированный Диффи-Хеллман и Fortezza (рисунок 1.2).

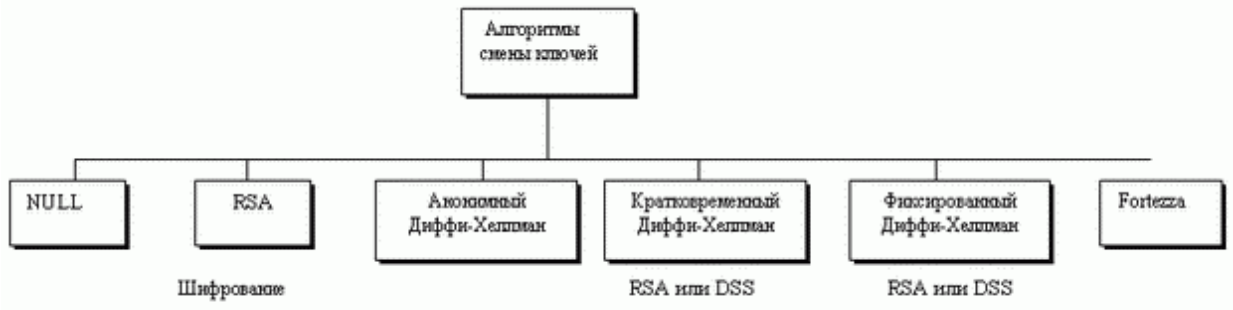


Рисунок 1.2 – Метод замены ключей

NULL (пустой указатель). В этом методе нет никакой смены ключей. Между клиентом и сервером не установлен предварительный главный секретный код. И клиент, и сервер должны знать значение предварительного главного секретного кода.

RSA. В этом методе предварительный главный секретный код - 48-байтовое случайное число, созданное клиентом, зашифрованное открытым ключом RSA-сервера и передаваемое серверу. Сервер должен передать свой сертификат шифрования/дешифрования RSA. Рисунок 1.2 иллюстрирует идею:

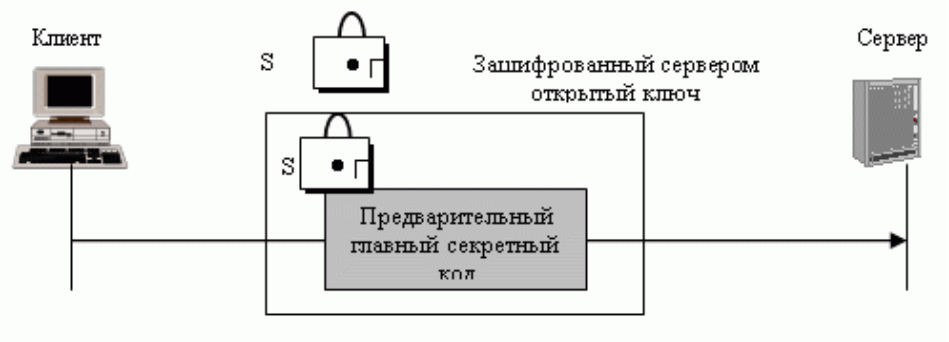


Рисунок 1.3 – RSA-смена ключа; открытый ключ сервера

Анонимный протокол Диффи-Хеллмана. Это самый простой и наиболее ненадежный метод. Предварительный главный секретный код устанавливают между клиентом и сервером, используя протокол Диффи-Хеллмана. При этом передают половину ключа в исходном тексте — это называется анонимным протоколом Диффи-Хеллмана, потому что ни одна сторона не известна другой. Как мы уже обсуждали, самый серьезный недостаток этого метода - возможность атаки "посредника". Рисунок 1.4 иллюстрирует идею анонимного метода.

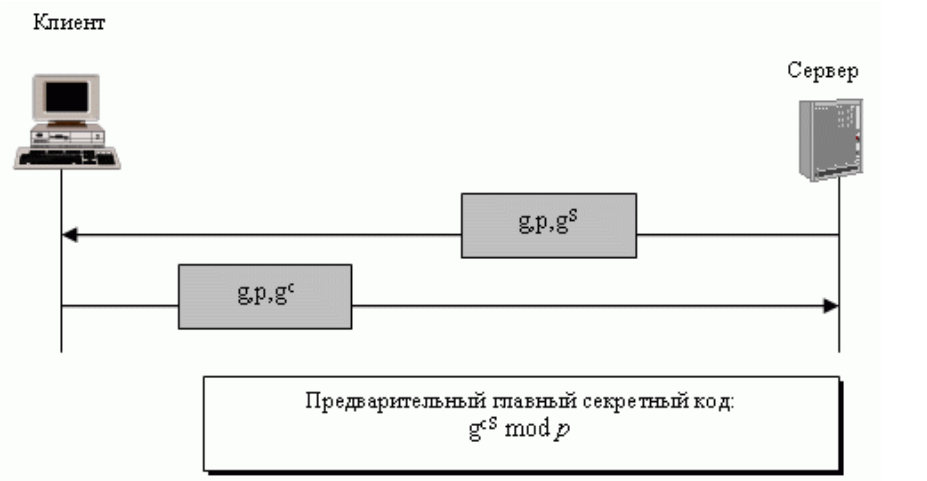


Рисунок 1.4 – Анонимный протокол Диффи-Хеллмана смены ключа

Кратковременный метод Диффи-Хеллмана. Чтобы сорвать атаку "посредника", может быть использована кратковременная смена ключей методом Диффи-Хеллмана. Каждая сторона передает ключ Диффи-Хеллмана, подписанный своим секретным ключом. На приемной стороне должны проверить подпись, используя открытый ключ передатчика. Обмен открытыми ключами для проверки использует либо RSA-, либо DSS-сертификат цифровой подписи. Рисунок 1.5 иллюстрирует идею.

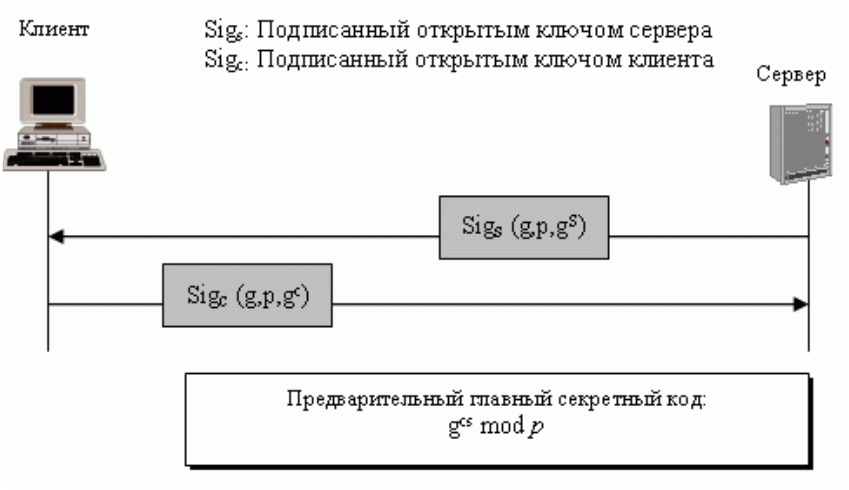


Рисунок 1.5 – Кратковременный протокол Диффи-Хеллмана смены ключей

Фиксированный метод Диффи-Хеллмана. Другое решение - фиксированный метод Диффи-Хеллмана. Все объекты в группе могут подготовить фиксированные параметры (g и p). Затем каждый объект может создать фиксированную половину ключа (gx). Для дополнительной безопасности каждая отдельная половина ключа Диффи-Хеллмана вставляется в сертификат, проверенный центром сертификации (CA). Другими словами, две стороны отдельно не обмениваются полуключами; CA передает полуключи в специальном сертификате RSA или DSS. Когда клиент должен вычислить предварительный главный секретный код, он использует свой собственный фиксированный полуключ и полуключ сервера, полученный в сертификате. Сервер делает то же самое, но в обратном порядке. В этом методе не передаются сообщения смены ключей, а происходит только обмен сертификатами.

Fortezza. Fortezza (образован из итальянского слова "крепость") - зарегистрированная торговая марка американского Агентства Национальной безопасности (NSA - National Security Agency). Это семейство протоколов безопасности, разработанных для Отдела Защиты.

Алгоритмы шифрования/дешифрования

Алгоритма шифрования/дешифрования можно разделить на 6 групп (рисунок 1.6):

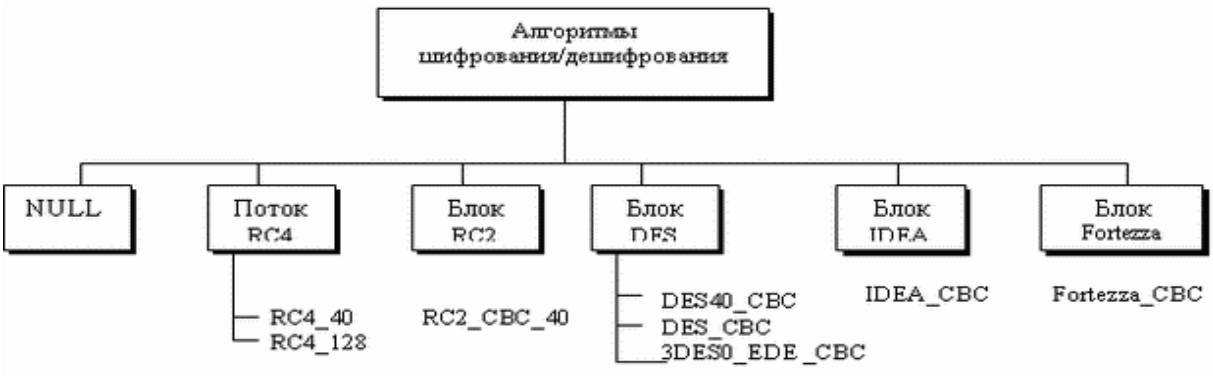


Рисунок 1.6 – Алгоритмы шифрования/дешифрования

NULL - категория, которая просто определяет отсутствие алгоритма шифрации/дешифрации.

Поток RC. В режиме потока RC определены два потока алгоритма: RC4-40 (ключ на 40 битов) и RC4-128 (ключ на 128 битов).

Блок RS. В режиме блока RC определен один алгоритм: RC2_CBC_40 (ключ на 40 битов).

CBC (Cipher Block Chaining) - сцепление шифрованных блоков.

DES. Все алгоритмы DES определены в режиме блока. DES40_CBC использует ключ на 40 битов. Стандартные DES определены как DES_CBC. 3DES_EDE_CBC используют ключ на 168 битов.

IDEA. В режиме блока IDEA определен один алгоритм - IDEA_CBC, с ключом на 128 битов.

Fortezza. В режиме блока Fortezza определен один алгоритм - FORTEZZA_CBC, с ключом на 96 бит.

Алгоритмы хеширования

SSL использует алгоритмы хэширования, чтобы обеспечить целостность сообщения (установление подлинности сообщения). Имеются хэш-функции, показанные на рис. 1.7.

Рисунок 1.7 – Алгоритмы хэширования

Null (Пустой указатель). Две стороны могут отказаться использовать алгоритм хэширования. В этом случае сообщение не заверено.

MD5.Две стороны могут выбрать MD5 как алгоритм хэширования. В этом случае используется алгоритм хэширования MD5 - 128-битовый.

SHA-1.Две стороны могут выбрать SHA как алгоритм хэширования. В этом случае используется алгоритм хэширования SHA-1 на 160 битов.

Набор шифров

Комбинация смены ключей, хэширования и алгоритмов шифрования определяет набор шифров для каждого сеанса SSL. В таблице 1.1 показаны наборы, применяемые в Соединенных штатах.

Таблица 1.1 – Список набора шифров SSL

Набор шифров	**Смена ключей**	**Шифрование**	**Хэш**
SSL-NULL-WITH-NULL- NULL	NULL	NULL	NULL
SSL_RSA_WITH_NULL_MD5	RSA	NULL	MD5
SSL_RSA_WITH_NULL_SHA	RSA	NULL	SHA-1
SSL_RSA_WITH_RC4_128_MD5	RSA	RC4	MD5
SSL_RSA_WITH_RC4_128_SHA	RSA	RC4	SHA-1 SHA-1
SSL_RSA_WITH_IDEA_CBC_SHA	RSA	IDEA	SHA-1
SSL_RSA_WITH_DES_CBC_SHA	RSA	DES	SHA-1
SSL RSA WITH 3DES EDE CBC SHA	RSA	3DES	
SSL_ DH_anon_ WITH_ RC4_128_ MD5	DH anon	RC4	MD5
SSL_ DH_ anon WITH_ DES_ CBC_ SHA	DH anon	DES	SHA-1
SSL _DH_ anon_ WITH_ 3DES_ EDE_ CBC SHA	DH anon	3DES	SHA-1
SSL_ DHE_ RSA_ WITH_ DES_ CBC_ SHA	DHE RSA	DES	SHA-1
SSL_ DHE_RSA_ WITH_3DES_ EDE CBC SHA	DHE RSA	3DES	SHA-1
SSL_DHE_DSS_ WITH_ DES_ CBC_ SHA	DHE DSS	DES	SHA-1
SSL_ DHE_ DSS_ WITH_ 3DES_ EDE_ CBC_ SHA	DHE DSS	3DES	SHA-1
SSL_ DH_ RSA_ WITH_ DES_ CBC_ SHA	DH RSA	DES	SHA-1
SSL_ DH_ RSA_ WITH_ 3DES_ EDE_ CBC_ SHA	DH RSA	3DES	SHA-1
SSL_ DH_ DSS_ WITH_ DES_ CBC_ SHA	DH DSS	DES	SHA-1

SSL_ DH_ DSS_ WITH_ 3DES_ EDE_ CBC_ SHA	DH DSS	3DES	SHA-1
SSL_FORTEZZA_DMS_WITH_NULL_SHA SSL_FORTEZZA_DMS_WITH_FORTEZZA_CBC_SHA SSL_FORTEZZA_DMS_WITH_RC4_128_SHA	Fortezza Fortezza Fortezza	NULL Fortezza RC4	SHA-1 SHA-1 SHA-1

Генерирование криптографических параметров

Чтобы обеспечить целостность и конфиденциальность сообщения, в SSL необходимо иметь: шесть криптографических объектов секретности, четыре ключа и два инициализирующих вектора (IV). Клиенту нужно: один ключ для передачи сообщения

установления подлинности, один ключ для шифрования и один IV для шифрования блока. Сервер нуждается в том же самом. SSL требует, чтобы ключи для одного направления отличались от ключей для другого направления. Если будет атака в одном направлении, она не затронет другое направление. Для генерации параметров используют следующую процедуру:

1. Клиент и сервер обмениваются двумя случайными числами, одно из которых создано клиентом, а другое - сервером.

2. Клиент и сервер обмениваются одним предварительным главным секретным кодом, используя один из алгоритмов смены ключей, которые мы обсуждали раньше.

3. Создается 48-байтовый главный секретный код (master secret) из предварительного главного секретного кода (pre-master secret), с применением хэш-функций (SHA-1 и MD5), как это показано на рисунке 1.8.

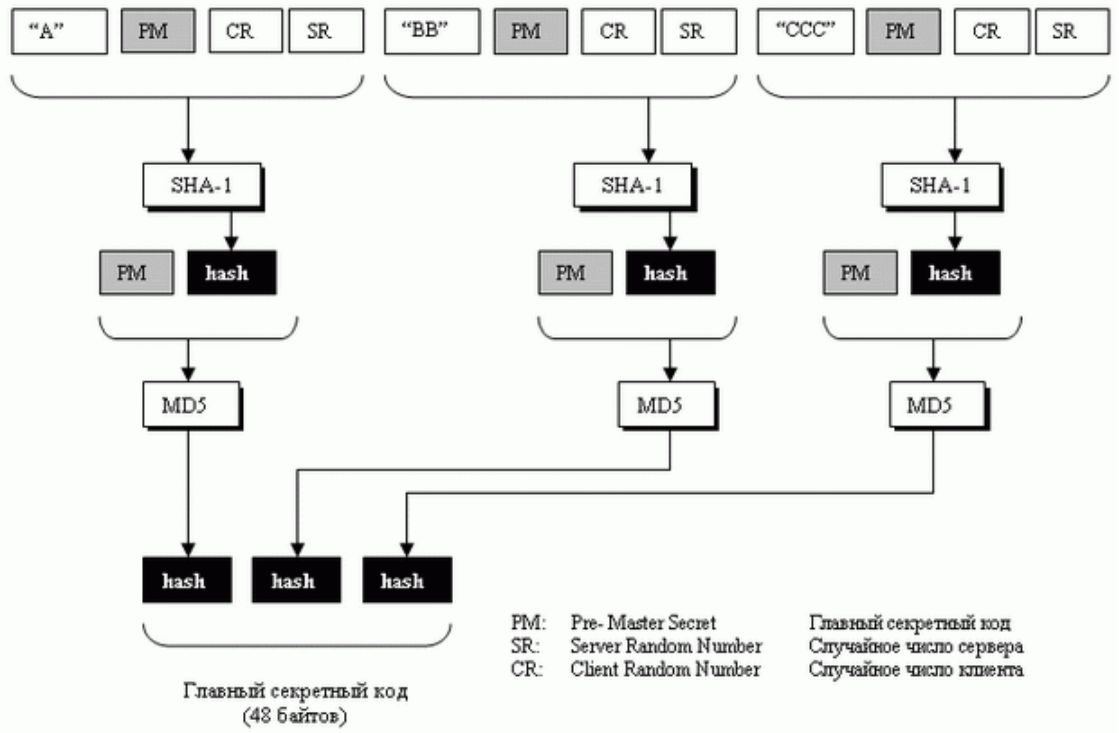


Рисунок 1.8 – Вычисление главного секретного кода из предварительного главного секретного кода

4. Главный секретный код используется для того, чтобы создать материал для ключей (key material), который имеет переменную длину. Для этого применяют то же самое множество хэш-функций, что и в предыдущем случае, и подставляют спереди различные константы, как это показано на рисунке 1.9. Алгоритм повторяется, пока не получится материал для ключа адекватного размера.

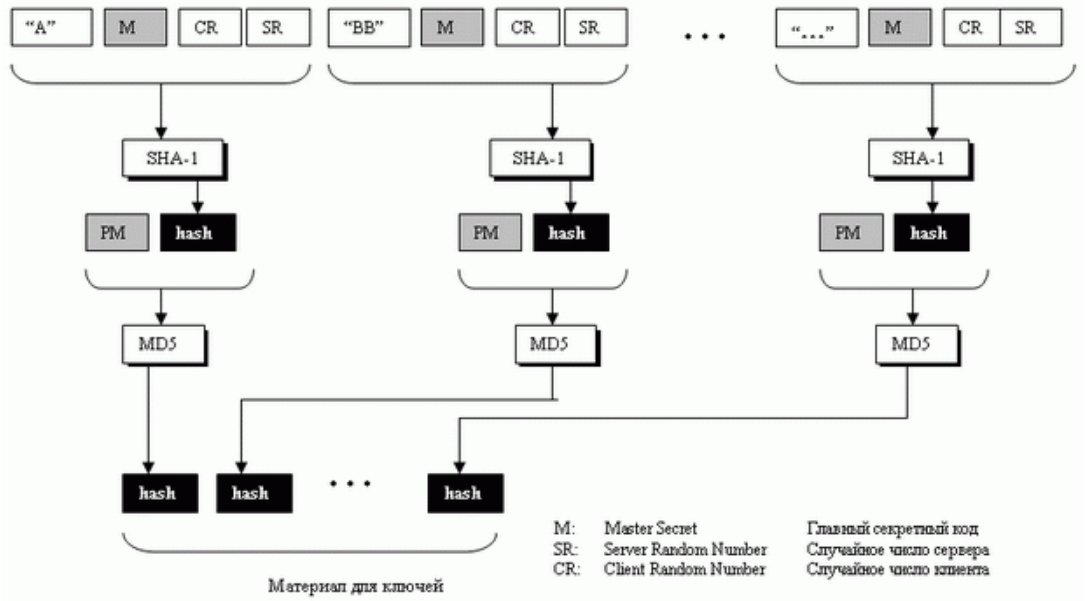


Рисунок 1.9 – Вычисление материала для ключей из главного секретного кода

5. Из материала для ключей извлекаются шесть различных ключей, как показано на рис. 1.10.

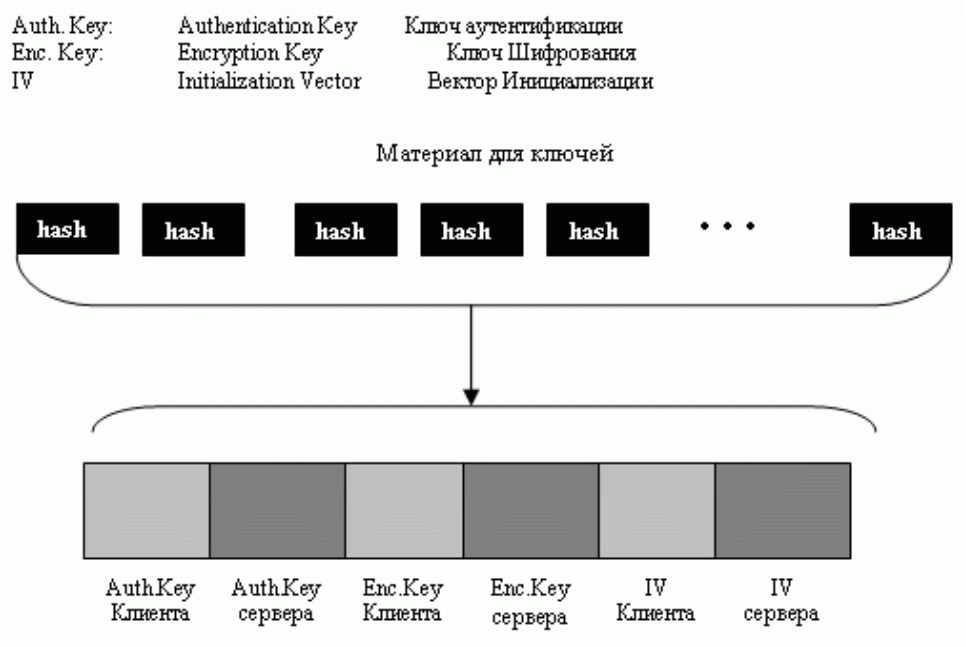


Рисунок 1.10 – Извлечение криптографических секретных кодов из материала

Сеансы и соединения

SSL отличает соединение от сеанса. Сеанс - связь между клиентом и сервером. Соединения - это множество параметров, установленных между двумя равными по уровню объектами.

В таблицах 1.2 и 1.3 представлены параметры SSL.

Таблицы 1.2 – Параметры состояния сеанса

Параметры	Описание
ID сеанса	Случайное 8-битовое число, выбранное сервером и определяющее сеанс
Сертификат уровня	Сертификат типа X509 .v.3. Этот параметр может быть пустым (null)
Метод сжатия	Метод сжатия
Набор шифров	Согласованный набор шифров
Главный секретный код	48-байтовый секретный код
Возможность повторения	Флаг "Да, Нет", который разрешает новое соединение в старом сеансе

Таблица 1.3 – Параметры состояния соединения.

Параметры	Описание
Случайные числа клиента и сервера	Последовательность байтов, выбранная для каждого соединения серверу и клиенту.
Записанный сервером секретный код подлинности сообщения	Ключ кода установления подлинности сообщения исходящего сервера для сохранения целостности сообщения.. Используется сервером для подписи, а клиентом для верификации.
Записанный клиентом секретный код установления подлинности сообщения	Ключ кода установления подлинности сообщения исходящего сервера для сохранения целостности сообщения.. Используется сервером для подписи, а клиентом для верификации.
Секретный код, записанный сервером	Ключ шифрования исходящего сервера для сохранения целостности сообщения
Секретный код, записанный клиентом	Ключ шифрования исходящего сервера для сохранения целостности сообщения
Вектор инициализации	Блочные шифры в режиме "цепочки блочных шифров" I (CBC) используют векторы инициализации. (IV). Для каждого шифровального ключа путем переговоров определен один вектор инициализации, который используется первым блоком обмена ключами. Зашифрованный текст из блока используется как вектор инициализации (IV) для следующего блока.
Порядковый номер	Каждая сторон имеет порядковый номер. Он начинается с 0 и увеличивается на 1. Он не должен быть больше 264 - 1.

Четыре протокола

SSL содержит четыре протокола на двух уровнях, как это изображено на рис. 1.11. Протокол передачи записей - переносящий информацию. Он переносит на транспортный уровень сообщения от трех других протоколов, а также данные, поступающие от прикладного уровня. Сообщения из протокола записей - это полезная нагрузка для транспортного уровня, обычно TCP. Протокол установления соединения обеспечивает параметры безопасности для

Протокола записей. Он устанавливает набор шифров и задает ключи и параметры безопасности.

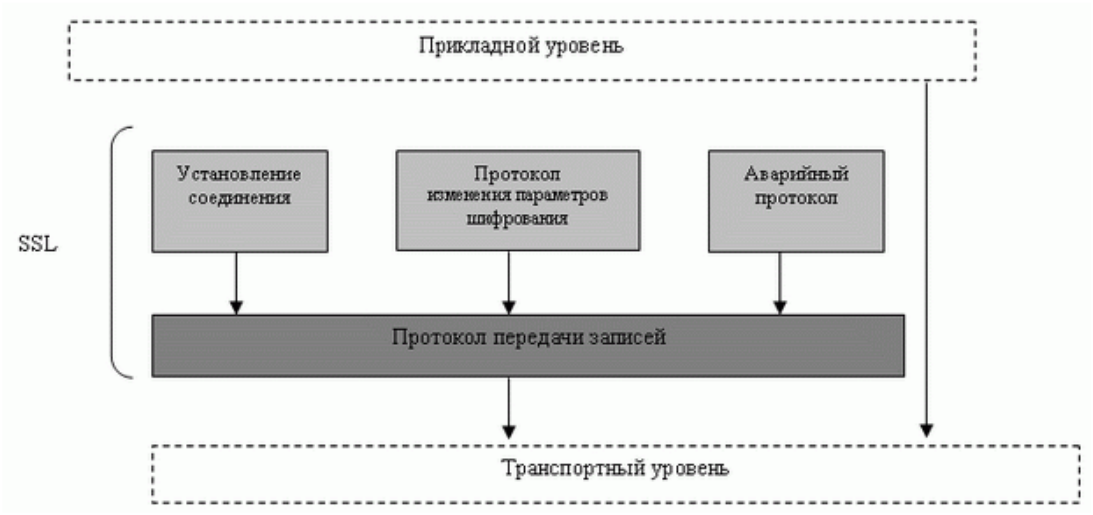


Рисунок 1.11 – Четыре протокола SSL

Протокол установления соединения используется при передаче сообщений, чтобы договориться, если это необходимо, о составе шифров от сервера к клиенту и от клиента к серверу и обменяться информацией, для того чтобы обеспечить криптографическую безопасность. Процедура установления связи происходит в 4 фазы; она проиллюстрирована на рис. 1.12.

Рисунок 1.12 – Протокол установления соединения

В Фазе I клиент и сервер объявляют свои характеристики безопасности, которые нужны и удобны для обоих. В этой фазе устанавливается и выбирается ID сеанса. Стороны согласуют конкретный метод сжатия.

В фазе II сервер, если необходимо, подтверждает свою подлинность. Передатчик может передать свой открытый ключ и может также запросить сертификат клиента.

Фаза III предназначена подтвердить подлинность клиента. От клиента серверу можно передать до трех сообщений: цепочка сертификатов, открытый ключ клиента, хэш-код для доказательства сертификации.

В Фазе IV клиент и сервер передают сообщения, чтобы изменить спецификацию шифра и закончить процедуру установления связи.

SSL использует аварийный протокол для того, чтобы известить об ошибках и ненормальном состоянии устройств. Имеется только один тип аварийного сообщения, которое описывает проблему и ее уровень (опасное или полный выход из строя). табл. 2.4 показывает типы аварийных сообщений, определенных для SSL.

Таблица 1.4 – Аварийные сообщения.

Цифровое обозначение	Описание	Значение
0	CloseNotify	Передатчик больше не будет посылать сообщений
10	UnexpectedMessage	Получено несоответствующее сообщение
20	BadRecordMAC	Получено некорректное MAC
30	DecompressionFailure	Невозможно получить несжатый текст
40	HandshakeFailure	Передатчик не может закончить установление соединения
41	NoCertificate	Клиент не сертифицировал послание
42	BadCertificate	Полученный сертификат искажен
43	UnsupportedCertificate	Тип полученного сертификата не поддерживается
44	CertificateRevoked	Подписавший аннулирует сертификат
45	CertificateExpired	Сертификат просрочен
46	CertificateUnknown	Сертификат неизвестен
47	Illegal Parameter	Поле не может быть обработано

ChangeCipherSpec - сообщение, которым обмениваются клиент и сервер в ходе выполнения протокола установления соединения и которое определено в протоколе

ChangeCipherSpec. По этой причине объекты не посылают или не получают сообщение. Передатчик и приемник нуждаются не в одном, а в двух состояниях. Одно - состояние ожидания, при котором сохраняются параметры и секретности. Другое - активное состояние, при котором параметры и секретность используются протоколом передачи записей, чтобы подписаться/проверить или зашифровать/расшифровывать сообщения.

Протокол передачи записей доставляет сообщения от верхнего уровня (протокол установления соединения, ChangeCipherSpec-протокол, аварийный протокол) или прикладных уровней. Сообщения фрагментированы и произвольно сжаты; MAC добавляется к сжатому сообщению, используя согласованный алгоритм хэш. Сжатый фрагмент и MAC зашифрованы с применением согласованного алгоритма шифрования. В конце к зашифрованному сообщению добавляется заголовок SSL.

Форматы сообщений

Сообщение в протокол передачи записей инкапсулируется из четырех различных источников на стороне передатчика. На стороне приемника протокол передачи записей извлекает сообщения и доставляет их различным пунктам назначения. Протокол передачи записей имеет общий заголовок, который добавляется к каждому сообщению, прибывающему от источников, как показано на рис. 1.13.

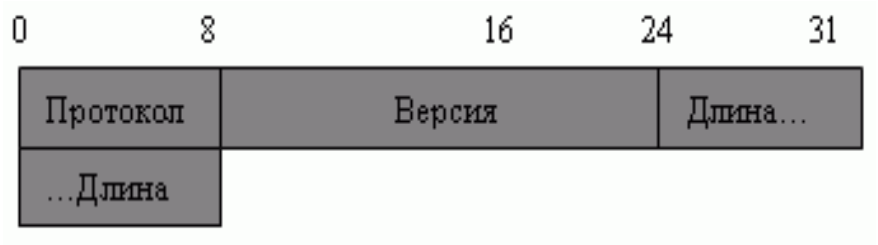


Рисунок 1.13 – Общий заголовок протокола передачи записей

Аварийный протокол, имеет одно сообщение - рапорт об ошибках в процессе (рис. 1.14).

Рисунок 1.14 – Аварийное сообщение

Сообщение ClientHello - первое сообщение в обмене при установлении соединения (рис. 1.15).

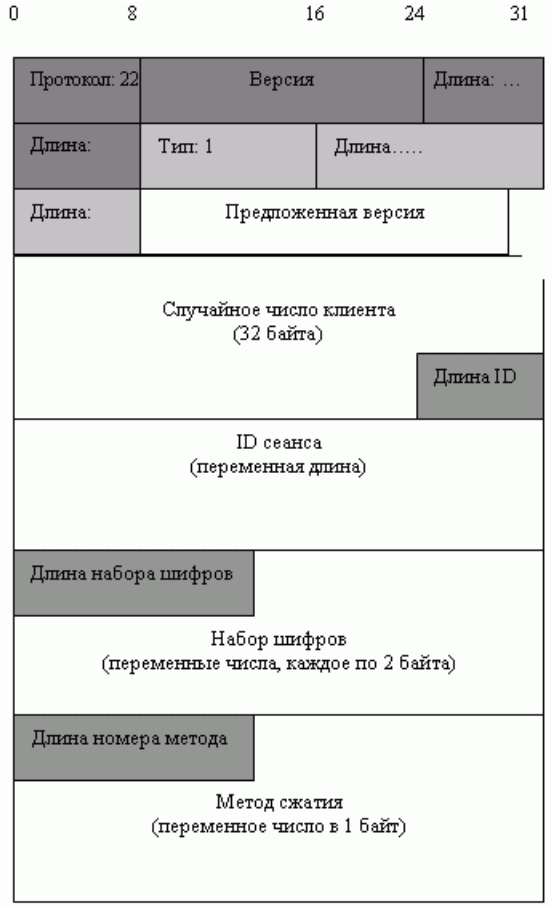


Рисунок 1.15 – Сообщение ClientHello

Сообщение ServerHello - ответ сервера на сообщение ClientHello (рис. 1.16).

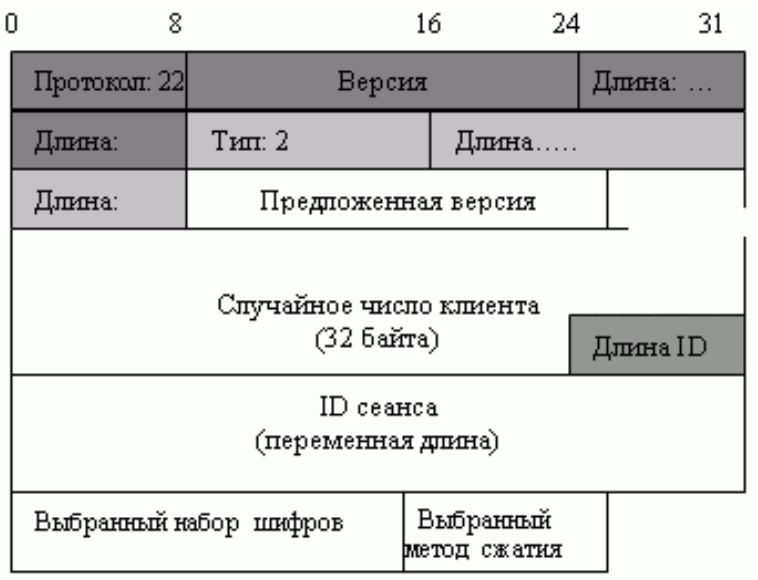


Рисунок 1.16 – Сообщение ServerHello

Безопасность транспортного уровня: TLS

Безопасность транспортного уровня (TLS - Transport Layer Security) - протокол IEFT, стандартная версия протокола SSL. Эти два протокола очень похожи, но имеют небольшие отличия.

Незначительное отличие между SSL и TLS - отсутствие поддержки Fortezza. TLS не поддерживает Fortezza для смены ключей или для шифрования/дешифрования. табл.1.5 показывает набор шифров для TLS.

Таблица 1.5 – Набор шифров TLS

Набор шифров	**Замена ключей**	**Шифрование**	**Хэш**
TLS_NULL_WITH_NULL_NULL	NULL	NULL	NULL
TLS_RSA_WITH_NULL_MD5	RSA	NULL	MD5
TLS_RSA_WITH_NULL_SHA	RSA	NULL	SHA-1
TLS_RSA_W1TH_RC4_128_MD5	RSA	RC4	MD5
TLS_RSA_WITH_RC4_128_SHA	RSA	RC4	SHA-!
TLS_RSA_WITH_IDEA_CBC_SHA	RSA	IDEA	SHA-1
TLS_RSA_WITH_DES_CBC_SHA	RSA	DES	SHA-1
TLS_RSA_WITH_3DES_EDE_CBC_SHA	RSA	3DES	SHA-1
TLS_DH_anon_WITH-RC4_l 28_MD5	DH _anon	RC4	MD5
TLS_DH_anon_WITH_DES_CBC_SHA	DH_anon	DES	SHA-1
TLS_DH_anon_WITH_3DES_EDE_CBC_SHA	DH_anon	3DES	SHA-1
TLS_ DHE_ RSA_ WITH_ DES_ CBC_ SHA	DHE_RSA	DES	SHA-1
TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA	DHE_RSA	3DES	SHA-1
TLS_ DHE_ DSS_ WITH_ DES_ CBC_ SHA	DHE_.DSS	DES	SHA-1
TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA	DHE_DSS	3DES	SHA-1

Генерация криптографической секретности в TLS более сложная, чем в SSL. TLS сначала определяет две функции: функцию расширения данных и псевдослучайную функцию.

Функция расширения данных использует заранее заданный код аутентификации на основе хэширования (HMAC-HASH-BASED MESSAGE AUTHENTICATION CODE), или MD5, или SHA-1 для того, чтобы расширить информацию засекречивания. Эту функцию можно рассматривать как функцию, содержащую множество секций, где каждая секция создает одно значение хэширования. Расширенная секретность - последовательное соединение значений

хэширования. Каждая секция использует два HMAC, информацию засекречивания и начальное число. Функция расширения данных - это формирование цепочки в виде многих секций (рис. 1.17).

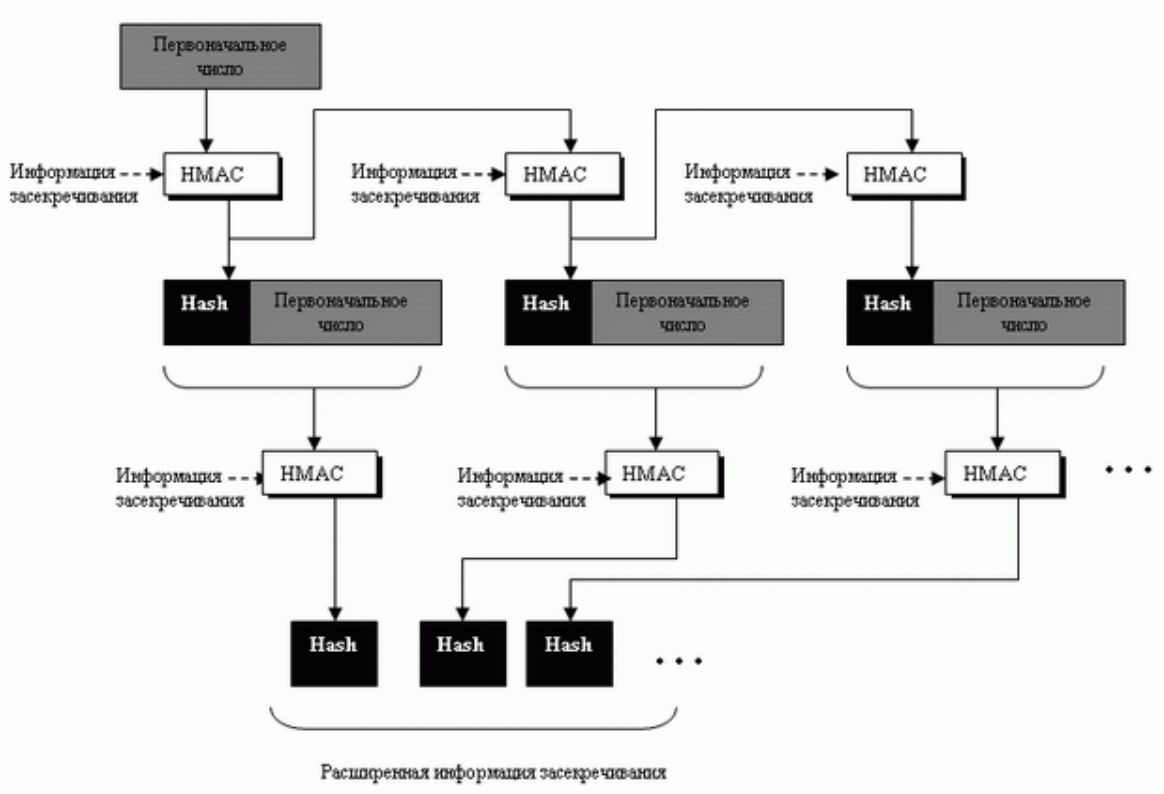


Рисунок 1.17 – Функция расширения данных

TLS определяет псевдослучайную функцию (PRF - PseudoRandom Function), чтобы получить комбинацию двух функций расширения данных: одна из них использует MD5 и другая - SHA-1. На PRF поступает три части информации: секретный код, метка и начальное число.

Метка и начальное число связаны и служат начальным числом для каждой функции расширения данных. Информация засекречивания разделена на две части; каждая часть используется как информация засекречивания для каждой функции расширения данных. Выходы двух функций расширения данных складывают по модулю два, чтобы создать конечную расширенную информацию засекречивания. Обратите внимание, что поскольку хэш создается MD5 и SHA-1, он имеет различные размеры, поэтому должны быть созданы дополнительные секции функций на базе MD5, чтобы сделать два вывода с одинаковым размером. рисунок. 1.18 показывает идею применения PRF.

Рисунок 1.18 – PRF

Рисунок 1.19 – Генерация главного секретного кода

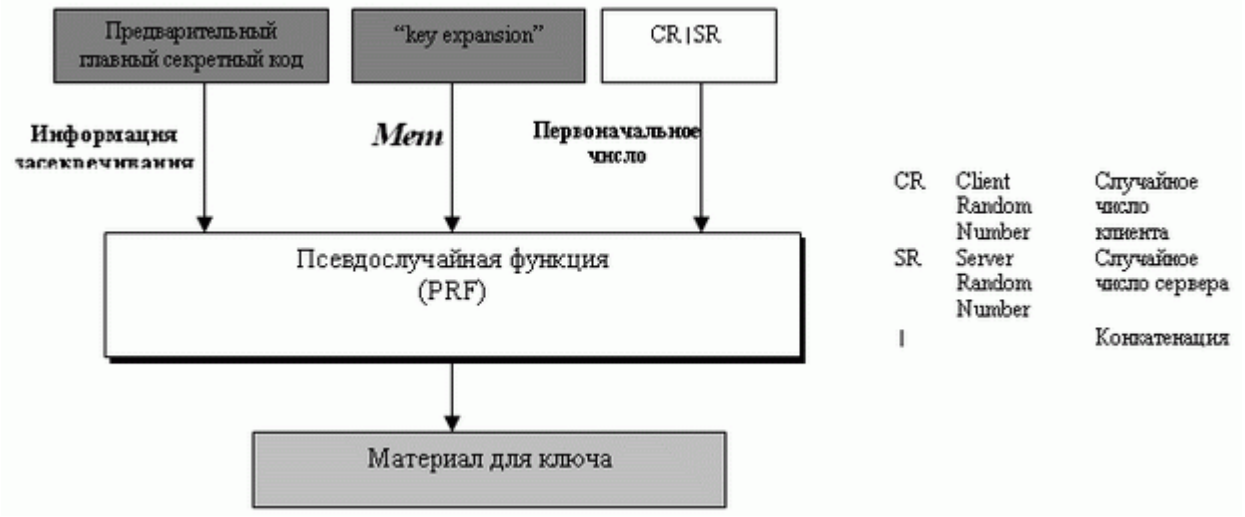


Рисунок 1.20 – Генерация материала для ключа

TLS поддерживает все аварийные сигналы, определенные в SSL, за исключением NoCertificate. TLS также добавляет к списку SSL некоторые новые.

TLS вносит некоторые изменения в протокол установления соединения. Были специально изменены детали сообщения CertificateVerify и сообщения Finished.

Единственное изменение в протоколе передачи записей - использование HMAC, чтобы подписать сообщение (рис. 1.21).

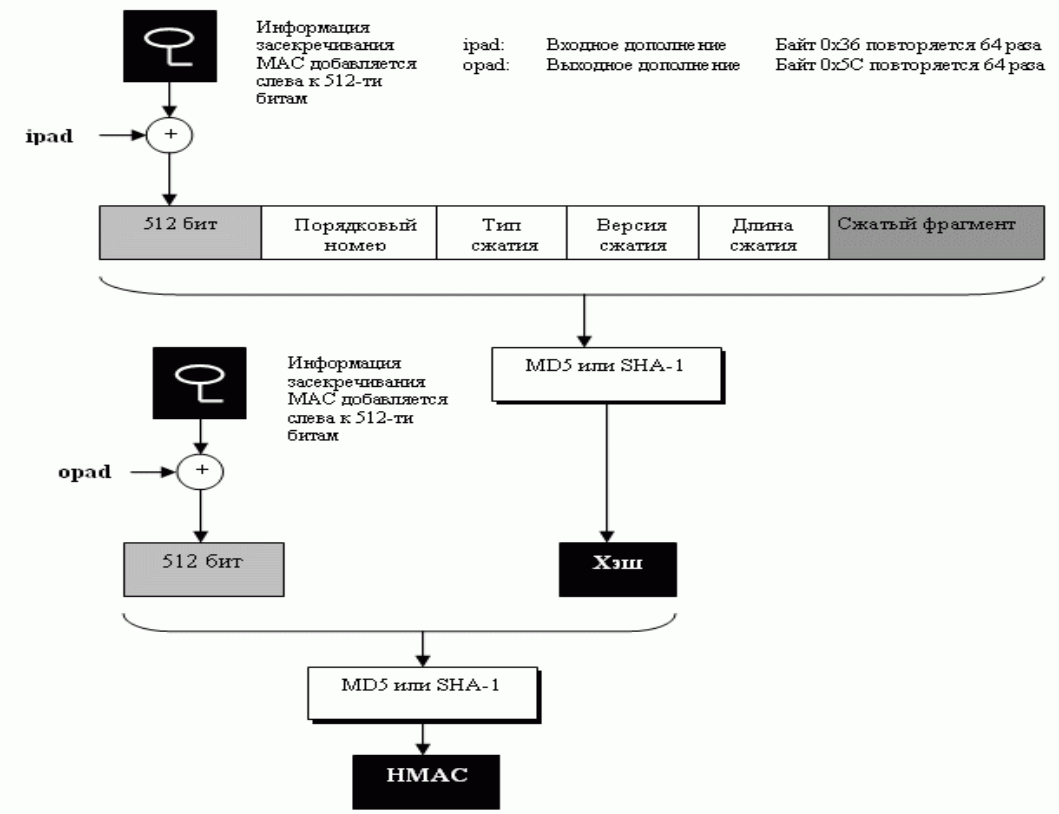


Рисунок 1.21 – HMAC для TSL

Таким образом, протокол безопасности транспортного уровня обеспечивает услуги безопасности из конца в конец для приложений, которые пользуются протоколами транспортного уровня, такими как, например, TCP. На сегодняшний день преобладает применение двух протоколов: протокол "Уровень Безопасных Розеток" (SSL - Secure Sockets Layer) и протокол "Безопасность Транспортного уровня" (TLS - Transport Layer Security).

SSL или TLS обеспечивают такие услуги, как фрагментация, сжатие, целостность сообщения, конфиденциальность и создание кадра данных, полученных от прикладного уровня. Как правило, SSL (или TLS) может получить прикладные данные от любого протокола прикладного уровня, но работает протокол обычно с HTTP.

Комбинация алгоритмов смены ключей, хэширования и алгоритм шифрования определяют набор шифров для каждого сеанса.

Для того чтобы обмениваться заверенными и конфиденциальными сообщениями, клиенту и серверу необходимо иметь шесть единиц криптографической секретности (четыре ключа и два вектора инициализации).

В SSL (или TLS) отличают подключение и сеанс. В сеансе одна сторона играет роль клиента, а другая - роль сервера; при подключении обе стороны играют одинаковые роли, на равном подуровне.

SSL (или TLS) определяет четыре протокола на двух уровнях: протокол установления соединения, протокол ChangeCipherSpec, аварийный протокол и протокол передачи записей. Протокол установления соединения использует несколько сообщений, чтобы договориться о наборе шифров, подтвердить подлинность сервера для клиента и клиента для сервера, если это необходимо, и обмениваться информацией для организации криптографической секретности. Протокол ChangeCipherSpec определяет процесс перемещения информации между состоянием ожидания и активным состоянием. Аварийный протокол передает извещения об ошибках и ситуациях, отклоняющихся от нормальных. Протокол передачи записей доставляет сообщения от верхнего уровня (протокол установления соединения, аварийный протокол, ChangeCipherSpec-протокол) или прикладного уровня.

Контрольные вопросы

1. Перечислите услуги, обеспеченные SSL или TLS.
2. Объясните, как в SSL создается из *предварительного главного секретного кода главный секретный код*.
3. Объясните, как в TLS создается из *предварительного главного секретного кода главный секретный код*.
4. Объясните, как в SSL создается из *главного секретного кода материал для ключей*.
5. Объясните, как в TLS создается из *главного секретного кода материал для ключей*.
6. Покажите различия между сеансом и соединением. Перечислите цель четырех протоколов, определенных в SSL или TLS.
7. Определите цель каждой фазы в протоколе установления соединения.
8. Сравните и противопоставьте протоколы установления соединения в SSL и TLS.
9. Сравните и противопоставьте *протоколы передачи записей* в SSL и TLS.
10. Каков риск использования ключей короткой длины в SSL или TLS? Какую атаку злоумышленник может применить, если ключи коротки?
11. Действительно ли SSL или TLS относительно безопасны к атаке "посредника"? Может ли злоумышленник создать *материал для ключей* между клиентом и самим собой и между собой и сервером?

1.1.2 БЕЗОПАСНОСТЬ НА СЕТЕВОМ УРОВНЕ: IPSEC

Вопрос безопасности всегда стоял перед компьютерными сетями, но сегодня как никогда растет осознание того, насколько важна безопасность компьютерных сетей в корпоративных инфраструктурах. В настоящее время для каждой корпоративной сети необходимо иметь четкую политику в областибезопасности. Эта политика разрабатывается на основе анализа рисков, определения критически важных ресурсов и возможных угроз.

IPSec – это набор правил или протоколов связи для настройки безопасных подключений по сети. Интернет-протокол (IP) – это общепринятый стандарт, определяющий то, как данные передаются поИнтернету. IPSec добавляет шифрование и аутентификацию, чтобы сделатьпротокол более безопасным. Например, он зашифровывает данные в источнике и расшифровывает их в пункте назначения. Также он проверяет подлинность источника данных. Средства безопасности для IP описываютсясемейством спецификаций IPSec, разработанных рабочей группой IP Security. Протоколы IPSec обеспечивают управление доступом, целостность вне соединения, аутентификацию источника данных, защиту от воспроизведения,конфиденциальность и частичную защиту от анализа трафика. Основополагающими понятиями IPSec являются:

- аутентификационный заголовок (AH);
- безопасное сокрытие данных (ESP);
- режимы работы: туннельный и транспортный;
- контексты (ассоциации) безопасности (SA);
- управление ключами (IKE).

Безопасный IP (IPSec) – совокупность протоколов, разработанных Группой Инженерной Поддержки сети Интернет (IETF Internet Engineering Task Force), чтобы обеспечить безопасность передачи пакетов на сетевом уровне. Сетевой уровень в Интернете упоминается часто как Интернет- протокол - Internet Protocol (IP). Протокол IPSec помогает создавать заверенные и конфиденциальные пакеты для уровня IP, как это показано на рисунке 1.22.

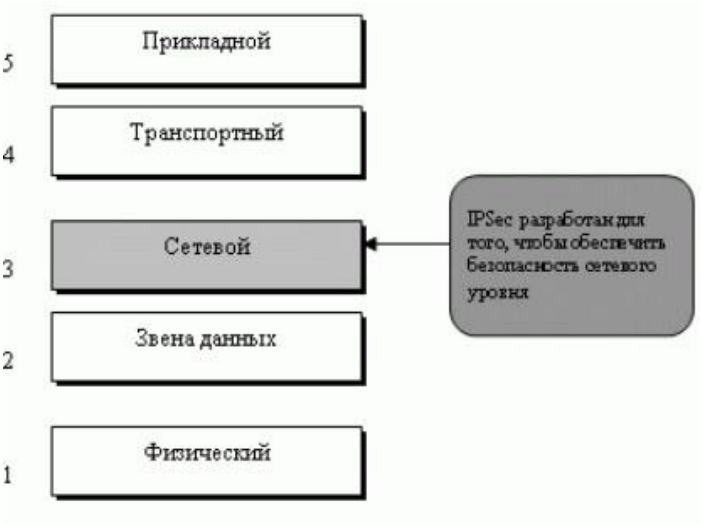


Рисунок 1.22 – Набор протоколов TCP/IP и IPSec

Рисунок 1.23 – Архитектура IPSec

IPSec может быть полезен в нескольких областях. Во-первых, он может увеличить безопасность программ "клиент-сервер", таких как электронная почта, которая использует свои собственные протоколы безопасности. Во- вторых, он может увеличить безопасность программ "клиент-сервер", которыеприменяют службы безопасности на транспортном уровне, - например, HTTP.Он может обеспечить безопасность программ "клиент-сервер", которые не пользуются службами безопасности транспортного уровня. Он может обеспечить безопасность для программ установления связи "от-узла-к-узлу", таких как маршрутизация.

Преимущества: Безопасность и надёжность защиты данных протокола проверена и доказана, так как протокол был принят как Интернет-стандарт; Работа в верхнем слое сетевого протокола и шифрование данных надуровнем сетевого протокола.

Недостатки: Сложность реализации; Дополнительные требования к оборудованию сети (маршрутизаторы и т. п.);

Существует много различных реализаций, не всегда корректно взаимодействующих друг с другом.

Два режима работы IPSec

IPSec работает в двух различных режимах - транспортном и туннельном.

Транспортный режим

В транспортном режиме IPSec защищает информацию, доставляемую оттранспортного уровня к сетевому уровню. Другими словами, транспортный режим защищает полезную нагрузку сетевого уровня, и полезная нагрузка должна быть инкапсулирована в сетевой уровень, как это показано на рисунке 1.24.

Рисунок 1.24 – Транспортный метод отличается от туннелирования много чем, но самоеважное – это метод инкапсуляции

Пакет IPSec в транспортном режиме показан на рисунке 1.25.

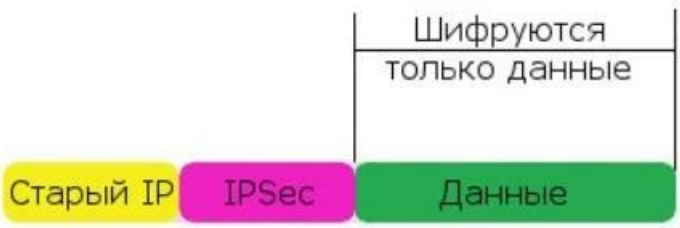


Рисунок 1.25 - Транспортный режим актуален тогда, когда IP-связность ужедостигнута, но трафик между узлами нужно шифровать

Удачным примером применения транспортного режима может бытьсхема сервер-клиент. Например, работа клиент-банка. Сервер и так уже доступен, но трафик нужно зашифровать.

Транспортный режим не защищает заголовок IP. Другими словами,транспортный режим не защищает весь пакет IP, а только пакет транспортногоуровня (полезная нагрузка Р-уровня). В этом режиме IPSec-заголовок (и конечная метка) добавляется к информации, прибывающей от транспортного уровня. Заголовок IP добавляется позже.

IPSec в транспортном режиме не защищает заголовок IP, а только информацию, прибывающую от транспортного уровня.

Транспортный режим обычно применяется, когда мы нуждаемся в защите данных на участке "хост-хост" ("из конца в конец"). Передающий хост использует IPSec, чтобы подтвердить подлинность и/или зашифровать полезную нагрузку, освобожденную от информации транспортного уровня. Приемный хост использует IPSec, чтобы проверить установление подлинности и/или расшифровать пакет IP и доставить его транспортному уровню. Рисунок 1.26 иллюстрирует эту концепцию.

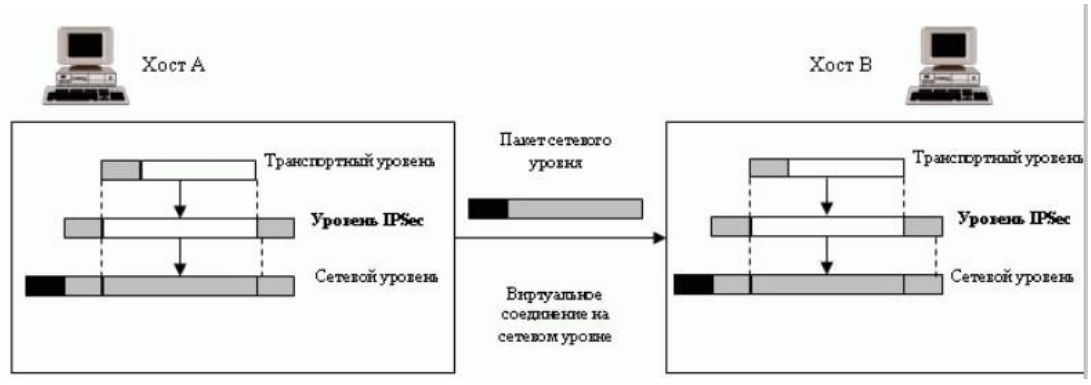


Рисунок 1.26 – Действия транспортного режима

Туннельный режим

В туннельном режиме IPSec защищает весь пакет IP. Он обрабатывает пакет IP (включая заголовок), применяя методы безопасности IPSec к полномупакету, и затем добавляет новый заголовок IP, как это показано на рисунке 1.27.

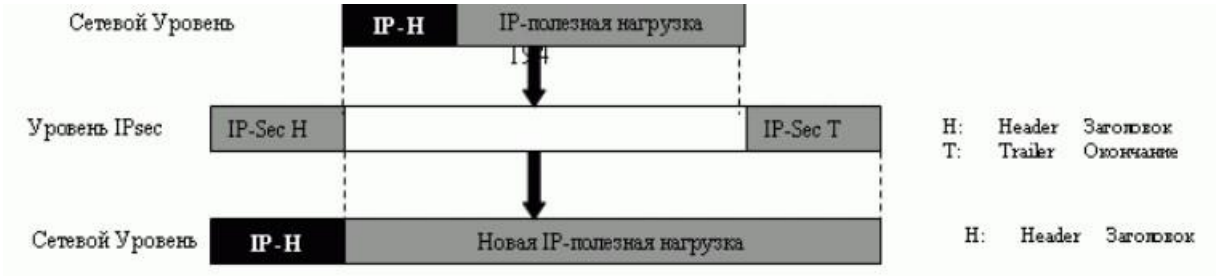


Рисунок 1.27 – Туннельный режим IPSec

В этом режиме берётся ваш изначальный IP-пакет, шифруется полностью, вместе с заголовком IP, добавляется служебная информация IPSeci новый заголовок IP (рис. 1.28):

Рисунок 1.28

Рисунок не точен и показывает лишь суть, на самом деле заголовков тамбольше, а также есть трейлеры в конце.

Новый заголовок IP содержит иную информацию, нежели первоначальный заголовок IP. Туннельный режим обычно используется между двумя маршрутизаторами, между хостом и маршрутизатором или между маршрутизатором и хостом, как это показано на рисунке 1.29. Другими словами, туннельный режим используется, либо как передатчик, либо как приемник не являющийся хостом.

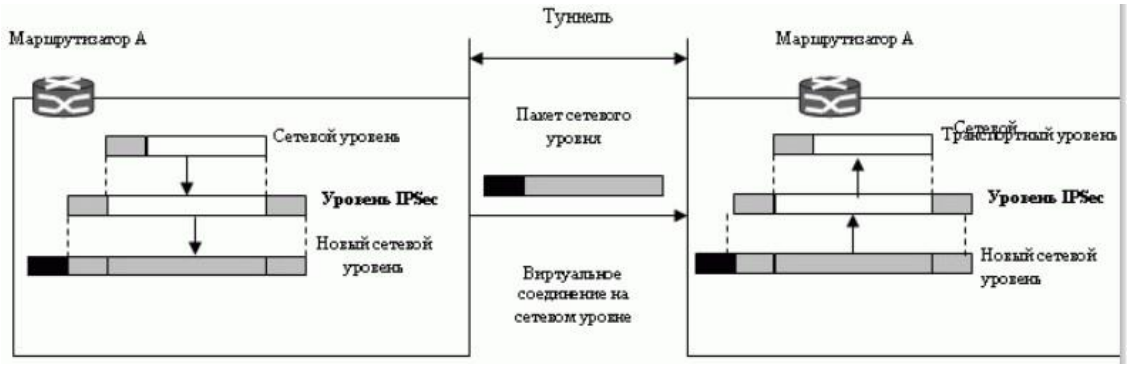


Рисунок 1.29 – Действия туннельного режима

Весь первоначальный пакет защищен от вмешательства между передатчиком и приемником, как будто весь пакет проходит мнимый туннель.

IPSec в туннельном режиме защищает первоначальный заголовок IP.

Сравнение

В транспортном режиме уровень IPSec располагается между транспортным уровнем и сетевым уровнем. В туннельном режиме поток проходит от сетевого уровня до уровня IPSec, а затем снова возвращается назад к сетевому уровню. рис. 1.30 сравнивает эти два режима.

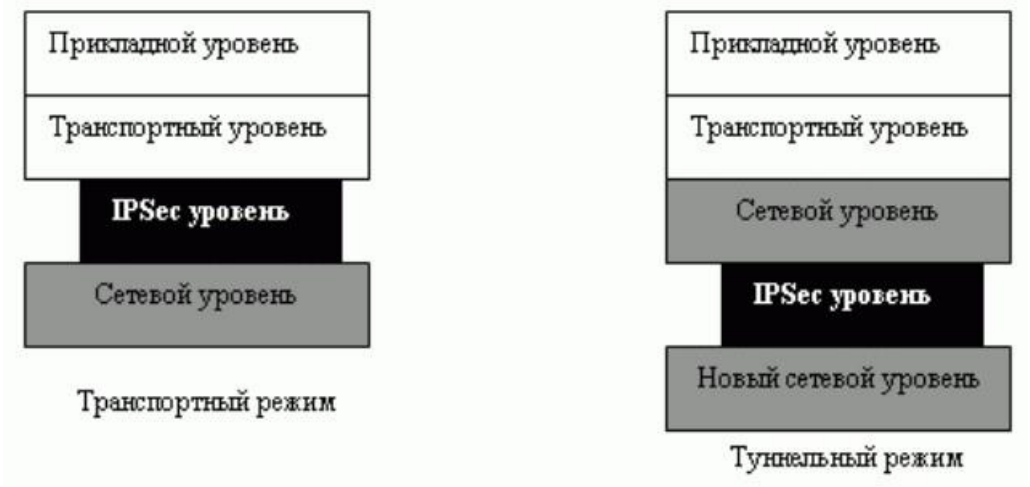


Рисунок 1.30 – Сравнение транспортного и туннельного режимов

Рисунок 1.31 - Сравнение транспортного и туннельного режимов

Протоколы безопасности

Протоколы сетевой безопасности – это сетевой тип протокола, который обеспечивает безопасность и целостность данных, передаваемых по сетевомусоединению.

Протоколы сетевой безопасности определяют процессы и методологию для защиты сетевых данных от любой незаконной попытки просмотра или извлечения содержимого данных.

IPSec определяет два протокола:

Протокол "Заголовок аутентификации (AH - Authentication Header)"; Протокол "Полезная нагрузка со встроенной защитой (ESP –

Encapsulating Security Payload)".

Их цель – обеспечить установление подлинности и/или шифрование дляпакетов на уровне IP.

Заголовок аутентификации (AH)

Протокол AH используется для аутентификации, но не для шифрованияIP трафика, и служит для подтверждения того, что мы связаны именно с тем,с кем предполагаем, что полученные данные не искажены и не подменены притранспортировке.

Аутентификация выполняется путем вычисления зашифрованного аутентификационного хэш-кода сообщения. Хэширование охватывает практически все поля IP пакета (исключая только те, которые могут модифицироваться при транспортировке, например, TTL или контрольная сумма заголовка). Этот код записывается в AH заголовке и пересылается получателю. Формат AH заголовка представлен на рисунок 1.32.

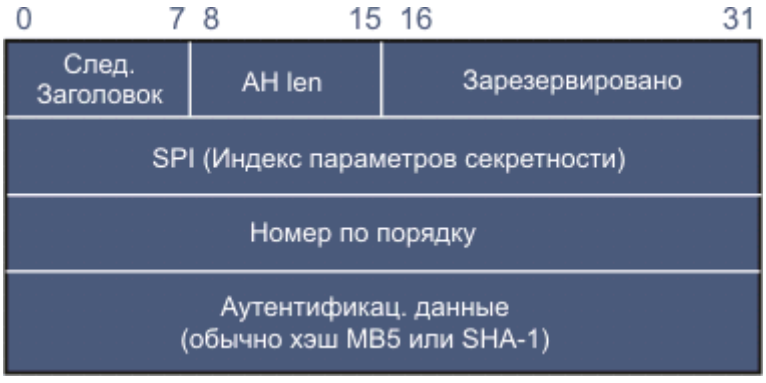


Рисунок 1.32 – Формат заголовка протокола AHЭтот AH заголовок содержит пять важных полей

«Следующий заголовок»: Идентифицирует тип протокола, используемого для следующего поляданных. Фактически это тип пакета, инкапсулированного в AH IPsec.

«AH len»: Определяет длину заголовка пакета, измеренную в 32-битовых словах,за

вычетом двух слов (это диктуется RFC 1883 для IPv6). «Зарезервировано»: Поле зарезервировано на будущее и должно содержать нули.

«Индекс параметров безопасности (SPI)»:

32-битовый идентификатор, который помогает получателю выбрать, к какому из входных обменов относится этот пакет. Каждый обмен, защищенный AH, использует хэш-алгоритм (MD5, SHA-1 и т.д..), какие-то секретные и возможно некоторые иные данные. SPI может рассматриваться как индекс таблицы наборов таких параметров, чтобы облегчить выбор нужного набора.

«Номер по порядку»:

Монотонно увеличивающийся идентификатор, который позволяет установить соответствие между посланным пакетом и откликом подтверждения его получения. Этот код включается в аутентификационные данные, что позволяет детектировать любые модификации, а также атаки воспроизведения.

«Аутентификационные данные»:

Это контрольная сумма ICV (Integrity Check Value), вычисленная для всего пакета, включая большинство полей заголовка (рисунок 1.11). Получатель повторно вычисляет тот же хэш. Если значения кодов не совпадут,пакет был поврежден в пути или не соответствует секретному ключу. Такие пакеты отбрасываются. ICV часто называется также MAC (Message Authentication Code). Для вычисления MAC используются следующие поля:

Поля IP-заголовка, которые не меняются при транспортировке.Заголовок AH, кроме поля данных *аутентификации*

Поле данных протокола верхнего уровня, которые остаются неизменными при транспортировке (рис. 1.33).

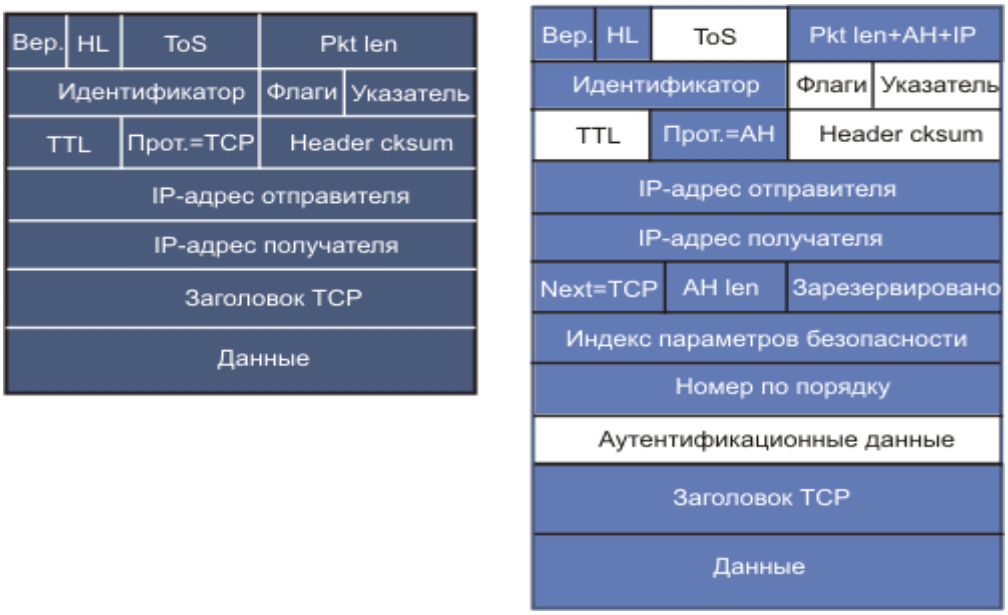


Рисунок 1.33 – Преобразование форматов в транспортном режиме AH IPSec

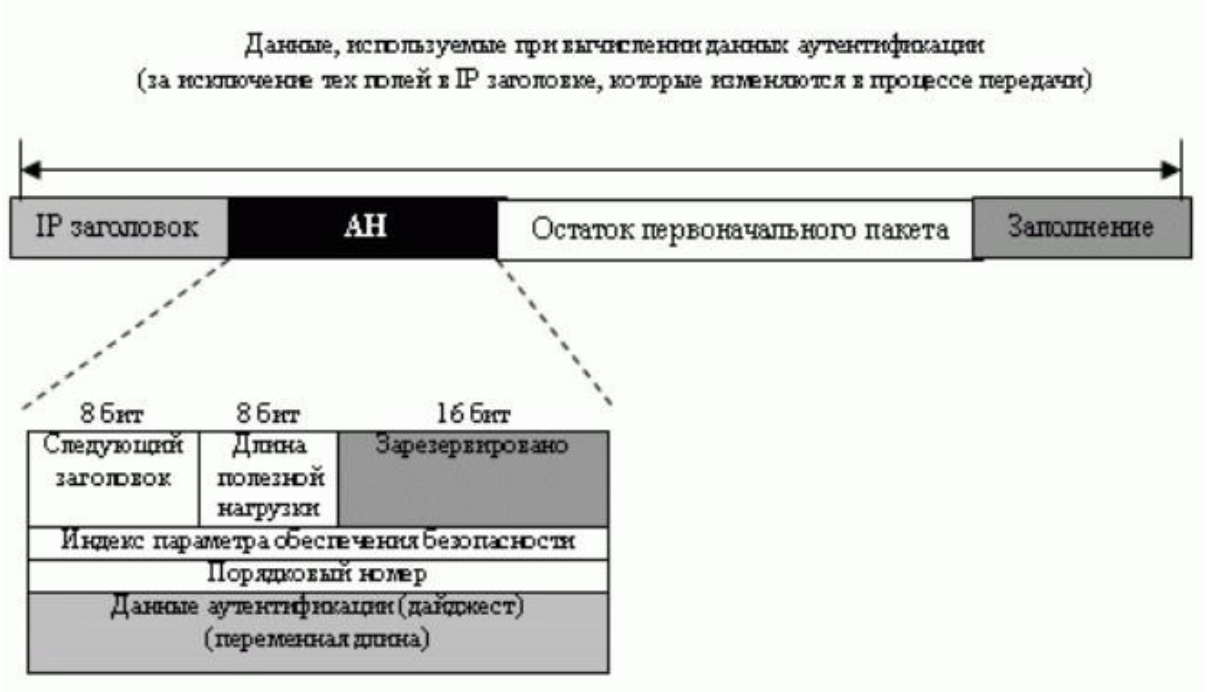


Рисунок 1.34 – Протокол «Заголовок аутентификации (AH)»

Когда дейтаграмма IP переносит заголовок аутентификации, первоначальное значение в поле протокола заголовка IP устанавливается в значение 51. Поле в заголовке аутентификации (следующее поле заголовка) содержит первоначальное значение поля протокола (тип полезной нагрузки, которую несет дейтаграмма IP). Добавление заголовка аутентификации проводится следующими шагами:

1. Заголовок аутентификации добавляется к полезной нагрузке с полем аутентификации данных, установленным на 0.

2. Заполнение добавляется, если нужно сделать полную длину сообщения для конкретного алгоритма хэширования.

3. Хэширование проводится на всем пакете. Однако в вычисление дайджеста сообщения (данные аутентификации) включены только те поля IP-заголовка, которые не изменяются в течение передачи.

4. Данные аутентификации вставляются в заголовок аутентификации.

5. Заголовок IP добавляется после изменения значения поля протокола на 51.

Протокол AH обеспечивает установление подлинности источника и целостность данных, но не конфиденциальность.

Полезная нагрузка со встроенной защитой (ESP)

Протокол АН не обеспечивает секретность, а только установление подлинности источника и целостность данных. Для IPSec был определенальтернативный протокол Полезная нагрузка со встроенной защитой (ESP- Encapsulating Security Payload), который гарантирует установление подлинности источника, целостность и секретность. ESP добавляет заголовоки конечную метку. Обратите внимание, что данные аутентификации ESPдобавляются в конце пакета – это делает их вычисление более простыми. Рисунок 1.34 показывает размещение заголовка ESP и конечной метки (рис.1.35).

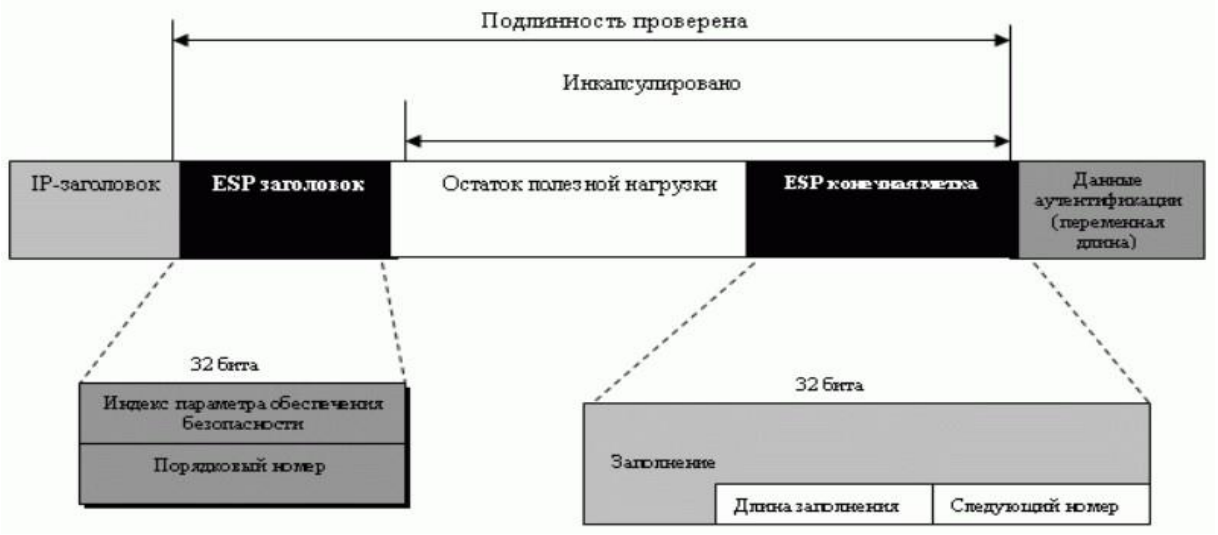


Рисунок 1.35 – Протокол «Полезная нагрузка со встроенной защитой» (ESP)

Когда дейтаграмма IP переносит заголовок ESP и конечную метку, значение поля протокола в заголовке IP равно 50. Поле в конечной метке ESP(поле следующего заголовка) содержит первоначальное значение поляпротокола (тип полезной нагрузки, которую несет дейтаграмма IP, такой как TCP или UDP). Процедура ESP выполняется следующими шагами:

1. Конечная метка ESP добавляется к полезной нагрузке.
2. Полезная нагрузка и конечная метка зашифровываются.
3. Добавляется заголовок ESP.
4. Заголовок ESP, полезная нагрузка и конечная метка ESP используются, чтобы создать данные аутентификации.
5. Данные аутентификации добавляются в конце конечной метки ESP.
6. Заголовок IP добавляется после изменения значения протокола на 50. Поля заголовка и конечной метки, следующие:

- Индекс параметра обеспечения безопасности. Поле индекса параметраобеспечения безопасности на 32 бита совпадет с тем, которое определено дляпротокола АН.

- Порядковый номер. Поле порядкового номера на 32 бита совпадет с тем, которое определено для протокола АН.
- Заполнение – это поле переменной длины (от 0 до 255 байтов), состоящее из нулей и служащее заполнением.
- Длина заполнения. Поле длины заполнения на 8 битов определяет число байтов заполнения между 0 и 255; максимальное значение используетсяредко.
- Следующий заголовок. Поле следующего заголовка на 8 битов совпадет с тем, которое определено для протокола АН. Оно выполняет ту же самую задачу, как и поле протокола в заголовке IP перед инкапсуляцией.
- Данные аутентификации. Наконец, поле данных аутентификации - результат применения схем аутентификации к частям дейтаграммы. Обратите внимание на отличие между данными аутентификации в АН и ESP. В АН частьзаголовка IP включена в вычисление данных аутентификации, а в ESP – нет.

ESP обеспечивает установление подлинности источника, целостность данных и секретность.

Сравнение АН и ESP

Протокол ESP был разработан после того, как протокол АН был уже в использовании. ESP умеет то, что АН делает только с дополнительными функциональными возможностями (секретность). Вопрос: почему же тогда мы по-прежнему нуждаемся в АН? Этот вопрос не имеет ответа. Однако реализация АН включена в несколько коммерческих продуктов, то есть АН останется частью Интернет, пока эти продукты не будут постепенно выведены из употребления.

Услуги, обеспечиваемые IPSec

Два протокола АН и ESP, могут обеспечить несколько услуг безопасности для пакетов на сетевом уровне. Таблица 1.6 показывает список услуг, доступных для каждого из этих протоколов.

Таблица 1.6 – Услуги IPSec

Услуги	**АН**	**ESP**
Управление доступом Access control	ДА	ДА
Управление подлинности сообщения (целостность сообщения) Message authentication (message integrity)	ДА	ДА

Установление подлинности объекта (установление подлинности источника данных) Entity authentication (data source authentication)	ДА	ДА
Конфиденциальность Confidentiality	НЕТ	ДА
Защита от атаки воспроизведения Replay attack protection	ДА	ДА

Управление доступом

IPSec обеспечивает управление доступом, косвенно использующее базу данных услуг обеспечения безопасности трафика (SAD - Security Association

Database), как мы это увидим в следующей секции. Когда пакет достигает пункта назначения и атрибуты службы обеспечения безопасности трафика, установленные для этого пакета, отсутствуют, пакет бракуется.

Целостность сообщения сохраняется и в AH, и в ESP. Дайджест данных создается и посылается передатчиком, который будет проверен приемником.

Службы обеспечения безопасности трафика и дайджест ключевого хэширования данных, посланных передатчиком, подтверждают подлинность передатчика данных и в AH, и в ESP.

Конфиденциальность

Шифрование сообщения обеспечивает конфиденциальность в ESP. AH однако, конфиденциальность не гарантирует. Если конфиденциальность необходима, нужно использовать ESP вместо AH.

Защита от атаки воспроизведения

В обоих протоколах предотвращается атака воспроизведения путем использования порядковых номеров и скользящего окна приемника. Каждый IPSec-заголовок содержит уникальный порядковый номер - когда установлены Службы обеспечения безопасности трафика. Числа начинаются от 0 и увеличиваются, пока не достигают значения $2^{32} – 1$ (размер поля порядкового номера - 32 бита). Когда порядковый номер достигает максимума, он сбрасывается в 0, и в то же самое время удаляются старые Службы обеспечения безопасности трафика и устанавливаются новые. Чтобы предотвращать пакеты дубликата обработки, IPSec использует фиксированный размер окна приемника. Размер окна приемника задан по умолчанию значением 64. Рисунок 1.36 показывает окно ответа. Окно имеет фиксированный размер W. Затемненные пакеты показывают, что полученные пакеты были проверены и аутентифицированы.

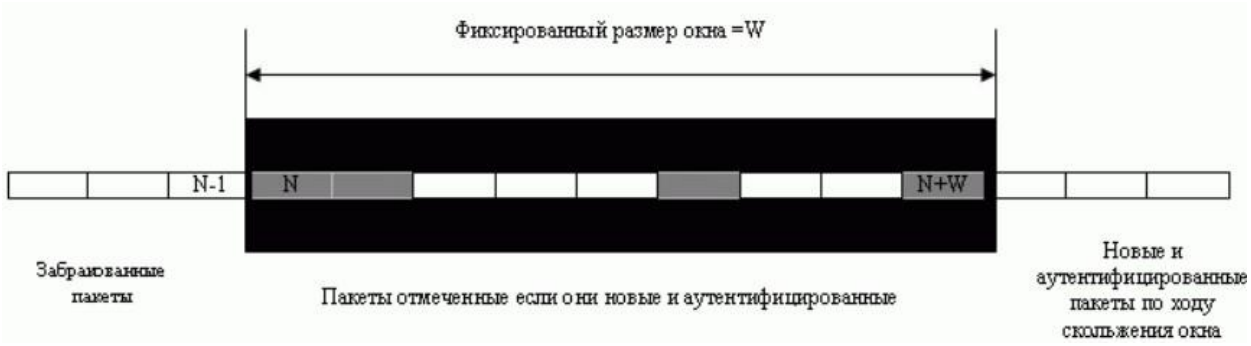


Рисунок 1.36 – Окно ответа

Когда пакет достигает приемника, в зависимости от значения порядкового номера может произойти одно из трех событий:

1. Порядковый номер пакета – меньше, чем N. Тогда пакет размещаетсяналево от окна и будет забракован. Он либо является дубликатом, либо времяего прибытия истекло.

2. Порядковый номер пакета - между N и (N + W - 1) включительно. Тогда пакет размещается в окне. В этом случае, если пакет новый (неотмеченный) и он содержит аутентификационный тест, порядковый номеротмечается, и пакет принимается. Иначе (если пакет не новый) он бракуется.

3. Порядковый номер пакета больше, чем (N + W - 1). Тогда пакет размещается справа от окна. В этом случае, если пакет аутентифицирован, соответствующий порядковый номер отмечается и окно перемещается(скользит) вправо и занимает отмеченный порядковый номер. Иначе (если не аутентифицирован) пакет бракуется. Обратите внимание, что это может случиться, если пакет прибывает с порядковым номером, намного большим, чем (N + W) (очень далеким от правого края окна). В этом случае скольжениевправо может привести к попаданию немаркированных порядковых номеров налево от окна. Эти пакеты, когда они прибывают, никогда не будут приниматься; их время истекло. Например, рисунок 1.35, если пакетприбывает с порядковым номером (N + W + 3), окно перемещается и левый край начнется с (N + 3). Это означает, что порядковый номер (N + 2) теперь вне окна. Если пакет прибывает с этим порядковым номером, он будет забракован.

Услуги обеспечения безопасности трафика

Услуги обеспечения безопасности трафика - очень важный аспект IPSec. IPSec требует между двумя хостами логических отношений, называемых услуги обеспечения безопасности трафика (SA – Security Association). В этом разделе вначале рассмотрим идею, а затем покажем, как она используется в IPSec.

Услуга обеспечения безопасности трафика (SA -Security Association) - соглашение между двумя сторонами для создания безопасного канала между ними. Предположим, что Алиса должна однонаправленно связаться с Бобом. Если Алиса и Боб интересуются только аспектом конфиденциальности и безопасности, они могут получить общедоступный ключ засекречивания для связи между собой. Мы можем сказать, что это есть две услуги обеспечения безопасности трафика (SA's) между Алисой и Бобом: одна SA - исходящая и одна SA - входящая. Каждый из них хранит значение ключа и имя алгоритма шифрования/дешифрования. Алиса использует алгоритм и ключ, чтобы зашифровать сообщение Бобу; Боб использует алгоритм и ключ, когда он должен расшифровать сообщение, полученное от Алисы. Рисунок 1.37 показывает эти простые услуги SA.

Услуги обеспечения безопасности трафика могут быть расширены, еслинаши две стороны нуждаются в гарантии целостности сообщения и установлении подлинности. Тогда каждое такое сообщество нуждается в дополнительных данных, например, таких как алгоритм для целостности сообщения, ключ и другие параметры. Все может быть намного сложнее, если стороны использовали различные протоколы, которые имеют разные алгоритмы и параметры - например, IPSec AH или IPSec ESP.

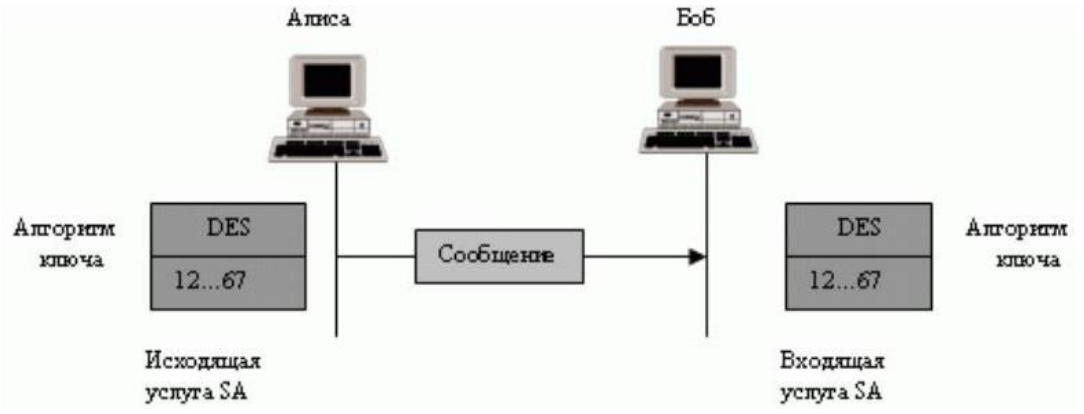


Рисунок 1.37 – Простые услуги обеспечения безопасности (SA)

База данных услуг обеспечения безопасности

Услуги обеспечения безопасности могут быть организованы очень сложно. Это особенно справедливо, если Алиса хочет передавать сообщения многим людям, а Боб должен получать сообщения от многих людей. Кроме того, каждая сторона должен иметь и входящие, и исходящие SA's, чтобы позволить осуществлять двунаправленную связь. Другими словами, мы нуждаемся во множестве SA's, которые могут быть собраны в базу данных. Эта база данных называется Базой данных Услуг обеспечения безопасности (SAD - Security Association Database). Базу данных можно представлять как двумерную таблицу с каждой строкой,

определяющей единственную SA. Обычно есть две SAD's: одна - входящая и одна - исходящая. Рисунок 1.38 показывает концепцию исходящих и входящих SAD's для одного объекта.

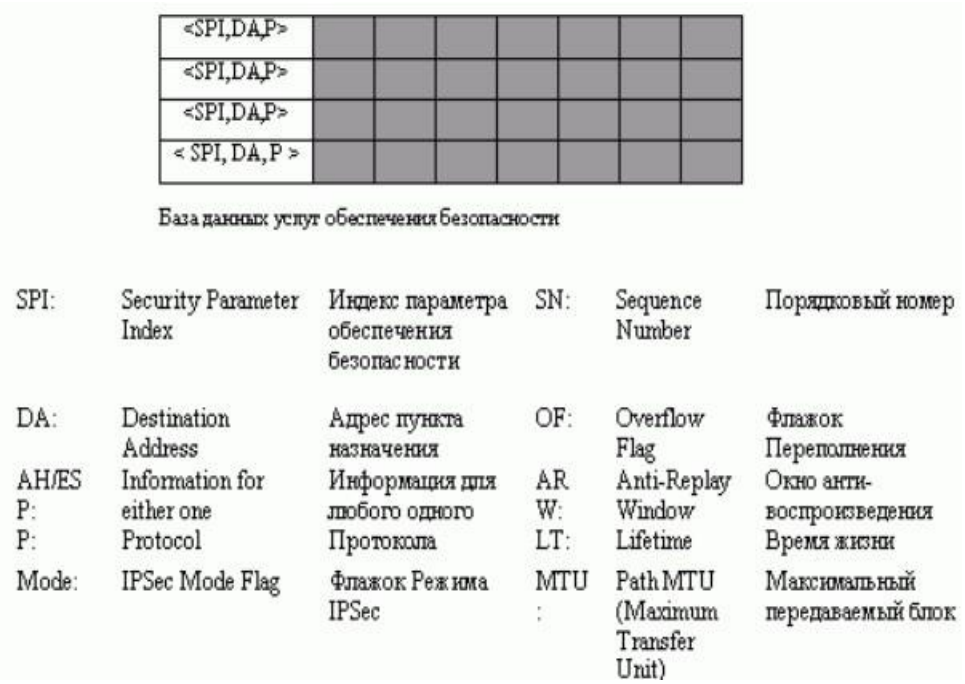


Рисунок 1.38 – База данных услуг обеспечения безопасности

Когда хост должен передать пакет, который должен доставить IPSec- заголовок, хост должен найти соответствующий исходящий SAD и найти информацию для того, чтобы применить услуги безопасности к пакету. Точнотак же, когда хост получает пакет, который должен доставить IPSec-заголовок,хост должен найти соответствующий вход во входящем SAD, найти нужную информацию для проверки безопасности пакета. Этот поиск должен быть задан так: приемный хост должен убедиться, что для обработки пакета используется правильная информация, чтобы обработать пакет. Каждый входво входящем SAD выбирается, используя тройной индекс: индекс параметра обеспечения безопасности, адрес пункта назначения и протокол.

Индекс параметра обеспечения безопасности. Индекс параметра обеспечения безопасности (SPI) - число на 32 бита, которое определяет SA в пункте назначения. Как мы увидим позже, SPI определен в течениеSAпереговоров. Тот же самый SPI включен во все IPSec-пакеты, принадлежащие одному и тому же входящему SA.

Адрес пункта назначения. Второй индекс - адрес пункта назначения хоста. Мы должны помнить, что хост в Интернете обычно имеет один индивидуальный адрес пункта назначения, но он может иметь несколько адресов групповой рассылки. IPSec требует, чтобы SA был уникален для каждого адреса пункта назначения.

Протокол. IPSec имеет два различных протокола безопасности: AH и ESP. Чтобы отделить параметры и информацию, используемую для каждого протокола, IPSec требует, чтобы пункт назначения определял различный SA для каждого протокола.

Входы для каждой строки называются параметрами SA. Типичные параметры показаны в таблица 1.7.

Таблица 1.7 – Типичные параметры SA

Счетчик порядкового номера Sequence Number Counter	Это значение на 32 бита, которое используется, чтобы генерировать порядковые номера для АНили ESP-заголовка
Переполнение порядкового номера Sequence Number Overflow	Это флажок, который определяет варианты состояния в случае переполнения порядкового номера.
Окно антивоспроизведения AntiReplay Window	Обнаруживает входящий воспроизведенный пакет АН или пакет ESP
Информация АН AH Information	Эта секция содержит информацию для протокола АН: 1. Алгоритмаутентификации 2. Ключи 3. Время жизни ключа 4. Другие связанные параметры
Информация ESP ESP Information	Эта секция содержит информацию для протокола ESP: 1. Алгоритм шифрования 2. Алгоритм аутентификации 3. Ключи 4. Время жизни ключа 5. Вектор инициализации 6. Другие связанные параметры
Время жизни SA LifetimeРежим IPSec IPSec Mode	Определяет время жизни для SA. Определяет режим, транспортный илитуннельный
Путь MTU Path MTU	Определяет путь МАКСИМАЛЬНЫЙ ПЕРЕДАВАЕМЫЙ БЛОК (фрагментацию). This defines the pathMTU (fragmentation).

Стратегия безопасности

Другой аспект импорта IPSec - Стратегия безопасности (SP - Security Policy), которая определяет тип безопасности, предоставляемой пакету, когдаего нужно передать или когда его нужно принять. Перед тем как использоватьSAD, рассмотренный в предыдущем разделе, хост должен определить заранеезаданную стратегию обслуживания этого пакета.

База данных стратегий безопасности

Каждый хост, который использует IPSec-протокол, должен хранить Базуданных стратегии безопасности (SPD - Security Policy Database). При этом, как и раньше, необходимо иметь входящую SPD и исходящую SPD. К каждому входу в SPD можно обратиться, используя индекс из шести позиций: исходный адрес, адрес пункта назначения, название, протокола, исходный порт и порт пункта назначения, как показано на рисунке 1.39.

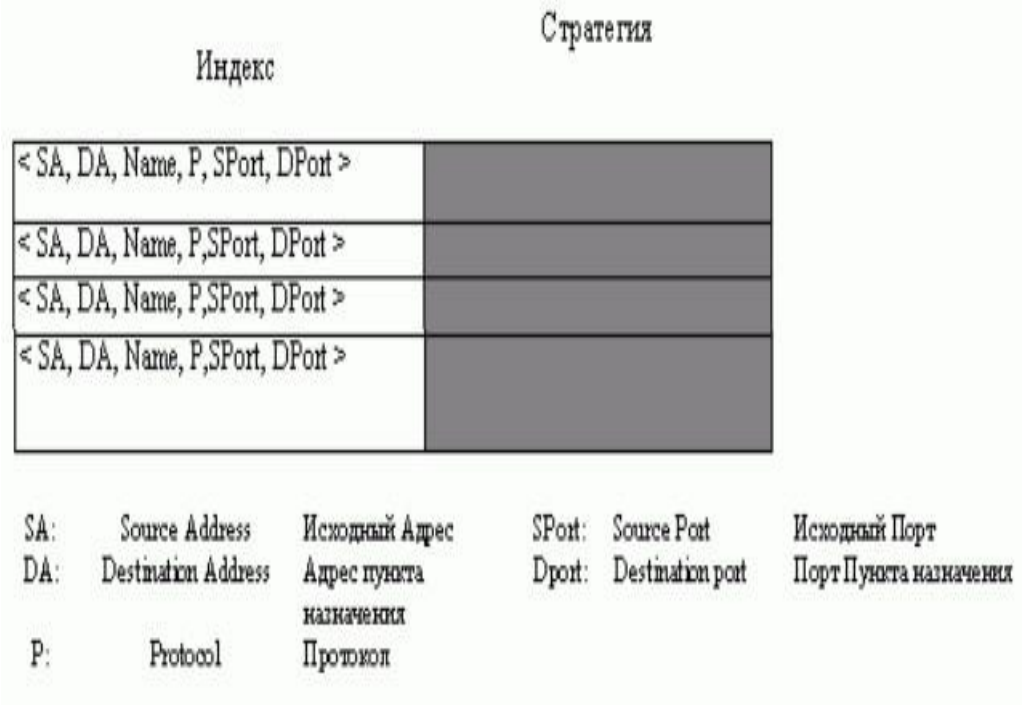


Рисунок 1.39 – База данных Стратегий Безопасности (SPD)

Исходящий SPD

Когда пакет нужно передать, то работают с исходящим SPD. Рисунок

1.38 показывает обработку пакета передатчиком. Вход исходящего SPDсостоит из шестизначного индекса; выход содержит один из трех результатов:

1. Скинуть. Это означает, что пакет, определенный этим индексом,нельзя передать; он отбрасывается.

2. Обход. Это означает, что нет никакой стратегии для пакета с этим индексом стратегии; пакет передают в обход, не применяя действий с заголовком безопасности.

3. Применить. В этом случае применяется работа с заголовкомбезопасности. При этом могут возникнуть две ситуации: о Если исходящую SA уже установили, возвращается тройной индекс SA, который выбирается соответствующей SA из исходящего SAD. Формируется AH или заголовок ESP; шифрование, установление подлинности или оба этих действия применяются в соответствии с выбранной SA. Пакет передается.

Если исходящая SA еще не установлена, то вызывается протокол Интернет обмена ключами (IKE - Internet Key Exchange) (см. следующую секцию), чтобы создать исходящую и входящую SA для этого трафика. Исходящая SA добавляется источником к исходящей SAD; входящая SAдобавляется пунктом назначения к входящей SAD.

Входящий SPD

Когда пакет прибывает, то проводится работа с входящей SPD. К каждому входу во входящей SPD также обращаются, используя тот же самыйшестикратный индекс. Рисунке 1.39 показывает обработку пакета приемником. Вход к входящему SPD - шестикратный индекс; выход - один изтрех результатов:

1. Сбросить. Это означает, что пакет, определенный стратегией, должен быть отброшен.

2. Обход. Это означает, что нет никакой стратегии для пакета с этим индексом стратегии; пакет обрабатывается, игнорируя информацию от AH илизаголовок ESP. Пакет доставляют транспортному уровню.

3. Применить. В этом случае заголовок безопасности должен быть обработан. Здесь могут возникнуть две ситуации: о если входящая SA уже установлена, возвращается тройной индекс SA, который выбирается соответствующей SA из входящего SAD. Применяются дешифрование, установление подлинности или оба этих действия. Если пакет передаеткритерии безопасности, AH или заголовок ESP забракован, и пакет доставляют транспортному уровню; о если SA еще не установлена, то пакет должен быть забракован.

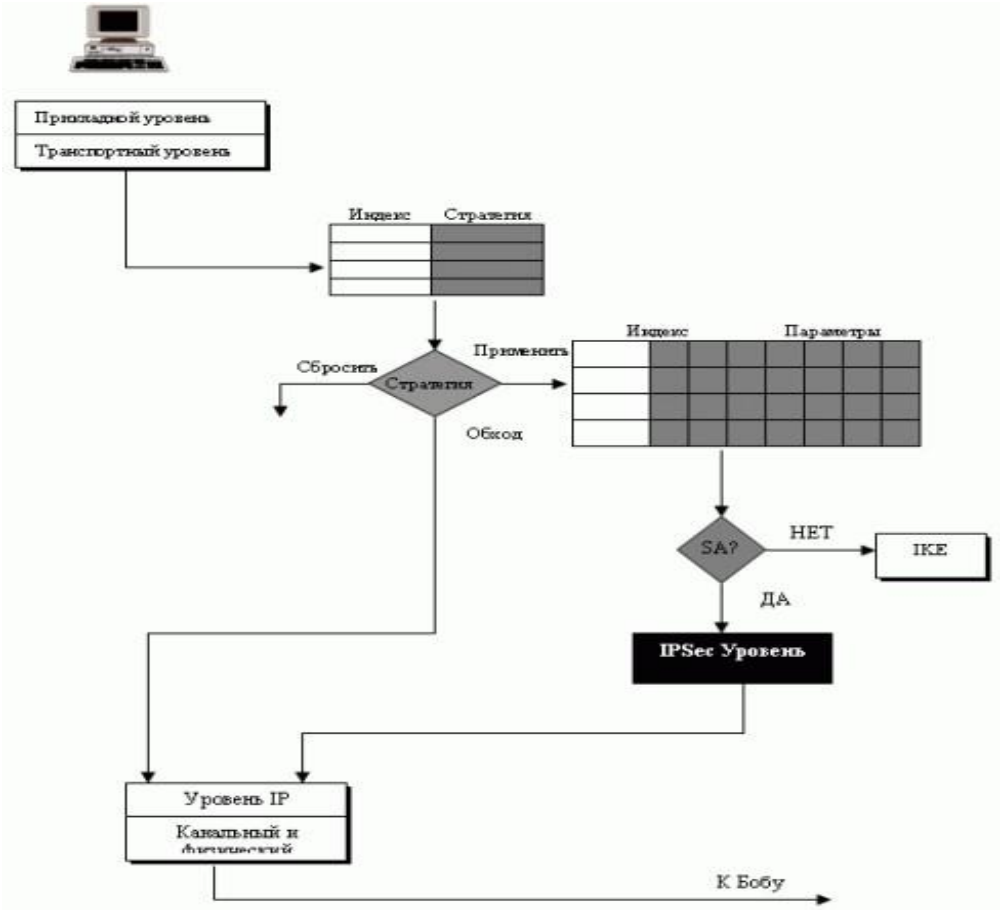


Рисунок 1.40 – Исходящий процесс

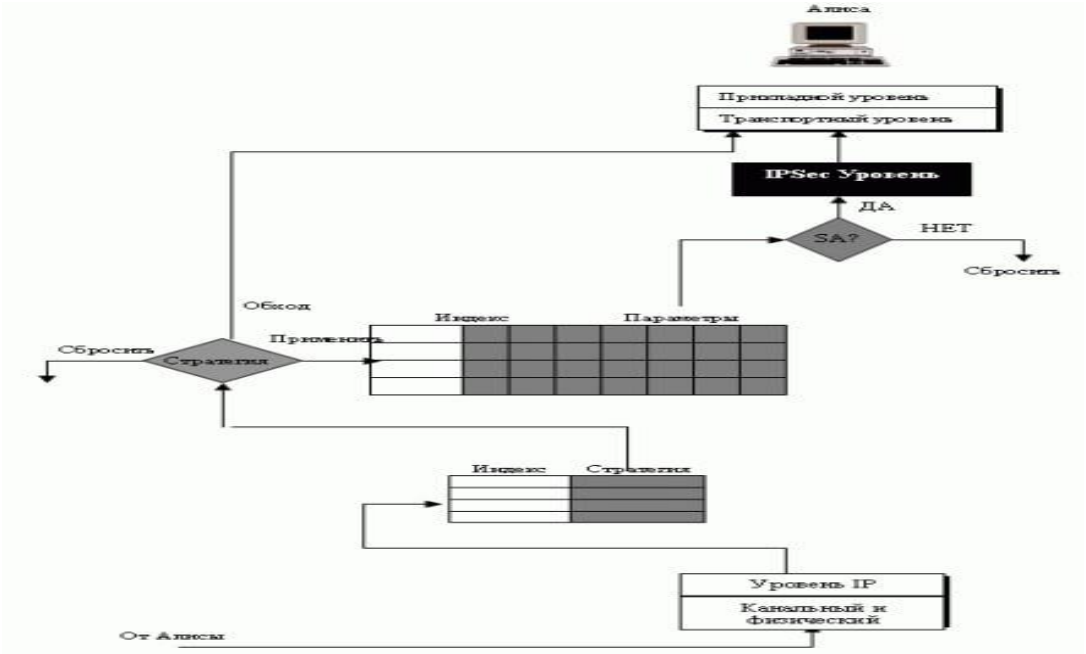


Рисунок 1.41 – Входящий процесс

Протокол Интернет-обмена ключами (IKE - Internet Key Exchange) должен создавать и входящие, и исходящие услуги обеспечения безопасности.Как мы обсуждали в предыдущей секции, когда пакет IP должен быть переданмежду равными уровнями, тогда обращаются к базе данных стратегии безопасности (SPDB), чтобы видеть, есть ли SA для такого типа трафика. Еслинет такой SA, вызывается IKE, чтобы установить ее.

Протокол Интернет-обмена ключами создает услуги обеспечения безопасности SA 's для протокола IPSec. IKE - сложный протокол, основанныйна трех других протоколах: OAKLEY, SKEME и ISAKMP, как показано на рисунке 1.42.

Рисунок 1.42 – Компоненты протокола управления ключами в Интернете

Фазы IKE

IKE создает SA's для обмена сообщениями протокола, такого как IPSec. IKE, однако, должен обмениваться конфиденциальными и аутентифицированными сообщениями. Какие SA's обеспечивает протокол IKE для себя? Можно догадаться, что он требует бесконечной цепочки SA's: IKE должен создать SA's для IPSec, протокол X должен создать SA's для IKE, протокол Y должен создать SA's для протокола X, и так далее. Для того чтобырешить эту дилемму и в то же время оставить IKE независимым от протоколаIPSec, разработчики IKE разделили IKE на две фазы. В фазе I IKE создает SA's для фазы II. В фазе II IKE создает SA's для IPSec или некоторых других протоколов.

Фаза I является базовой; фаза II задана для протокола. IKE разделен на две фазы: фаза I и фаза II. Фаза I создает SA 's для фазы II; фаза II создает SA 'sдля протокола обмена данными, например, такого как IPSec.

Фазы и режимы

Чтобы учесть разнообразие методов обмена, IKE определил для фаз режимы. В настоящее время есть два режима для фазы I: главный режим и энергичный режим. Единственный режим для фазы II - быстрый режим. рис. 1.43 показывает отношения между фазами и режимами.

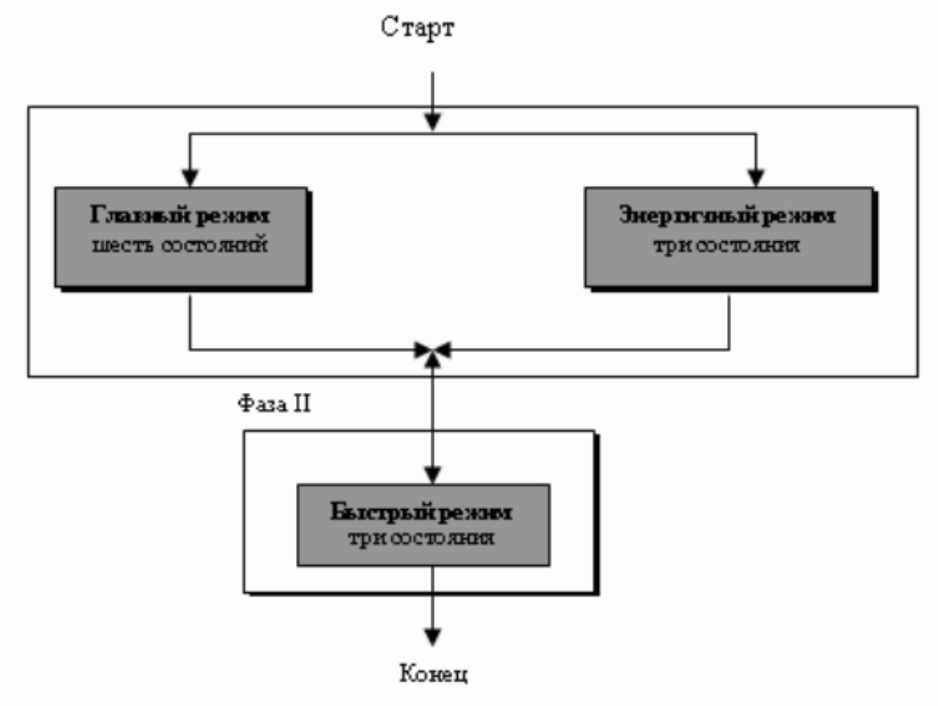


Рисунок 1.43 – Фазы IKE

В зависимости от характера предварительной секретности между этимидвумя сторонами, режимы фазы I могут использовать один из четырех различных методов аутентификации: метод предварительного открытого ключа засекречивания, метод первоначального открытого ключа, метод пересмотренного открытого ключа или метод цифровой подписи, как это показано на рисунке 1.44.

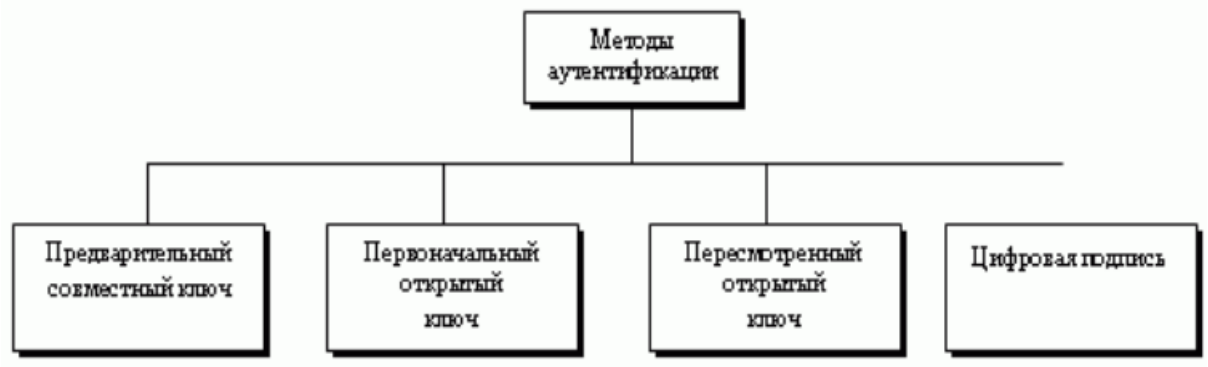


Рисунок 1.44 – Методы основного или энергичного режима

Фаза I: основной режим

В основном режиме инициатор и респондент обмениваются шестью сообщениями. В первых двух сообщениях они обмениваются cookies (чтобы защитить против засоряющей атаки и договориться о параметрах SA): инициатор передает ряд предложений; респондент выбирает одно из них. Когда они обменяются первыми двумя сообщениями, инициатор и респондентзнают параметры SA и уверены, что другая сторона существует и засоряющаяатака не возникнет.

В третьих и четвертых сообщениях инициатор и респондент обычно обмениваются своими полуключами (g^i и g^r метода Диффи-Хеллмана) и их nonce (для защиты ответа). В некоторых методах обмениваются другой информацией, о которой мы поговорим позже. Обратите внимание, что полуключи и nonce не передаются с первыми двумя сообщениями, потому что две стороны должны сначала получить гарантию, что засоряющая атака невозможна.

Безопасность сети ПД на сетевом уровне IPSec

IPSec является неотъемлемой частью IPv6 - Интернет-протоколаследующего поколения, и расширением существующие версииИнтернетпротокола IPv4. IPSec определён в RFC с 2401 по 2412.

Практически все механизмы сетевой безопасности могут быть реализованы на третьем уровне эталонной модели ISO/OSI. Кроме того, IP-уровень можно считать оптимальным для размещения защитных средств, поскольку при этом достигается удачныйкомпромисс между защищенностью, эффективностью функционирования и прозрачностью для приложений (рис. 1.45).

Уровни TCP/IP	Уровни ISO/OSI
4. Прикладных программ	7. Прикладных программ 6. Представление данных
3. Транспортный	5. Сеансовый 4. Транспортный
2. Межсетевой	3. Сетевой
1. Доступа к сети	2. Канальный 1. Физический

Рисунок 1.45 – Модель OSI/ISO

Стандартизованными механизмами IP-безопасности могут (и должны) пользоваться протоколы более высоких уровней и, в частности, управляющие протоколы, протоколы конфигурирования и маршрутизации.

Средства безопасности для IP описываются семейством спецификаций IPSec, разработанных рабочей группой IP Security.

Основные структуры безопасности

Архитектура средств безопасности для IP-уровня специфицирована в документе Security Architecture for the Internet Protocol. Это, прежде всего протоколы обеспечения аутентичности (протокол аутентифицирующего заголовка - Authentication Header, AH) и конфиденциальности (протокол инкапсулирующей защиты содержимого - Encapsulating Security payload, ESP),а также механизмы управления криптографическими ключами. На более низком архитектурном уровне располагаются конкретные алгоритмы шифрования, контроля целостности и аутентичности. Наконец, роль фундамента выполняет так называемый домен интерпретации (Domain ofInterpretation, DOI), являющийся, по сути, базой данных, хранящей сведения об алгоритмах, их параметрах, протокольных идентификаторах (рис. 1.46).

Рисунок 1.46 – Основные элементы архитектуры средств безопасности IP-уровня

Деление на уровни важно для всех аспектов информационных технологий. Там же, где участвует еще и криптография, важность возрастает вдвойне, поскольку приходится считаться не только с чисто техническими факторами, но и с особенностями законодательства различных стран, с ограничениями на экспорт и/или импорт криптосредств.

IPSec поддерживает две формы целостности: целостность соединения и частичную целостность последовательности. Целостность соединения является сервисом безопасности,

который определяет модификацию конкретной IP датаграммы, безотносительно последовательности датаграмм впотоке трафика. Частичная целостность последовательности является antireply сервисом, с помощью которого определяется получение дубликатов IP датаграм.

Алгоритмическая независимость протоколов имеет и оборотную сторону, состоящую в необходимости предварительного согласования набора применяемых алгоритмов и их параметров, поддерживаемых общающимися сторонами. Иными словами, стороны должны выработать общий контекст безопасности (Security Association, SA) и затем использовать такие его элементы, как алгоритмы и их ключи. SA подробно рассматривается далее. За формирование контекстов безопасности в IPSec отвечает особое семейство протоколов ISAKMP, которое рассматривается также в отдельном разделе. Безопасность, обеспечиваемая IPSec, зависит от многих факторов операционного окружения, в котором IPSec выполняется. Например, от безопасности ОС, источника случайных чисел, плохих протоколов управления системой.

Таким образом, данном разделе была рассмотрена безопасностьсетевого уровня IPSec – это метод безопасного и зашифрованного обменаданными между клиентом и хостом. Клиентом может быть устройство, например, ноутбук или частная сеть. Хостом чаще всего бывает частная сеть.

Сам IPsec не является протоколом; это набор протоколов, которые используются вместе. Протоколы, которыми пользуется IPsec, начинаются на уровне Layer 3 модели OSI, что, возможно, делает IPsec безопаснее, чем TLS или SSL. IPsec обычно используется для VPN, то также подходит для подключения двухчастных сетей.

IPSec работает в двух режимах транспортом (защищает информацию, доставляемую от транспортного уровня к сетевому, но не защищает заголовокIP) и туннельном (защищает весь пакет IP, включая первоначальный заголовок IP).

IPSec определяет два протокола: протокол заголовка аутентификации (AH) и полезную нагрузку со встроенной защитой нагрузка (ESP). Эти протоколы обеспечивают аутентификацию, шифрование или и то и другое дляпакетов на уровне IP. Протокол заголовка аутентификации (AH) подтверждаетподлинность хоста источника и гарантирует целостность полезной нагрузки, которую несет пакет IP. Протокол " полезная нагрузка со встроенной защитой"(ESP) обеспечивает исходную аутентификацию, целостность и секретность. ESP добавляет к формату заголовок и конечную метку.

Контрольные вопросы

1. Какие механизмы безопасности обеспечивает IPSec?
2. Определите протокол AH и услуги безопасности, которые он обеспечивает.
3. Определите протокол ESP и услуги безопасности, которые он обеспечивает.
4. Определите услуги безопасности, которые он обеспечивает протокол управление ключами (IKE).
5. Определите услуги обеспечения безопасности (SA) и объясните их цель.
6. Опишите основные элементы архитектуры средств безопасности IP-уровня.
7. Опишите работу транспортного режима работы IPSec.
8. Опишите работу туннельного режима работы.
9. Опишите работу протокола обмена ключами – IKE.
10. Дайте понятие "безопасные ассоциации" (Security Association - SA), которые является фундаментальным в IPSec.
11. Приведите типичные параметры SA.
12. Опишите метод смены ключей Диффи-Хеллмана.
13. Какие алгоритмы хэширования использует IPSec?
14. Какие алгоритмы шифрования в режиме CBC использует IPSec?

1.1.3 БЕЗОПАСНОСТЬ НА ПРИКЛАДНОМ УРОВНЕ: PGP

PGP называется *Очень хорошей конфиденциальностью (PGP - Pretty Good Privacy). PGP* был изобретен Филом Цимерманном (Phil Zimmermann), чтобы обеспечить секретность, целостность и установление подлинности *электронной почты*. *PGP* может использоваться, чтобы создать безопасное почтовое сообщение или надежно сохранить файл для будущего извлечения.

Сценарии

Сначала обсудим общую идею *PGP*, продвигаясь от простого сценария к сложному. Мы используем термин "Данные", чтобы указать сообщение или файл для обработки.

Открытый текст

Самый простой сценарий - это передать почтовое сообщение (или накопленный файл) в исходном тексте, как это показано на 1.46. В этом сценарии нет сохранения целостности сообщения или конфиденциальности. Алиса (передатчик) составляет сообщение и передает его Бобу (приемнику). Сообщение сохраняется в почтовом ящике Боба, пока не будет извлечено им (рис. 1.47).

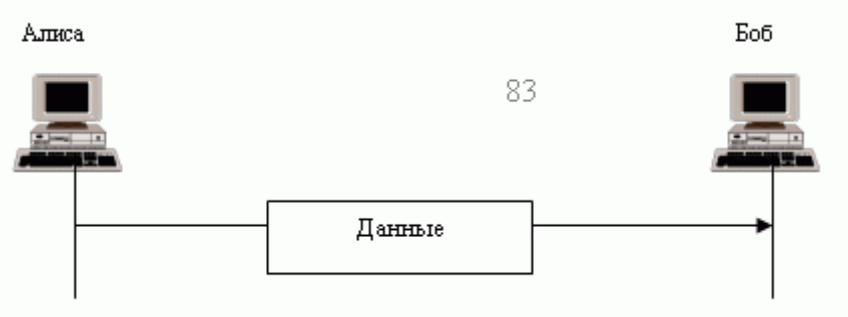


Рисунок 1.47 - Сообщение с обычным текстом

Целостность сообщения

Вероятно, следующее усовершенствование должно позволить Алисе подписывать сообщение. Алиса создает дайджест сообщения и подписывает его своим секретным ключом. Когда Боб получает сообщение, он проверяет его, используя открытый ключ Алисы. Для этого сценария необходимы два ключа. Алиса должна знать свой секретный ключ; Боб должен знать открытый ключ Алисы. Рис. 1.48 показывает ситуацию.

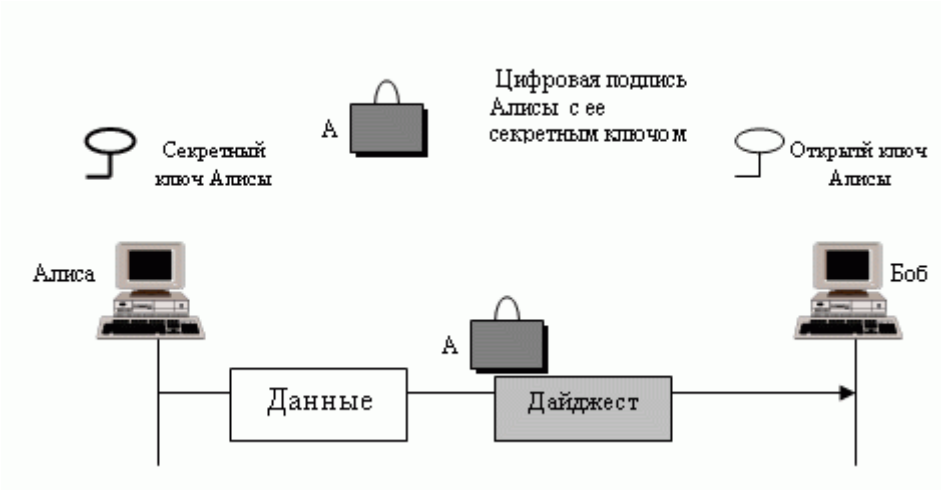


Рисунок 1.48 - Сообщение с подтверждением подлинности передатчика

Сжатие

Дальнейшее усовершенствование позволяет сжать сообщение и дайджест, чтобы сделать пакет более компактным. Это усовершенствование не имеет никаких преимуществ с точки зрения безопасности, но существенно уменьшает трафик. Рисунок 1.49 показывает новый сценарий.

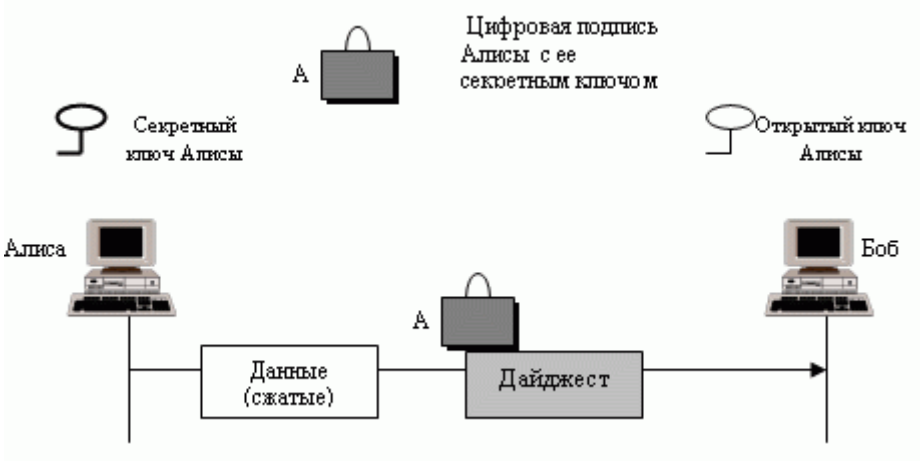


Рисунок 1.49 - Сообщение со сжатым текстом

Конфиденциальность с одноразовым ключом сеанса

Как мы уже говорили раньше, конфиденциальность в почтовой системе может быть достигнута за счет применения обычного шифрования одноразовым ключом сеанса. Алиса может создать ключ сеанса, использовать ключ сеанса для шифрования сообщения и

дайджеста и передать ключ непосредственно с сообщением. Однако для защиты ключа сеанса Алиса зашифровала его открытым ключом Боба. Рисунок 1.49 - показывает ситуацию, когда Боб получает пакет, он сначала расшифровывает ключ, используя свой секретный ключ, чтобы удалить этот секретный ключ. Затем он использует ключ сеанса, чтобы расшифровать остальную часть сообщения. После расширения (декомпрессации) остальной части сообщения Боб создает дайджест сообщения и проверяет, равен ли он дайджесту, передаваемому Алисой. Если дайджесты равны, то сообщение подлинно (рис. 1.50).

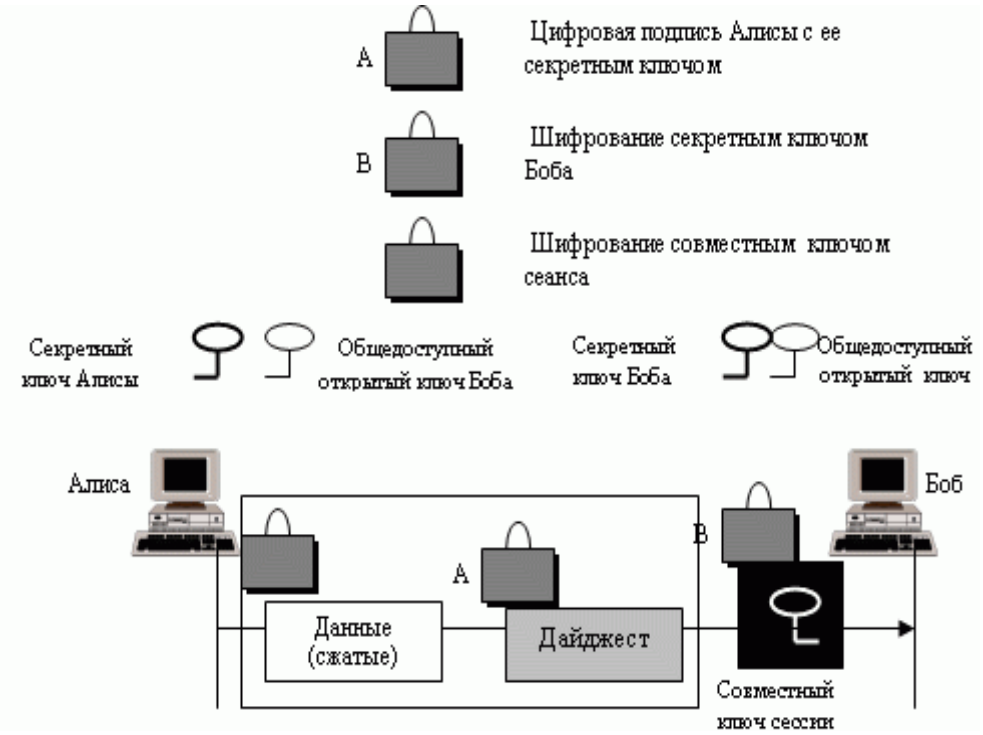


Рисунок 1.50 - Конфиденциальное сообщение

Преобразование кода

Другая услуга, предоставляемая *PGP*, - преобразование кода. Большинство почтовых систем позволяет передать сообщение, состоящее только из символов ASCII. Чтобы перевести символы, не входящие в множество ASCII, *PGP* использует преобразование *Radix-64*. Каждый посылаемый символ (после того как будет зашифрован) преобразуется в код *Radix-64*, который мы обсудим позже в этой лекции.

Сегментация

PGP позволяет сегментацию сообщения после того, как оно было преобразовано к *Radix-64*, чтобы сделать каждый переданный модуль одинаковым по размеру, в соответствии с основным почтовым протоколом.

Кольца ключей

Во всех предыдущих сценариях мы предполагали, что Алиса должна передать сообщение только Бобу. Но это не единственный вариант передачи. Возможно, Алиса должна передать сообщения многим людям; ей нужны *кольца ключей*. В этом случае Алиса нуждается в кольце открытых ключей, включающем ключ(и), принадлежащий каждому человеку, которому Алиса должна передать или от которого может получить сообщение. *PGP* -разработчики определили *кольцо частных/открытых ключей*. Алиса может иметь причины время от времени изменить пару ключей. Другой случай: Алиса, возможно, желает, чтобы ее ключи соответствовали различным группам людей (друзья, коллеги и так далее). Поэтому каждый пользователь должен иметь два множества колец: *кольцо частных/общедоступных ключей* и *кольца общедоступных ключей* других людей. Рисунок 1.51 - показывает сообщество четырех человек, где каждый имеет *кольцо пары частных/открытых ключей* и, в то же самое время, изображены кольца открытых ключей, принадлежащих другим людям в этом сообществе.

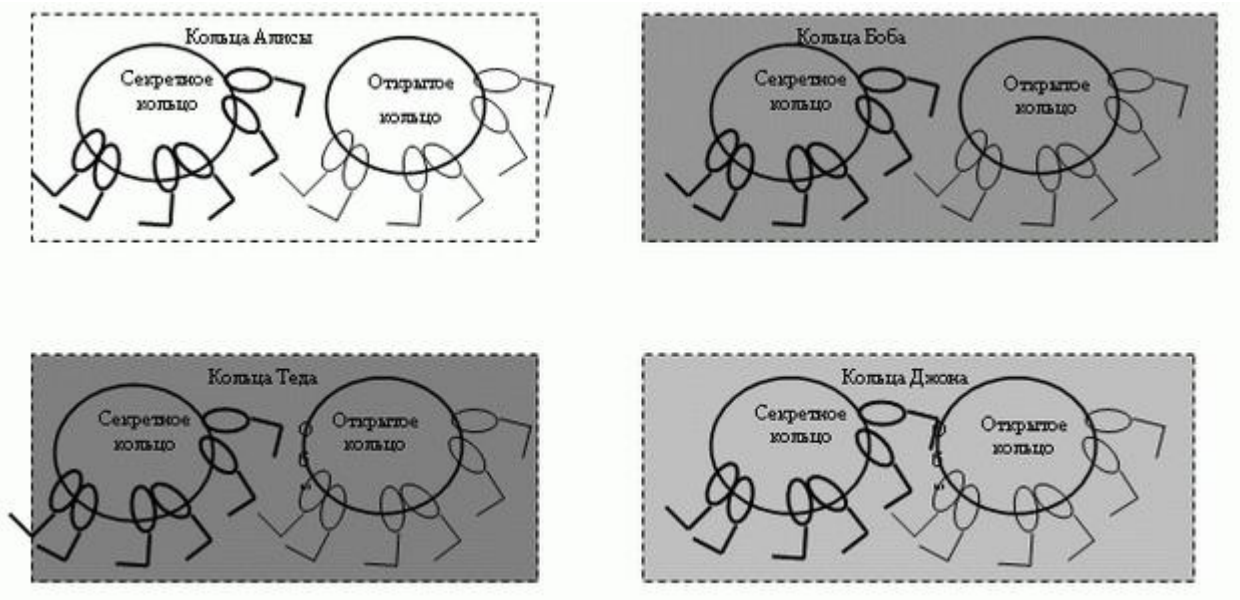


Рисунок 1.51 - Кольца ключей в PGP

Алиса, например, имеет несколько пар частных/открытых ключей, принадлежащих ей, и открытые ключи, принадлежащие другим людям. Обратите внимание, что каждый может иметь больше чем один открытый ключ. Могут возникнуть два случая.

1. Алиса должна передать сообщение другому человеку в сообществе.
   - Она использует свой секретный ключ, чтобы подписать дайджест.
   - Она использует открытый ключ приемника, чтобы зашифровать недавно созданный ключ сеанса.
   - Она шифрует сообщение и подписанный дайджест с созданным ключом сеанса.
2. Алиса получает сообщение от другого человека, состоящего в этом сообществе.
   - Она использует свой секретный ключ, чтобы расшифровать ключ сеанса.
   - Она использует свой ключ сеанса, чтобы расшифровать сообщение и дайджест.
   - Она использует свой открытый ключ, чтобы проверить дайджест.

PGP **-алгоритмы**

В *PGP* используются нижеследующие алгоритмы.

Алгоритмы открытого ключа. Алгоритмы открытого ключа, которые применяются, чтобы подписать дайджесты или зашифровать сообщения, перечислены в Таблица 1.8.

Таблица 1.8. Алгоритмы общедоступного ключа

ID	Описание
1	RSA (для шифрования и подписи)
2	RSA (только для шифрования)
3	RSA (только для подписи)
16	Эль-Гамаль (только для шифрования)
17	DSS
18	Зарезервировано для эллиптической кривой
19	Зарезервировано для ECDSA
20	Эль-Гамаль (для шифрования и подписи)
21	Зарезервировано для Диффи-Хеллмана
100-110	Секретные алгоритмы

Алгоритмы симметричного ключа. Алгоритмы симметричного ключа, которые используются для удобного шифрования, показаны в Таблице 1.9.

Таблица 1.9. Алгоритмы с симметричными ключами

ID	Описание
0	Не зашифровано
1	IDEA
2	Тройной DES
3	CAST-128
4	Blowfish
5	SAFER-SK128
6	Зарезервировано для DES/SK
7	Зарезервировано для AES-128
8	Зарезервировано для AES-192
9	Зарезервировано для AES-256
100-110	Частные алгоритмы

Хэш-алгоритмы. Хэш-алгоритмы, которые используются для создания хэша в PGP, показаны в Таблице 1.10.

Таблица 1.10. Хэш-алгоритмы

ID	Описание
1	MD5
2	SHA-1
3	RIPE-MD/160
4	Зарезервировано для SHA двойной ширины
5	MD2
6	TIGER/192
7	Зарезервировано для HAVAL
100-110	Частные алгоритмы

Алгоритмы сжатия. Алгоритмы сжатия, которые используются для сжатия текста, показаны в Таблице 1.11.

Таблица 1.11. Методы сжатия

ID	**Описание**
0	Без сжатия
1	ZIP
2	ZLIP
100-110	Частные методы

PGP-сертификаты

PGP, подобно другим протоколам, как мы видели до сих пор в других системах, использует сертификаты, чтобы подтвердить подлинность открытых ключей. Однако в этом случае процесс полностью отличается от уже рассмотренных

Сертификаты X.509

Протоколы, которые используют сертификаты X.509, зависят от иерархической структуры доверия. Есть заранее заданная цепочка доверия от корня до любого сертификата. Каждый пользователь полностью доверяет администрации CA на заданном уровне корня. Корень вырабатывает сертификаты для второго уровня, второй уровень CA вырабатывает сертификаты для третьего уровня, и так далее. Каждый партнер, который хочет быть доверенным для выдачи сертификатов, должен быть представлен на одной из ветвей дерева какого-либо CA. Если Алиса не доверяет выпускающему сертификат для Боба, она может обратиться к администрации более высокого уровня вплоть до корня (которому необходимо доверять для того, чтобы система работала). Другими словами, есть один-единственный путь к сертификату от CA, которому полностью доверяют.

В X.509 есть единственный путь от администрации, которому полностью доверяют, к любому сертификату.

PGP-Сертификаты

В *PGP* нет никакой потребности в CA - любой в кольце может подписать сертификат для кого-либо другого в кольце. Боб может подписать сертификат для Теда, Джона, Алисы и так

далее. В *PGP* нет иерархии доверия; нет дерева. Отсутствие иерархической структуры может привести к тому, что Тед может иметь один сертификат от Боба и другой сертификат - от Джона. Если Алиса хочет исследовать линию получения сертификатов для Теда, у нее есть два пути: начинать от Боба и начинать от Джона. Интересно, что Алиса может полностью доверять Бобу, но только частично доверяет Джону. Может быть много путей доверия, приводящих к сертификату от администрации, которой доверяют полностью или частично. В *PGP* выпускающего сертификат обычно называют *поручителем*.

В *PGP* может быть много путей доверия, приводящих к сертификату от администрации, которой доверяют полностью или частично.

Доверие и законность

Полная операция *PGP* базируется на доверии поручителя, доверии сертификата и законности общедоступных ключей.

Уровни доверия поручителя. При отсутствии центральной администрации очевидно, что кольцо не может быть очень большим, если каждый пользователь в кольце *PGP* - пользователей не имеет полного доверия к каждому члену сообщества. (Даже в реальной жизни мы не можем полностью доверять каждому человеку, которого мы знаем.). Чтобы решить эту проблему, *PGP* позволяет различные уровни доверия. Число уровней, главным образом, зависит от реализации, но для простоты давайте назначим три уровня доверия к любому поручителю: *никакой, частичный* и *полный*. Уровень доверия поручителя определяет уровни доверия, выработанные поручителем для других людей в кольце. Например, Алиса может полностью доверять Бобу, частично доверять Анне и не доверять Джону. Вообще, в *PGP* нет механизма, чтобы решить, как принять решение поручителя о заслуживающем доверия партнере; это может сделать только пользователь.

Уровни доверия сертификата, Когда Алиса получает сертификат от поручителя, она хранит сертификат под именем субъекта (сертифицированный объект) и назначает уровень доверия этому сертификату. Уровень доверия сертификата обычно тот же, что и уровень доверия поручителя, который выдал сертификат. Предположим, что Алиса полностью доверяет Бобу, частично доверяет Анне и Джанетт и не имеет никакого доверия Джону. Могут быть следующие сценарии:

1. Боб вырабатывает два сертификата, один для Линды (с открытым ключом K1) и один для Лесли (с открытым ключом K2). Алиса хранит открытый ключ и сертификат для Линды под названием "Линда" и назначает *полный* уровень доверия этому сертификату. Алиса также

хранит сертификат и открытый ключ для Лесли под названием "Лесли" и назначает полный уровень доверия этому сертификату.

2. Анна вырабатывает сертификат для Джона (с открытым ключом K3). Алиса хранит этот сертификат и открытый ключ под названием "Джон", но назначает уровень для этого сертификата - *частичный*.

3. Джанетт вырабатывает два сертификата: один для Джона (с общедоступным ключом K3) и один для Ли (с открытым ключом K4). Алиса хранит сертификат Джона под его именем и сертификатом "Ли" под этим именем (Ли), каждый с *частичным* уровнем доверия. Обратите внимание, что Джон теперь имеет два сертификата: один от Анны и один от Джанетт, каждый с *частичным* уровнем доверия.

4. Джон вырабатывает сертификат для Лиз. Алиса может отказаться или сохранить этот сертификат с надписью " *не доверяю"*.

Законность ключей. Цель использования поручителя и сертификата доверия - определить законность открытого ключа. Алиса должна знать, насколько законны открытые ключи Боба, Джона, Лиз, Анны и так далее. *PGP* определяет очень ясную процедуру для того, чтобы определить законность ключей. Уровень законности ключей для пользователя - это взвешенные уровни доверия пользователя. Например, предположим, что мы назначаем следующие веса на уровни доверия сертификата:

1. вес 0 - сертификату, которому не доверяют;
2. вес 1/2 - сертификату с частичным доверием;
3. вес 1 - сертификату с полным доверием.

Тогда для полного доверия объекту Алиса нуждается в одном сертификате, которому доверяет полностью, или в двух частичных сертификатах доверия для этого объекта. Например, Алиса может использовать открытый ключ Джона в предыдущем сценарии, потому что и Анна, и Джанетт выработали сертификат для Джона, каждый с уровнем доверия сертификата 1/2. Обратите внимание, что законность общедоступного ключа, принадлежащего объекту, не имеет никакого отношения к уровню доверия других людей к этому человеку. Хотя Боб может использовать открытый ключ Джона, чтобы передать сообщение ему, Алиса может не принять ни одного сертификата, выпущенного Джоном, потому что для Алисы Джон имеет уровень " *нет доверия"*.

Старт кольца

Вы, возможно, нашли серьезную проблему в вышеупомянутом обсуждении. А если никто не передал сертификат, что он полностью или частично доверяет объекту? Например, как можно решить проблему законности общедоступного ключа Боба, если никто не передал сертификат Боба? В *PGP* законность ключа, которому доверяют, или объекта, которому частично доверяют, может быть также определена другими методами.

1. Алиса может физически получить открытый ключ Боба. Например, Алиса и Боб могут встретиться лично. И при этом обменяться открытым ключом, написанным на обрывке бумажки или на диске.
2. Если голос Боба распознаваем Алисой, Алиса может по телефону вызвать его и получить его открытый ключ.
3. Лучшее решение, предложенное *PGP* для Боба, - передать его открытый ключ Алисе *электронной почтой*. И Алиса, и Боб делают дайджест ключа 16 байтов MD5 (или 20 байтов SHA-1). Дайджест обычно отображается как восемь групп по 4 цифры (или десять групп по 4 цифры) в шестнадцатеричном виде и называется *отпечатком пальца*. Алиса может тогда вызвать Боба и проверить *отпечаток пальца* по телефону. Если ключ изменился или изменен во время почтовой передачи, два отпечатка пальца не будут соответствовать друг другу. Для того чтобы сделать проверку более удобной, *PGP* создал список слов, каждое из которых представляет комбинацию из 4-х цифр. Когда Алиса вызывает Боба, Боб может объявить эти восемь слов (или десять слов) для Алисы. Слова тщательно выбраны *PGP*, чтобы избежать путаницы при произношении; например, если *печь* находится в списке, то слово *речь* не включается в список.
4. В *PGP* ничто не препятствует Алисе получать открытый ключ Боба от CA по отдельной процедуре. Она может тогда вставить открытый ключ в *кольцо открытого ключа*.

Таблица кольца ключей

Каждый пользователь, такой как Алиса, сохраняет множество двух колец ключей: одно *кольцо секретного ключа* и одно *кольцо открытого ключа*. *PGP* определяет структуру для каждого из этих *колец ключей* в форме таблицы.

Секретное кольцо

ID Пользователя	ID Ключа	Открытый ключ	Секретный ключ	Метка времени
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.

Рисунок 1.52 - Формат таблицы кольца секретного ключа

Рисунок 1.52 показывает формат таблицы кольца секретного ключа.

- **Пользовательский ID**. Пользовательский ID - обычно адрес *электронной почты* пользователя. Однако пользователь может определять уникальный адрес *электронной почты* или псевдоним для каждой ключевой пары. Таблица перечисляет пользовательские ID, связанные с каждой парой.
- **ID ключа**. Этот столбец уникально определяет открытый ключ среди открытых ключей пользователя. В *PGP* ID ключа для каждой пары - первые (самые младшие) 64 бита открытого ключа. Другими словами, ключ ID вычисляется как - ключ mod 2^{64}). Ключ ID необходим для работы *PGP*, потому что Боб может иметь несколько открытых ключей, принадлежащих Алисе, в его кольце открытого ключа. Когда он получает сообщение от Алисы, Боб должен знать, какой ID ключа надо использовать, чтобы проверить сообщение. Ключ ID, который передают с сообщением, как мы увидим, дает возможность Бобу использовать заданный открытый ключ для Алисы из своего открытого кольца. Можно спросить, почему не передают полностью открытый ключ. Ответ на это вопрос такой: в криптографии открытого ключа размер открытого ключа может быть очень большой. Передавая только 8 байт, мы уменьшаем размер сообщения.
- **Открытый ключ**. Этот столбец только перечисляет открытые ключи, принадлежащие конкретной паре "секретный ключ / открытый ключ".
- **Зашифрованный секретный ключ**. Этот столбец показывает зашифрованное значение секретного ключа в паре "секретный ключ / открытый ключ". Хотя Алиса - единственный человек, имеющий доступ к своему секретному кольцу, *PGP* сохраняет только зашифрованную версию секретного ключа. Мы увидим позже, как зашифровывается и расшифровывается секретный ключ.
- **Метка времени**. Этот столбец содержит время и дату создания пары ключей. Это помогает пользователю решать, когда произвести чистку старых пар и когда создавать новые пары.

Модель доверия в PGP

Как предложил Циммерман, мы можем создать модель доверия для любого пользователя в кольце, с пользователем в качестве центра активности. Такая модель может выглядеть как показано на рис. 1.53. Рисунок показывает модель доверия для Алисы в некоторый момент. Диаграмма может измениться с любыми изменениями таблицы кольца общедоступного ключа.

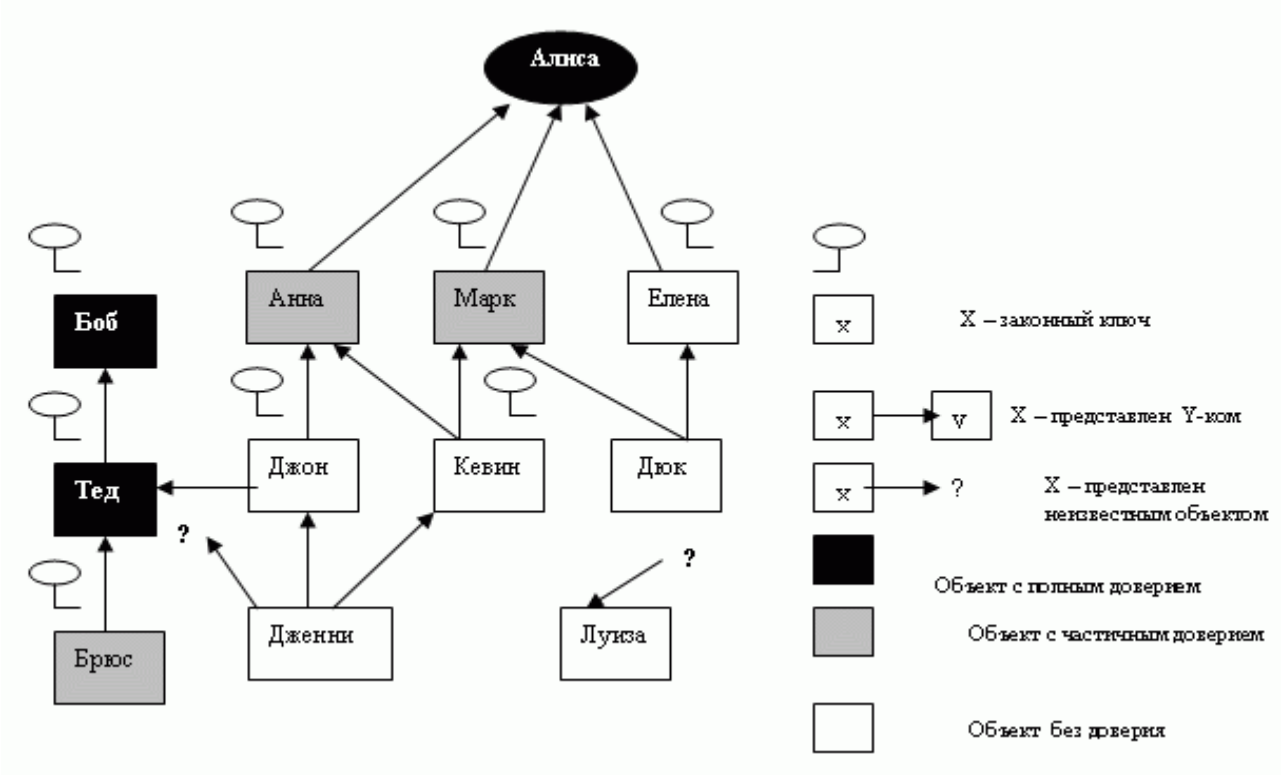


Рисунок 1.53 - Модель доверия

Рассмотрим рисунок. Здесь показаны три объекта с полным доверием в кольце Алисы (сама Алиса, Боб и Тед) и три объекта с частичным доверием (Анна, Марк и Брюс). Есть также шесть объектов без доверия. Девять объектов имеют законный ключ. Алиса может зашифровать сообщение к любому из этих объектов или проверить, что подпись получена от одного из этих объектов (ключ Алисы никогда не используется в этой модели). Есть также три объекта, которые не имеют никаких законных ключей с Алисой.

Боб, Анна и Марк сделали свои ключи законными, посылая их *электронной почтой* и подтверждая отпечатками пальцев по факсу. Елена передала сертификат от CA, потому что ей не доверяет Алиса и проверка по телефону невозможна. Хотя Теду полностью доверяют, он дал Алисе сертификат, подписанный Бобом. Джон передал Алисе два сертификата, один -

подписанный Тедом и один - Анной. Кевин передал два сертификата Алисе, один - подписанный Анной и один - Марком. Каждое из этих удостоверений дает Кевину половину уровня законности; поэтому ключ Кевина законен. Дюк передал два удостоверения Алисе, одно - подписанное Марком и другое - Еленой. Так как Марку "полудоверяют", а Елене не доверяют, Дюк не имеет законного ключа. Дженни передала четыре сертификата, один - подписанный объектом, которому "полудоверяют", два - незаконными объектами и один - неизвестным объектом. Дженни имеет не много шансов, чтобы ее ключ признали законным. Луиза передала один сертификат, подписанный неизвестным объектом. Заметим, что Алиса может сохранить имя Луизы в таблице в случае, если в будущем прибудет сертификат для Луизы.

Сеть доверия

PGP может в конечном счете сделать *сеть доверия* между группой людей. Если каждый объект вводит все больше и больше объектов другим объектам, *кольцо открытого ключа* для каждого объекта становится большим и большим, и объекты в кольце могут тогда передавать безопасную *электронную почту* друг другу.

Аннулирование ключей

Аннулирование ключей может стать необходимым для объекта, если надо отменить его или ее общедоступный ключ в кольце. Это может случиться, если владелец ключа чувствует, что ключ скомпрометирован (например, украден) или слишком старый, чтобы быть безопасным. Чтобы отменять ключ, владелец может передать сертификат аннулирования, подписанный им самим. Сертификат аннулирования должен быть подписан старым ключом и распространен всем людям в кольце, которые использовали общедоступный ключ.

Распаковка информации из колец

Как мы видели, передатчик и приемник каждый имеет два кольца ключей: один частный и один открытый. Давайте посмотрим, какая информация нужна для того, чтобы послать и получить сообщение, извлеченное из этих колец.

Сторона передатчика

Предположим, что Алиса посылает *электронную почту* Бобу. Алиса нуждается в пяти единицах информации: ключ ID открытого ключа, который она использует, ее секретный ключ, ключ сеанса, ID открытого ключа Боба и открытый ключ Боба. Для того чтобы получить эти пять единиц информации, Алиса должна передать *PGP* четыре единицы информации: ее

пользовательский ID (для *электронной почты*), ее фразу-пароль (последовательность нажатия клавиш с возможными паузами) и пользовательский ID Боба (см. рис. 1.54).

ID открытого ключа Алисы (полученный с сообщением) и ее секретный ключ (для подписи сообщения) хранятся в таблице кольца секретного ключа. Алиса выбирает пользовательский ID (user ID - ее адрес *электронной почты*), который она хочет использовать как индекс к этому кольцу. *PGP* извлекает ключ ID и зашифрованный секретный ключ. *PGP* использует заранее заданный алгоритм дешифрования и ее хэширования, фразу-пароль (как ключ), чтобы расшифровать этот секретный ключ.

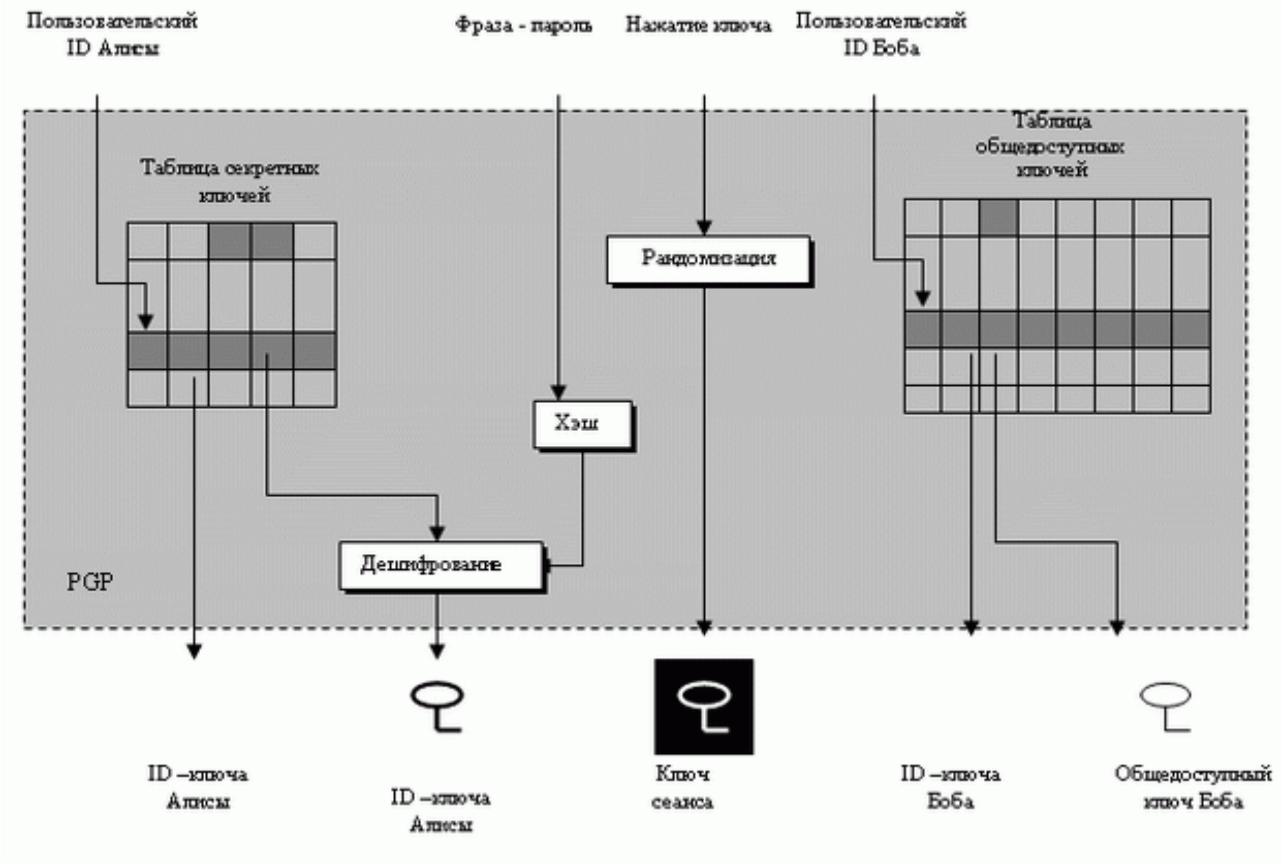


Рисунок 1.54 - Распаковка информации из колец на стороне передатчика

Алиса также нуждается в секретном ключе сеанса. Ключ сеанса в *PGP* - случайное число с размером, который определен в алгоритме шифрования/дешифрования. *PGP* использует генератор случайных чисел, чтобы создать случайный ключ сеанса; начальное число - множество произвольных нажатий клавиши, напечатанных Алисой на ее клавиатуре. Каждое нажатие клавиши преобразовано в 8 бит, а каждая пауза между нажатиями клавиши преобразована в 32 бита. Комбинация проходит сложный генератор случайных чисел, чтобы создать очень достоверное случайное число как ключ сеанса. Обратите внимание, что ключ

сеанса в *PGP* - одноразовый случайный ключ (см. приложение K), используемый только один раз.

Алиса также нуждается в ID ключа Боба (чтобы послать его с сообщением) и в открытом ключе Боба (чтобы зашифровать ключ сеанса). Эти две части информации извлекаются из таблицы кольца открытых ключей, используя пользовательское ID Боба (его адрес *электронной почты*).

Сторона приемника

На стороне приемника Бобу нужны три части информации: секретный ключ Боба (расшифровывает ключ сеанса), ключ сеанса (чтобы расшифровать данные) и ключ Алисы (чтобы проверить подпись) (см. рис. 1.55).

Боб использует ID своего открытого ключа, передаваемый Алисой, чтобы определить, что его соответствующий секретный ключ может расшифровать ключ сеанса. Эта часть информации может быть извлечена из таблицы кольца секретного ключа Боба. Секретный ключ, однако, зашифрован при записи в память. Боб должен использовать фразу-пароль и хэш-функцию, чтобы расшифровать ключ.

Зашифрованный ключ сеанса передали с сообщением; Боб применяет свой расшифрованный секретный ключ, чтобы расшифровать ключ сеанса.

Боб использует ID ключа Алисы, переданный с сообщением, чтобы извлечь открытый ключ Алисы, который хранится в таблице кольца открытого ключа Боба.

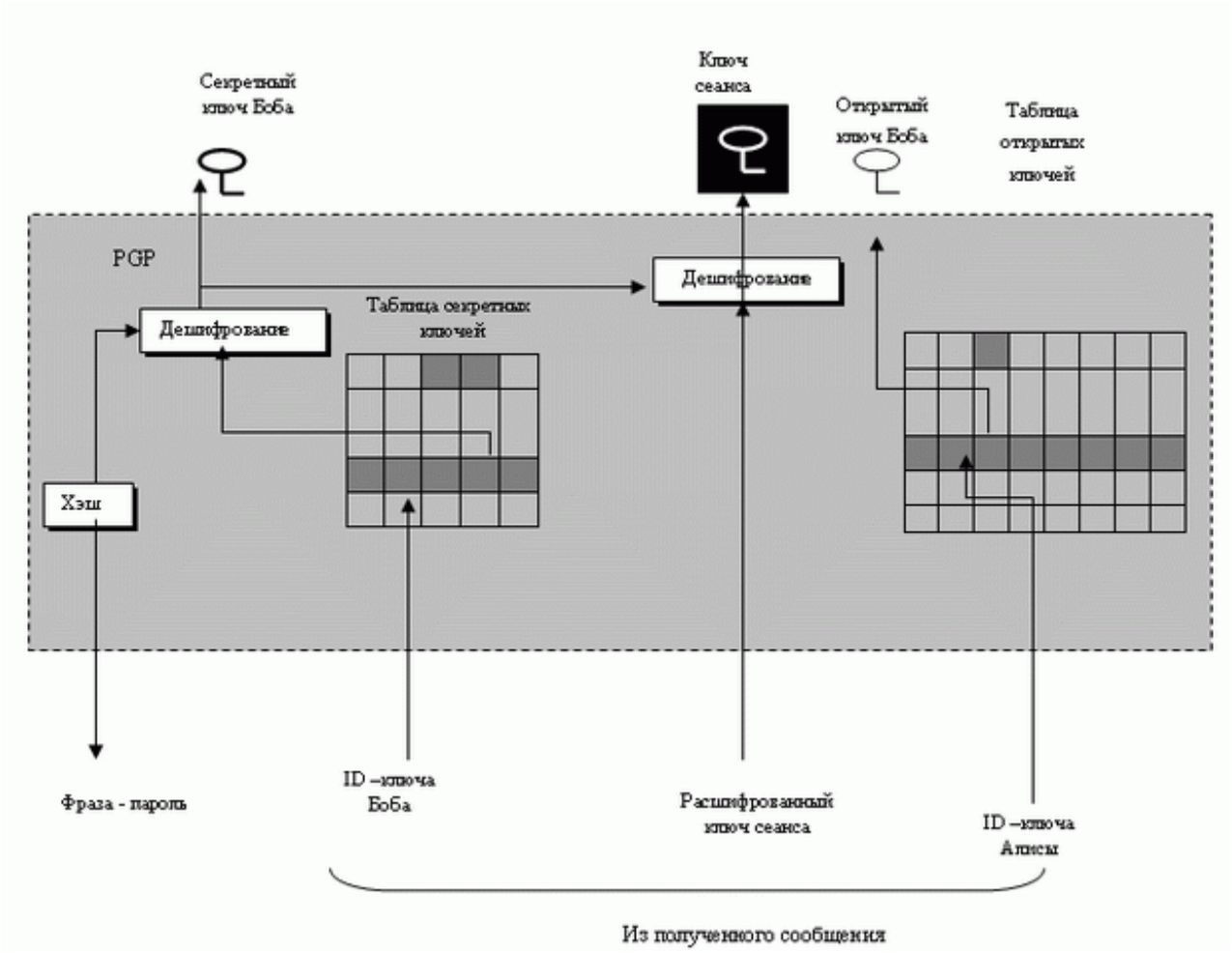


Рисунок 1.55 - Распаковка информации из колец на стороне приема

PGP-пакеты

Сообщение в *PGP* состоит из одного или более пакетов. В течение эволюции *PGP* формат и число типов пакета изменились. Подобно другим протоколам, известным до сих пор, *PGP* имеет типовой заголовок, который применяется к каждому пакету. Типовой заголовок нынешней версии имеет только два поля, как показано на рис. 1.56.

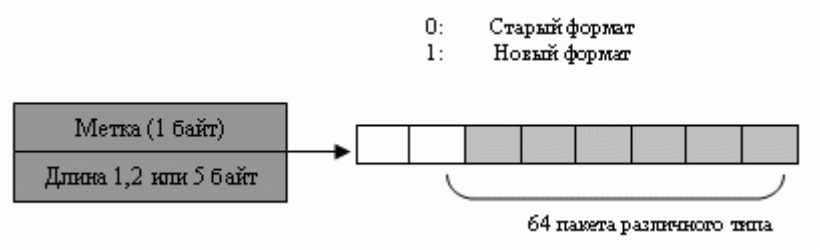


Рисунок 1.56 - Формат заголовка пакета

- **Метка (тег)**. Нынешний формат для этого поля определяет метку как флажок на 8 битов; первый бит (самый старший) - всегда 1. Второй бит - 1, если мы используем последнюю

версию. Остающиеся шесть бит могут определить до 64 различных типов пакетов, как показано в табл 1.12.

- **Длина**. Поле длины определяет длину полного пакета в байтах. Размер этого поля является переменным; он может быть 1, 2 или 5 байтов.

Таблица 1.12. Некоторые обычно используемые типы пакетов

Значение	***Тип пакета***
1	Ключ сеанса, зашифрованный открытым ключом
2	Пакет подписи
5	Пакет секретного ключа
6	Пакет открытого ключа
8	Пакет сжатых данных
9	Пакет данных, зашифрованный секретным ключом
11	Пакет литеральных (буквенных данных)
13	Пакет пользовательского ID

Приемник может определить число байтов поля длины, на основании значения байта, идущего сразу после поля метки.

- Если значение байта после поля метки меньше, чем 192, то поле длины - только один байт. Длина текстового блока (пакет минус заголовок) вычисляется как

длина текстового блока = первый байт

- Если значение байта после поля метки между 192 и 223 (включая крайние значения), то поле длины - два байта. Длина текстового блока может быть вычислена как

длина текстового блока = (первый байт - 192) << 8 + второй байт + 192

- Если значение байта после поля метки между 224 и 254 (включая крайние значения), то поле длины - один байт. Этот тип поля длины определяет только длину части текстового блока (частичная длина текстового блока). Частичная длина текстового блока может быть вычислена как

Длина текстового блока = второй байт <<24 | третий байт << 16| четвертый байт << 8 |пятый байт

Обратите внимание, что формула означает 1 2 (первый байт & 0x1F). Степень - это фактически значение пяти самых правых битов. Поскольку поле - между 224 и 254 включительно значение пяти самых правых битов - между 0 и 30 включительно. Другими словами, частичная длина текстового блока может быть между единицей (2^0) и 1 073 741 824 (2^{30}). Когда пакет представлен несколькими частичными текстовыми блоками, применима частичная длина текстового блока. Каждая частичная длина текстового блока определяет одну часть длины. Последнее поле длины не может быть частичной длиной текстового блока созданного сообщения. Например, если пакет имеет четыре части, он может иметь три частичных поля длины и одно поле длины другого типа.

- Если значение байта после поля метки - 255, то поле длины состоит из пяти байтов. Длина текстового блока вычисляется как

Пакет литеральных данных. Пакет буквенных данных переносит или содержит текущие данные, которые передаются или сохраняются. Этот пакет - самый элементарный тип сообщения; то есть, он не может нести никакой другой пакет. Формат пакета показан на рис. 1.57.

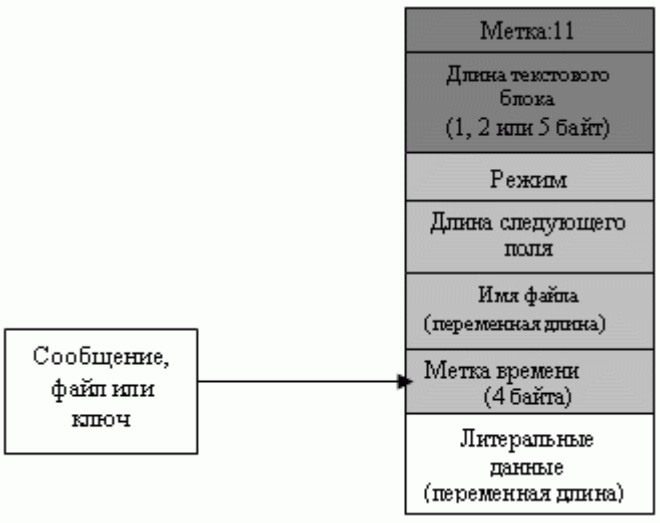


Рисунок 1.57 - Пакет буквенных данных

Режим. Это однобайтовое поле определяет, как данные написаны в пакете. Значение этого поля может быть "b" для двоичных данных, "t" - для текста, или любое другое значение, определенное для собственных целей.

Длина следующего поля. Это однобайтовое поле определяет длину следующего поля (поля имени файла).

Имя файла. Это поле переменной длины определяет название файла или сообщения в виде строки ASCII.

Метка времени. Это четырехбайтовое поле определяет время создания или последней модификации сообщения. Значение может быть 0 - это означает, что пользователь выбирает опцию "не определять время".

Литеральные данные. Это поле переменной длины, переносящее фактические данные (файл или сообщение) в тексте или двоичном виде (в зависимости от значения поля режима).

Сжатый пакет данных. Этот пакет, переносящий пакеты сжатых данных. Рис. 1.58 показывает формат пакет сжатых данных.

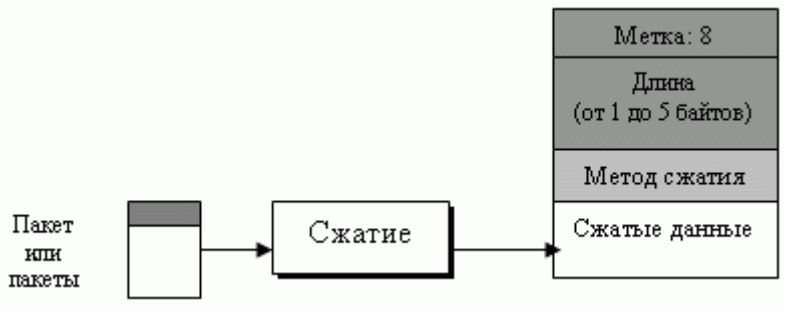


Рисунок 1.58 - Пакет сжатых данных

- **Метка (метод сжатия)**. Это однобайтовое поле определяет метод сжатия, используемый для сжатия данных (следующее поле). Значения, определенные для этого поля, пока 1 (ZIP) и 2 (ZLIP). Также реализация может применять другие экспериментальные методы сжатия. Метод ZIP обсуждается в приложении M.
- **Сжатые данные**. Это поле переменной длины переносит данные после сжатия. Обратите внимание, что в этом поле может быть один пакет данных или последовательное соединение двух или более пакетов. Общая ситуация - единственный пакет литеральных данных или комбинация пакета подписи, сопровождаемого пакетом литеральных данных.

Пакет данных, зашифрованных ключом засекречивания. Этот пакет переносит данные от одного пакета или комбинации пакетов, которые были зашифрованы, с использованием обычного алгоритма с симметричными ключами. Обратите внимание, что

пакет, несущий одноразовый ключ сеанса, передается перед этим пакетом. Рис. 1.59 показывает формат пакета зашифрованных данных.

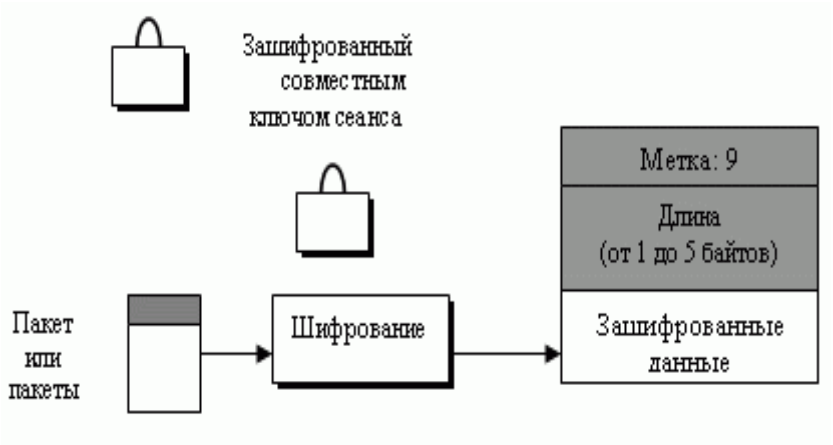


Рисунок 1.59 - Пакет зашифрованных данных

Пакет подписи.Пакет подписи мы уже обсуждали раньше, когда рассматривали защиту целостности данных. Рис. 1.60 показывает формату пакета подписи.

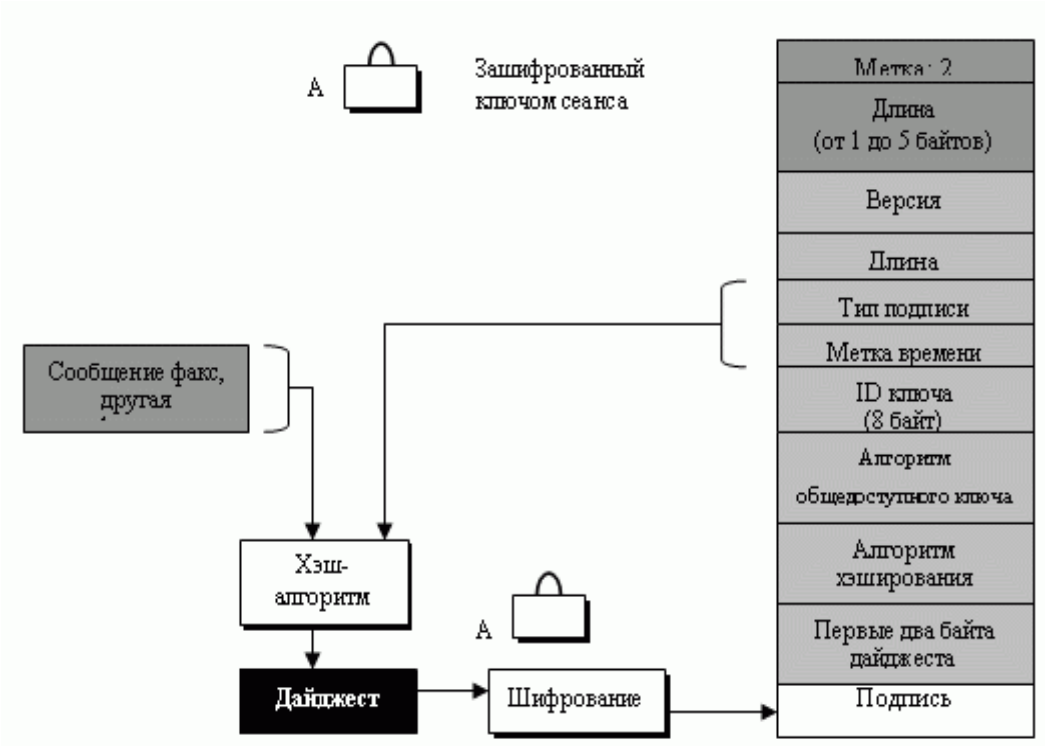


Рисунок 1.60 - Пакет подписи

Таблица 1.13. Некоторые значения подписи

Значение	***Подпись***
0x00	Подпись двоичного документа (сообщение или файл)
0x01	Подпись текстового документа (сообщение или файл)
0x10	Общий сертификат пользовательского ID и пакета открытого ключа. Подписывающее лицо не указывает никаких данных о владельце ключа
0x11	Персональный сертификат пользовательского ID и пакет открытого ключа. Не проводится верификация владельца ключа
0x12	Случайный сертификат пользовательского ID и пакет открытого ключа. Некоторая случайная верификация владельца ключа
0x13	Положительный сертификат пользовательского ID и пакет открытого ключа. Делается существенная верификация
0x30	Подпись аннулирования сертификата. Она удаляет более ранний сертификат (от Ox10 до Ox13)

- **Версия**. Это однобайтовое поле определяет используемую версию *PGP*.
- **Длина**. Это поле сначала было выделено для того, чтобы показать длину следующих двух полей, но теперь размер этих полей установлен, поэтому значение этого поля равно 5.
- **Тип подписи**. Это однобайтовое поле определяет цель подписи. Оно документирует подпись. табл. 1.13 показывает некоторые типы подписи.
- **Метка времени**. Это четырехбайтовое поле, которое определяет время, когда подпись была вычислена.
- **Ключ ID**. Это восьмибайтовое поле определяет ID открытого ключа подписывающего лица. Оно указывает верификатору, какой открытый ключ подписывающего лица должен быть использован, чтобы расшифровать дайджест.
- **Алгоритм открытого ключа**. Это однобайтовое поле дает код для алгоритма открытого ключа, который применялся для шифрования дайджеста. Верификатор использует тот же самый алгоритм, чтобы расшифровать дайджест.
- **Алгоритм хэширования**. Это однобайтовое поле дает код для алгоритма хэширования, обычно создает дайджест.

- **Первые два байта дайджеста сообщения**. Эти два байта используются как своего рода контрольная сумма. Они гарантируют, что приемник использует правильный ключ ID, чтобы расшифровать дайджест.
- **Подпись**. Это поле переменной длины. Оно содержит зашифрованный дайджест, подписанный передатчиком.

Пакет ключа сеанса, зашифрованный открытым ключом.

Этот пакет используется, чтобы передать ключ сеанса, зашифрованный открытым ключом приемника. Формат пакета показан на рис. 1.61.

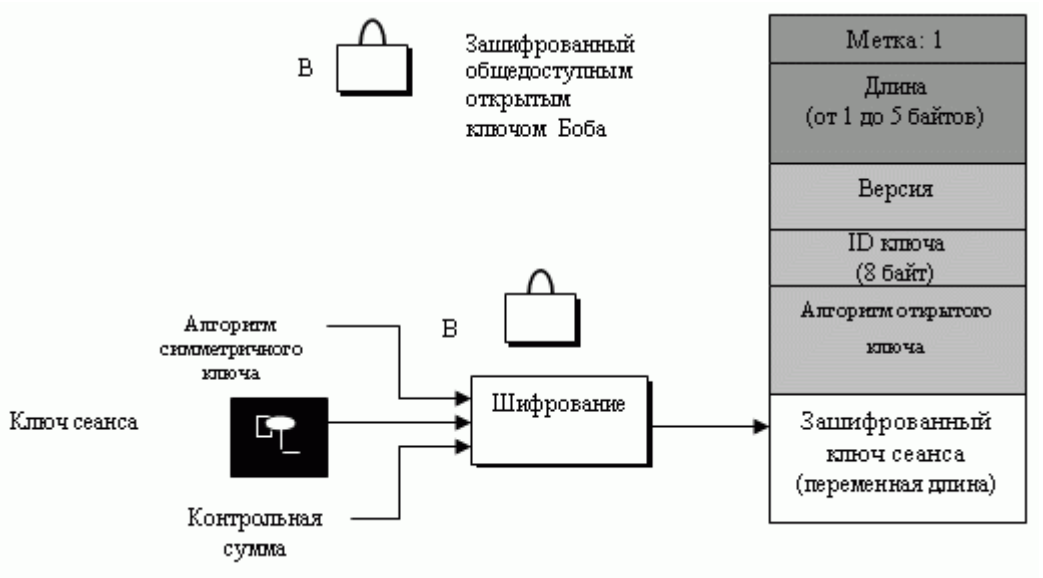


Рисунок 1.61 - Пакет ключа сеанса

- **Версия**. Это однобайтовое поле определяет используемую версию *PGP*.
- **Ключ ID**. Это восьмибайтовое поле определяет ID общедоступного ключа передатчика. Он указывает приемнику, какой общедоступный ключ передатчика должен использоваться, чтобы расшифровать ключ сеанса.
- **Алгоритм открытого ключа**. Это однобайтовое поле дает код для алгоритма открытого ключа, использованного для шифрования ключа сеанса. Приемник применяет тот же самый алгоритм, чтобы расшифровать ключ сеанса.
- **Сеанс шифрования**. Это область переменной длины, которая содержит зашифрованное значение ключа сеанса, созданного отправителем и посланного приемнику. Шифрование основано на следующих средствах:
  - симметричный алгоритм шифрования с одним октетом;

- ключ сеанса;
- контрольная сумма с двумя октетами равняется сумме октетов ключей предыдущих сеансов.

Пакет открытого ключа.Этот пакет содержит общедоступный ключ отправителя. Формат пакета показан на рис. 1.62.

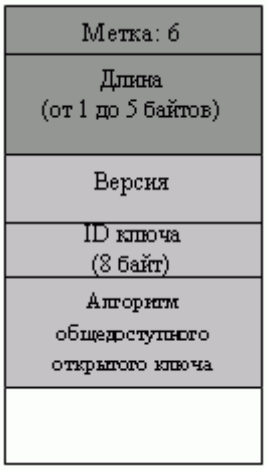


Рисунок 1.62 - Пакет открытого ключа

- **Версия**. Эта однобайтовая область определяет используемую версию *PGP*.
- **Метка времени**. Эта четырехбайтовая область определяет время, когда был создан ключ.
- **Законность**. Эта двухбайтовая область показывает число дней, в продолжении которых ключ является действительным. Если значение равно 0, это означает, что действие ключ не заканчивается.
- **Алгоритм открытого ключа**. Эта однобайтовая область дает код для алгоритма общедоступного ключа.
- **Открытый ключ**. Эта область переменной длины содержит общедоступный ключ. Его содержание зависит от алгоритма общедоступного ключа.

Пакет пользовательского ID.Этот пакет идентифицирует пользователя и обычно связывает пользователя и содержание с открытым ключом передатчика. рис. 1.63 показывает формат пакет пользовательского ID. Обратите внимание, что поле длины общего заголовка - только один байт.

Рисунок 1.63 - Пакет пользовательского ID

- **Пользовательский ID**. Эта строка переменной длины определяет пользовательский ID передатчика. Это обычно имя пользователя, сопровождаемое адресом *электронной почты*.

PGP-сообщения

Сообщение в *PGP* - комбинация упорядоченных и/или вложенных пакетов. Даже притом, что не все комбинации пакетов могут составлять сообщение, список комбинаций пакетов достаточно длинный. В этой секции мы приведем несколько примеров, чтобы проиллюстрировать идею.

Зашифрованное сообщение

Зашифрованное сообщение может быть последовательностью двух пакетов: пакета ключа сеанса и симметрично зашифрованного пакета. Последний обычно представляет собой вложенный пакет. рис. 1.64 показывает такую комбинацию.

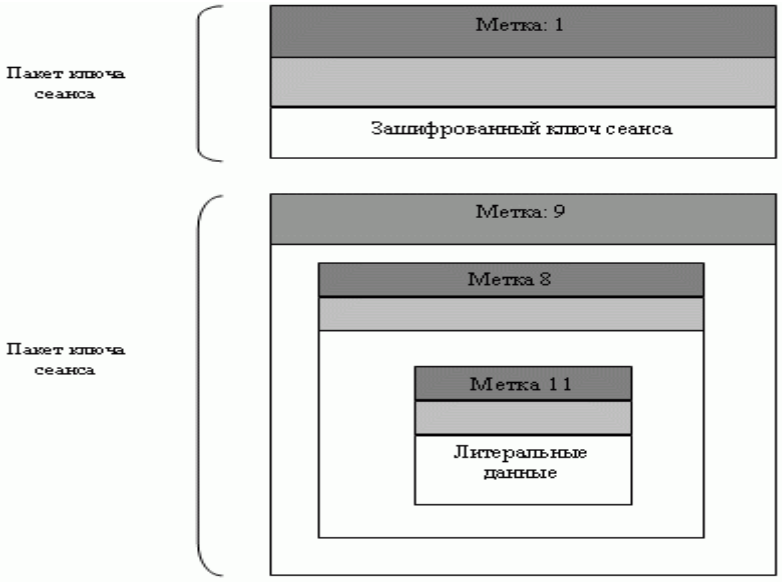


Рисунок 1.63 - Зашифрованное сообщение

Обратите внимание, что пакет ключа сеанса содержит только единственный пакет. Зашифрованный пакет данных состоит из сжатого пакета. Сжатый пакет состоит из пакета литеральных данных. Последний содержит литеральные данные.

Подписанное сообщение

Подписанное сообщение может быть комбинацией пакета подписи и литерального пакета, как это показано на рис. 1.65.

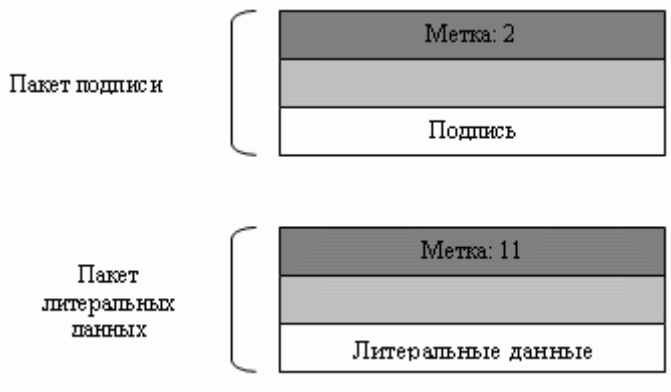


Рисунок 1.65 - Подписанное сообщение

Сертифицирующее сообщение

Хотя сертифицирующее сообщение может принимать множество форм, один простой пример - комбинация пользовательского ID пакета и пакета открытого ключа, как показано на рис. 1.66. Подпись здесь вычислена для последовательного соединения ключевого и пользовательского ID.

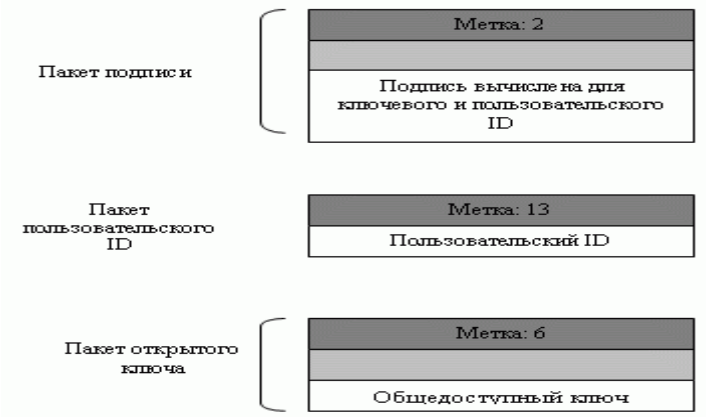


Рисунок 1.66 - Сертифицирующее сообщение

Приложения PGP

PGP широко применялся для персональной *электронной почты*. Это использование, вероятно, продолжится.

Итоги

- Поскольку при использовании электронной почтовой связи отсутствует сеанс, передатчик сообщения должен включить в сообщение названия или идентификаторы алгоритмов, используемых в *электронной почте*. В *электронной почте* шифрование/дешифрование делается при помощи алгоритма с симметричными ключами, но при этом ключ засекречивания для расшифровки сообщения зашифрован открытым ключом приемника и передается с сообщением.
- Первый протокол, рассмотренный в этой лекции, называется *"Очень хорошей конфиденциальностью" (PGP)*. Он был изобретен Филом Циммерманом для обеспечения *электронной почты* услугами секретности, целостности и установления подлинности. *PGP* может использоваться для того, чтобы создать безопасное почтовое сообщение или надежно хранить файл для будущего чтения.
- В *PGP* Алиса нуждается в кольце открытых ключей для каждого человека, с которым она переписывается. Она также нуждается в кольце принадлежащих ей частных/открытых ключей.
- В *PGP* не нужны Центры Сертификации; любой в кольце может подписать сертификат для кого-либо еще в этом кольце. В *PGP* нет иерархии доверия; нет дерева иерархии. Может быть много путей от администрации, которой полностью или частично доверяют, к любому объекту.
- Работа *PGP* базируется на доверии поручителя, уровне доверия и законности открытых ключей. *PGP* организует *сеть доверия* между группами людей.
- *PGP* определил несколько типов пакетов: литеральных данных, сжатый пакет данных, пакет данных, зашифрованный ключом засекречивания, пакет подписи, пакет ключа сеанса, зашифрованный открытым ключом, пакет открытого ключа и пользовательский ID пакета.

Контрольные вопросы

1. Объясните, как Боб, когда он получает *PGP* -сообщение от Алисы, узнает, какие криптографические алгоритмы она использовала.
2. Объясните, как Боб и Алиса в *PGP* обмениваются секретным ключом шифрования сообщений.

3. Объясните, как Боб и Алиса обмениваются секретным ключом шифрования сообщений.
4. Назовите три типа сообщений в *PGP* и объясните их цели.
5. В *PGP* в почтовом сообщении можно использовать два различных алгоритма открытого ключа для шифрования и подписи. Как это определяется в сообщении, передаваемом от Алисы к Бобу?
6. Какие типы пакетов нужно передать в *PGP*, чтобы обеспечить следующие услуги безопасности:
    1. Конфиденциальность
    2. Целостность сообщения
    3. Определение подлинности

1.2. КОМПЬЮТЕРНЫЙ ПРАКТИКУМ ПО СЕТЕВЫМ ПРОТОКОЛАМ

Программная реализация

Для создания защищенного соединения необходимо получить сертификат или создать его самому. Создать SSL сертификат самому можно средствами свободно распространяемого пакета IIS6 Resource Kit Tools. Может потребоваться установка служб. Для этого в Панели управления необходимо выбрать раздел «Программы и компоненты» и в открывшемся окне в меню слева нажать на ссылку «Включение или отключение компонентов Windows». Далее в открывшемся окне нужно включить компонент «Службы IIS» (рис. 1.67).

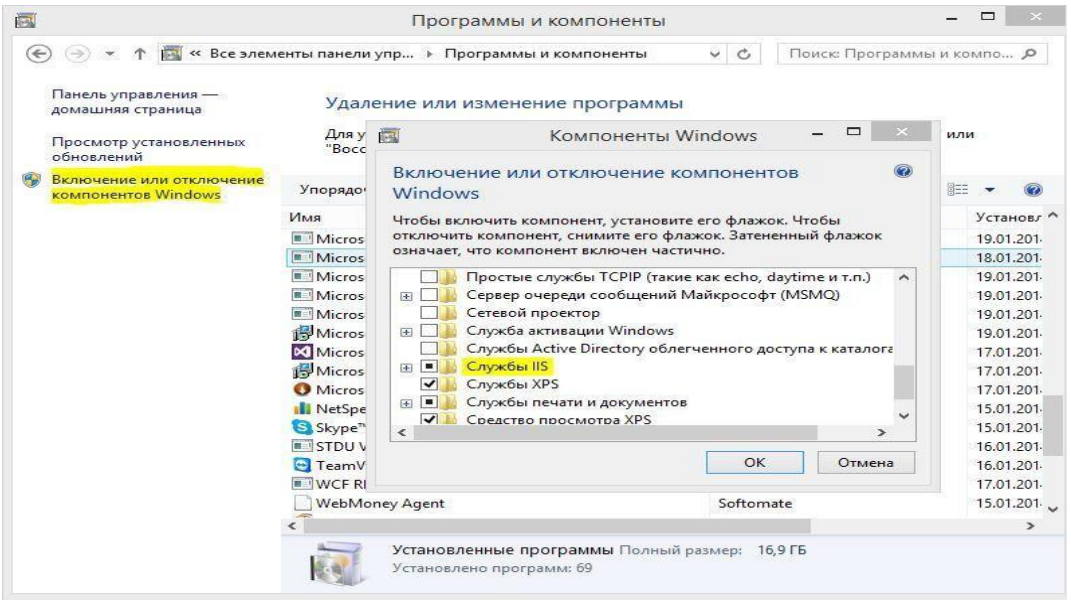


Рисунок 1.67 - Активация службы ISS

Выбираем службу ISS

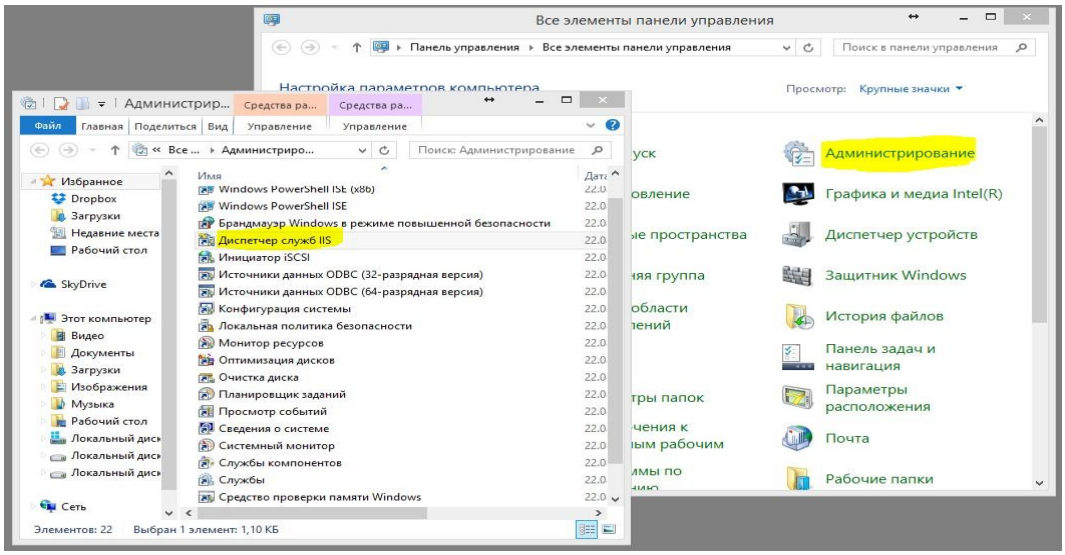


Рисунок 1.68 - Выбор службы ISS

В открывшемся окне жмем на «Сертификаты сервера»

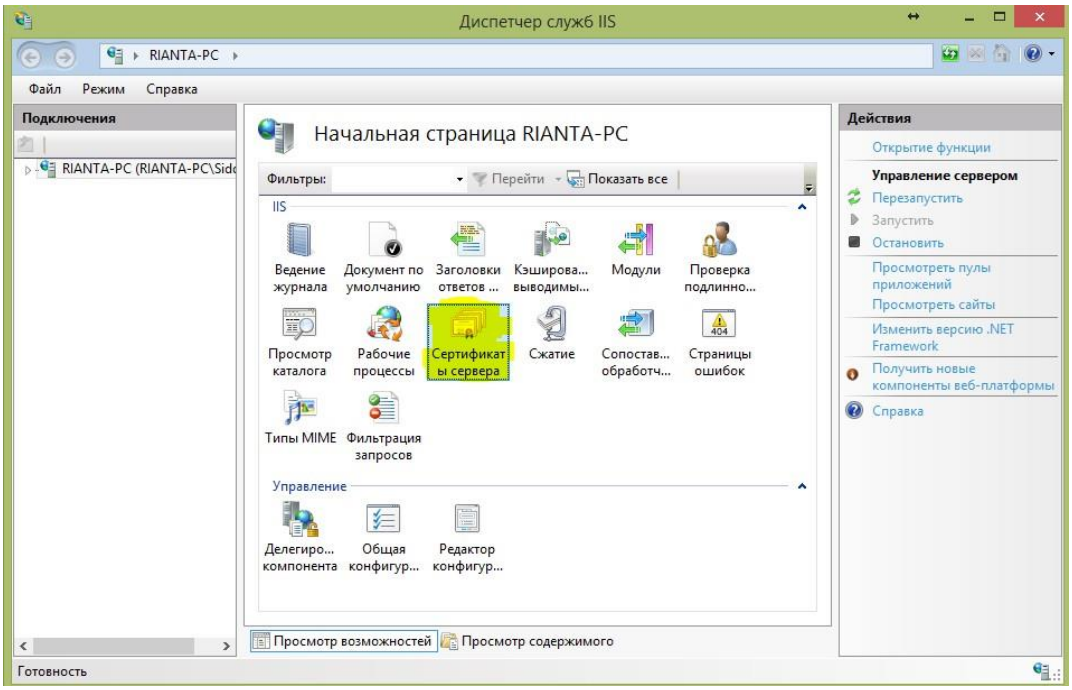


Рисунок 1.69 - Создание сертификата

Создаем самозаверенный сертификат

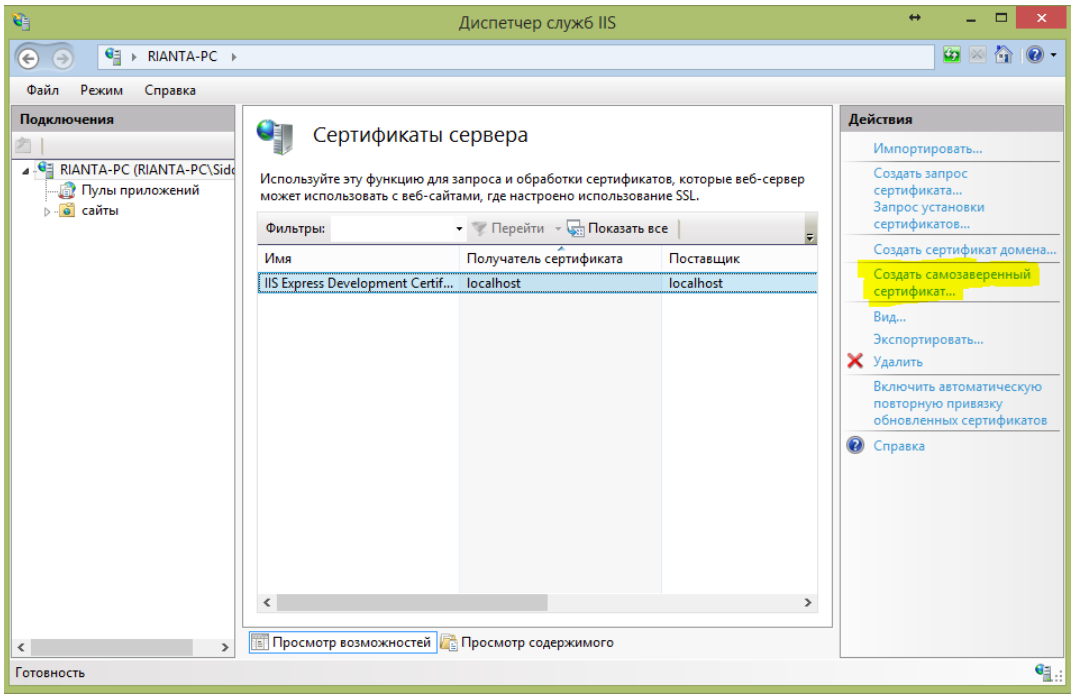


Рисунок 1.70 - Создание самозаверенного сертификата

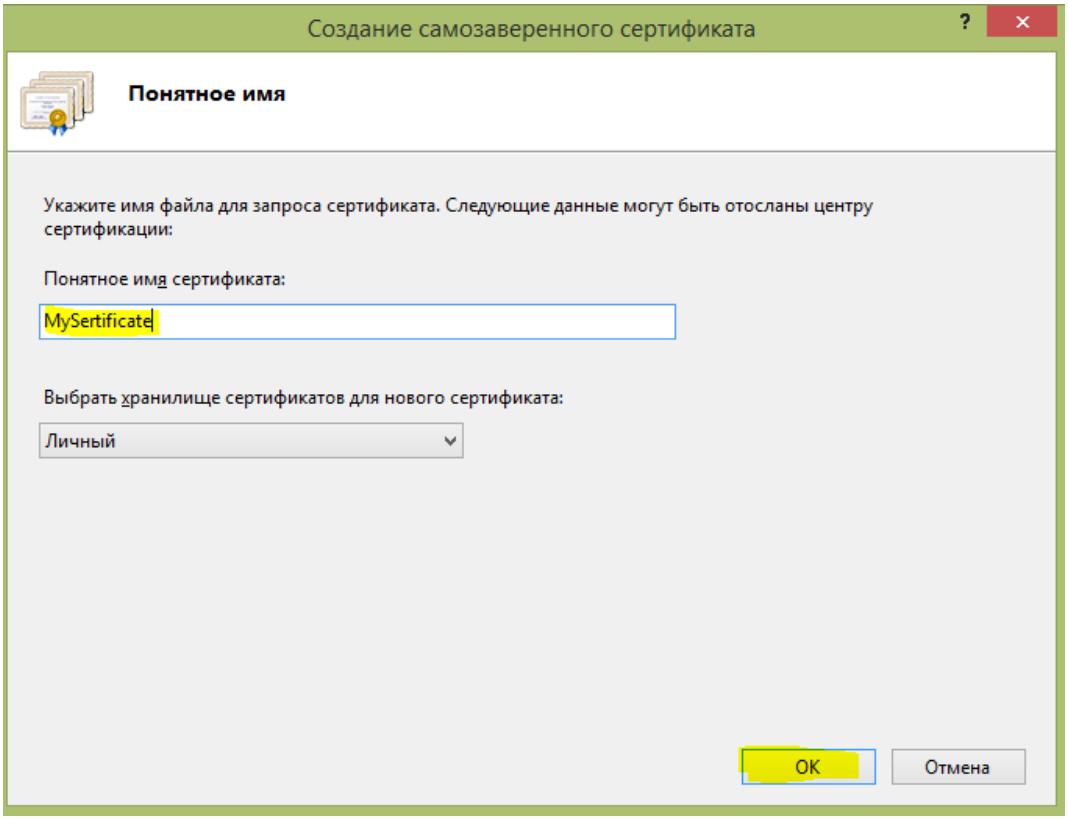


Рисунок 1.71 - Ввод имени сертификата

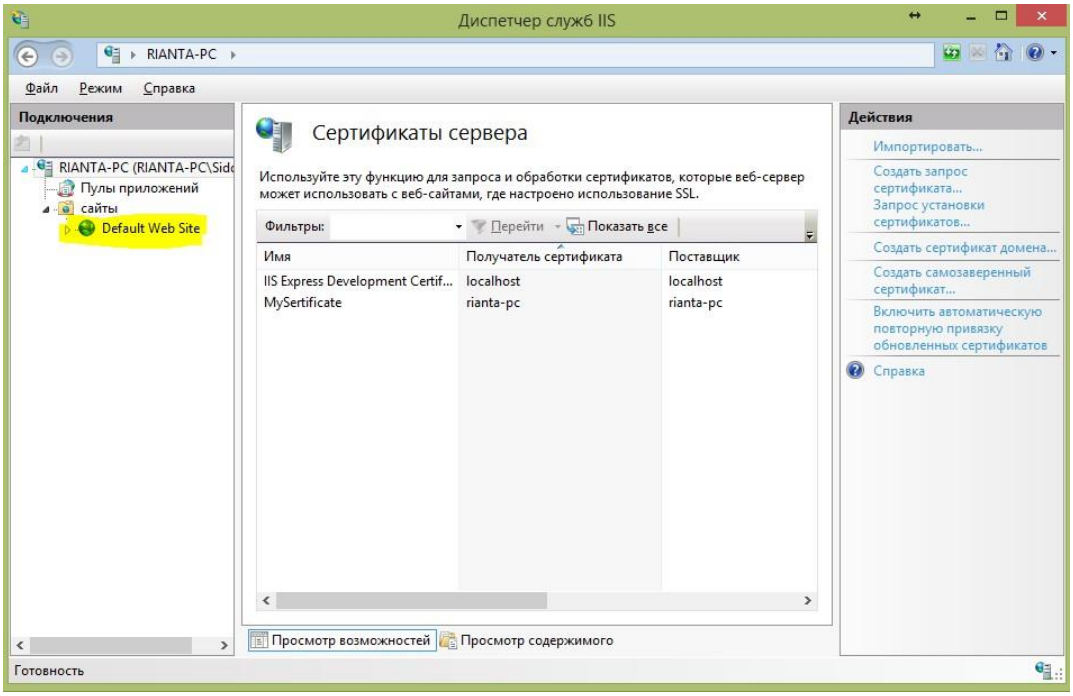


Рисунок 1.72 - Отображение сертификата

Привязываем сертификат к нужному IP

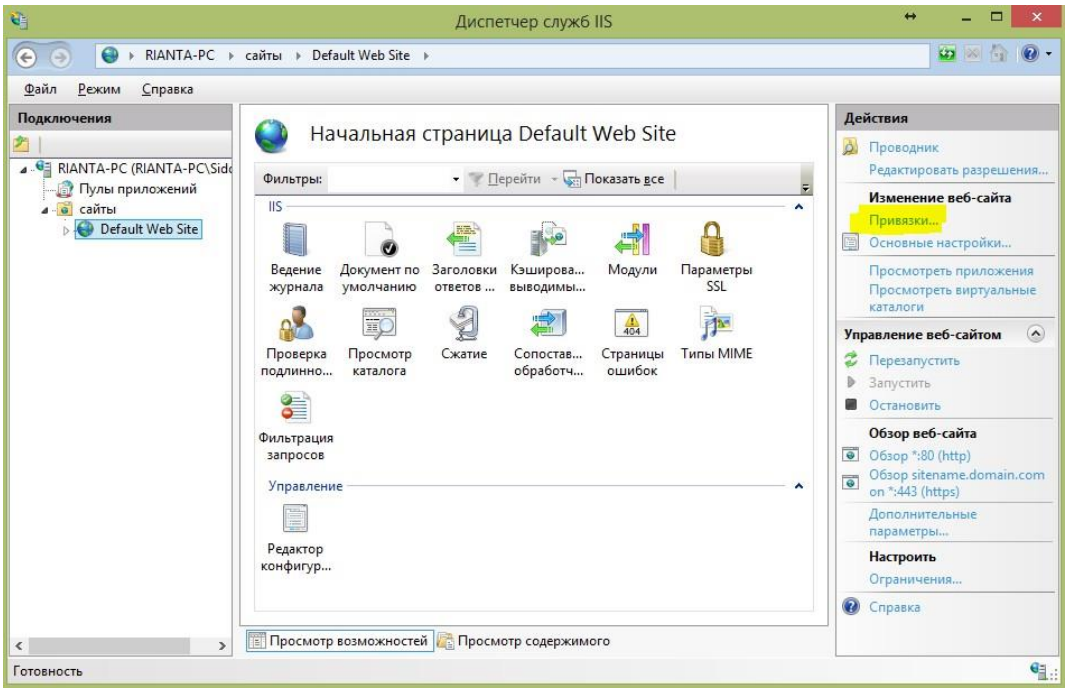


Рисунок 1.73 - Привязка сертификата к сайту (компьютеру)

Затем с помощью программы THEGREENBOW устанавливаем защищенное SSL – соединение.

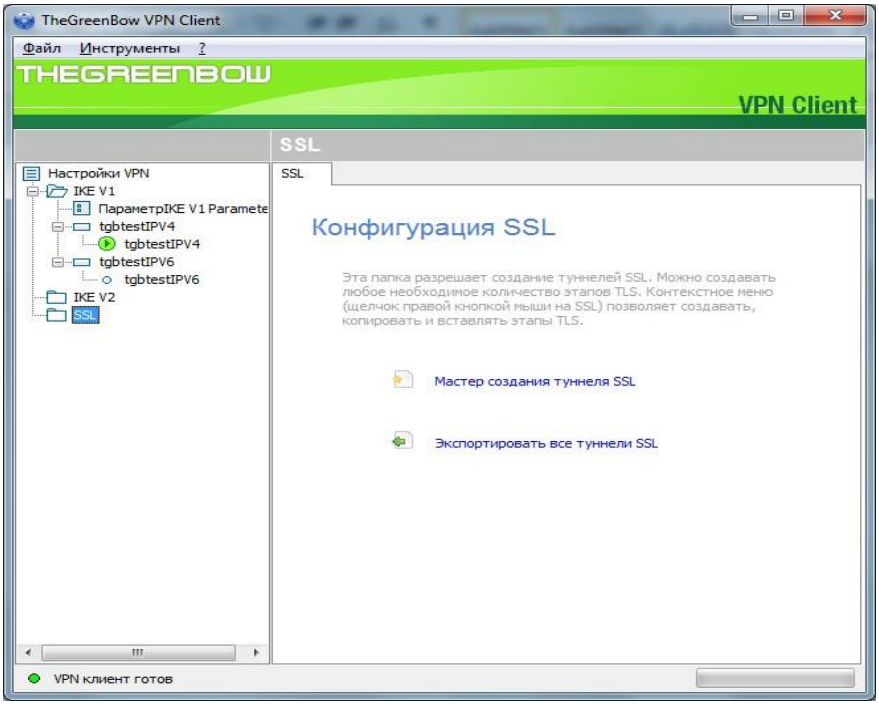


Рисунок 1.74 - Окно раздела SSL

Указываем путь к сертификату

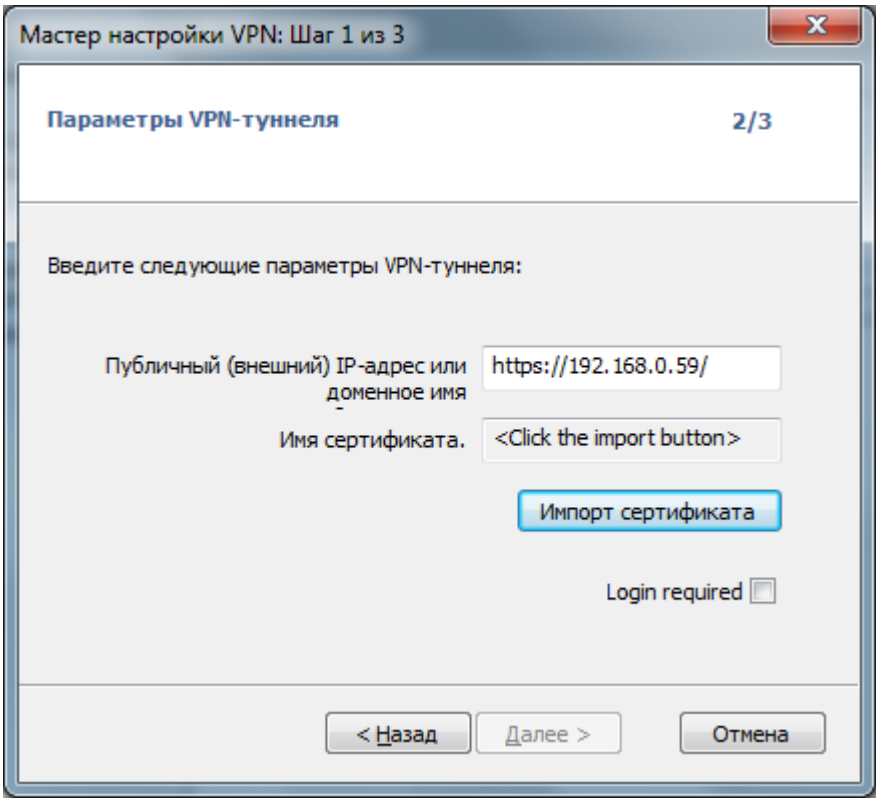


Рисунок 1.75 - Окно мастера настройки VPN-туннеля

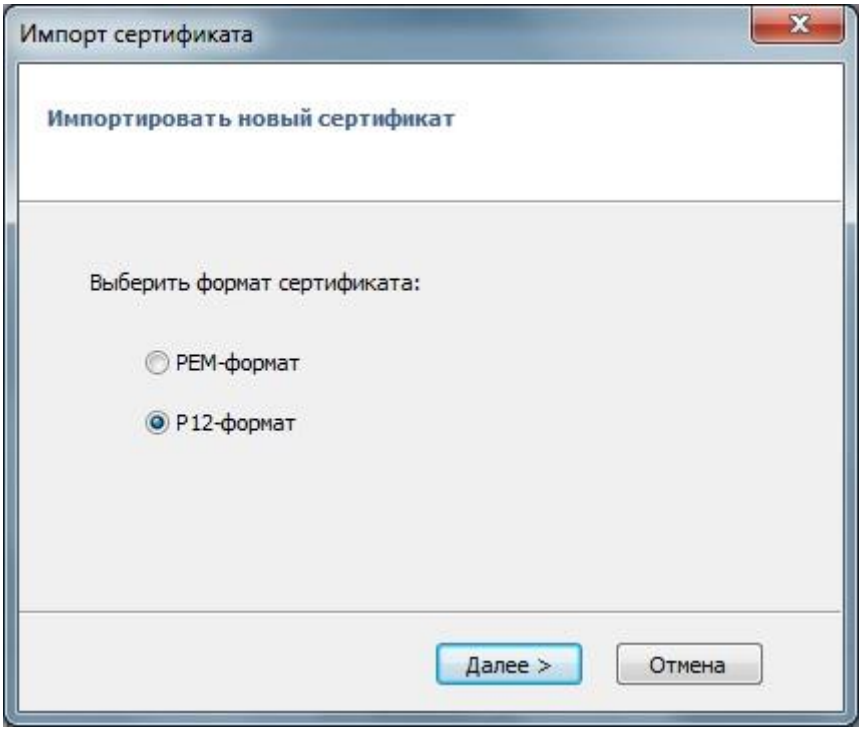


Рисунок 1.76 - Импорт сертификата

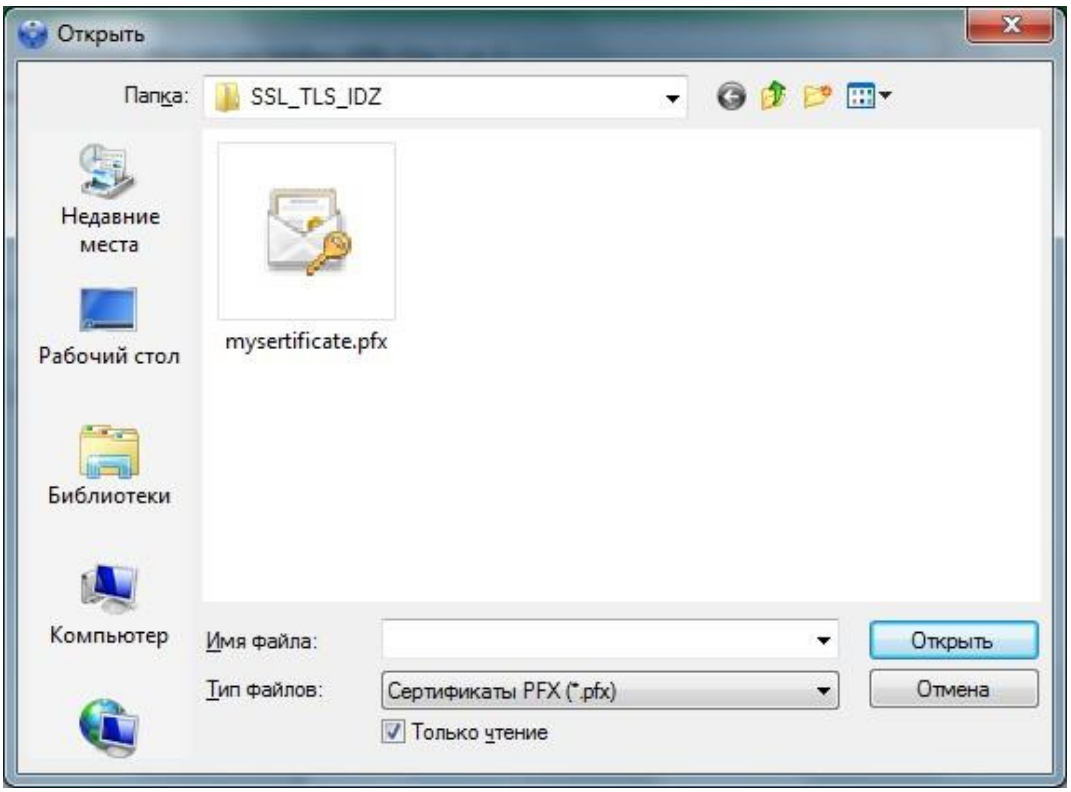


Рисунок 1.77 - Выбор созданного сертификата

Вводим пароль для доступа к сертификату

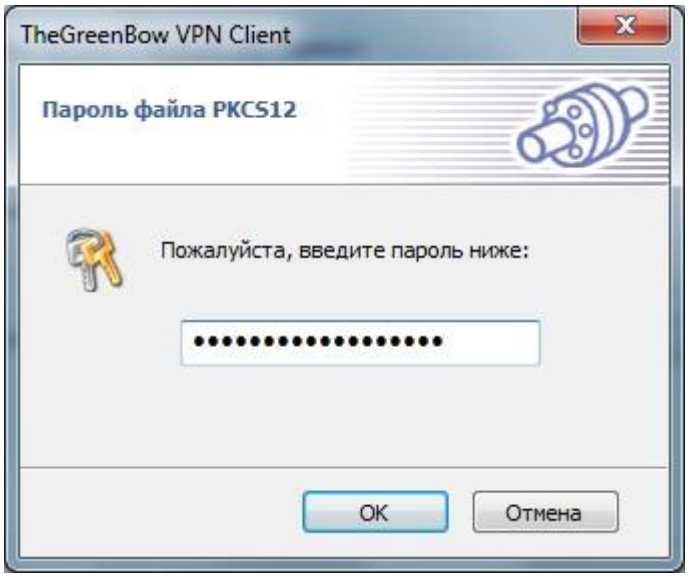


Рисунок 1.78 - Окно ввода пароля для доступа к сертификату

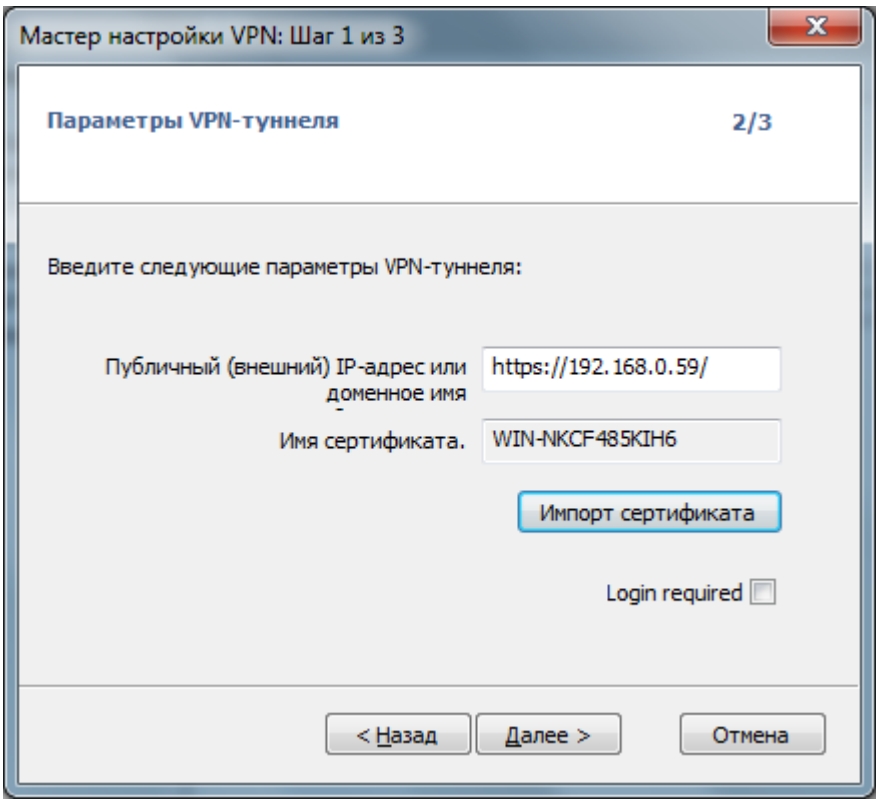


Рисунок 1.79 - Окно мастера настройки VPN-туннеля

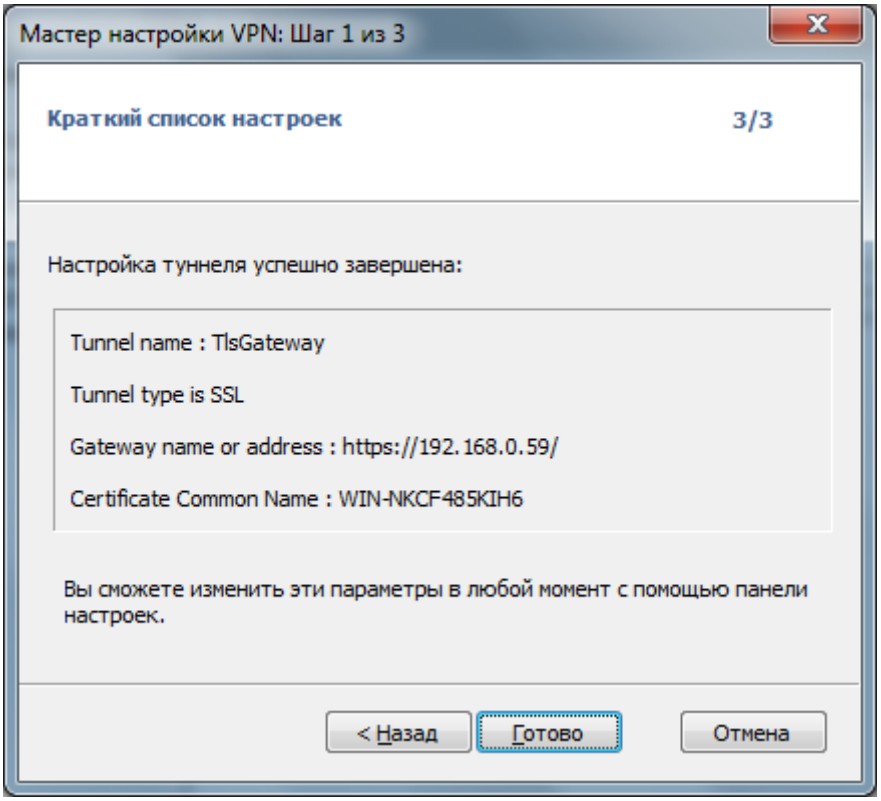


Рисунок 1.80 - Подтверждение об успешно созданном туннеле

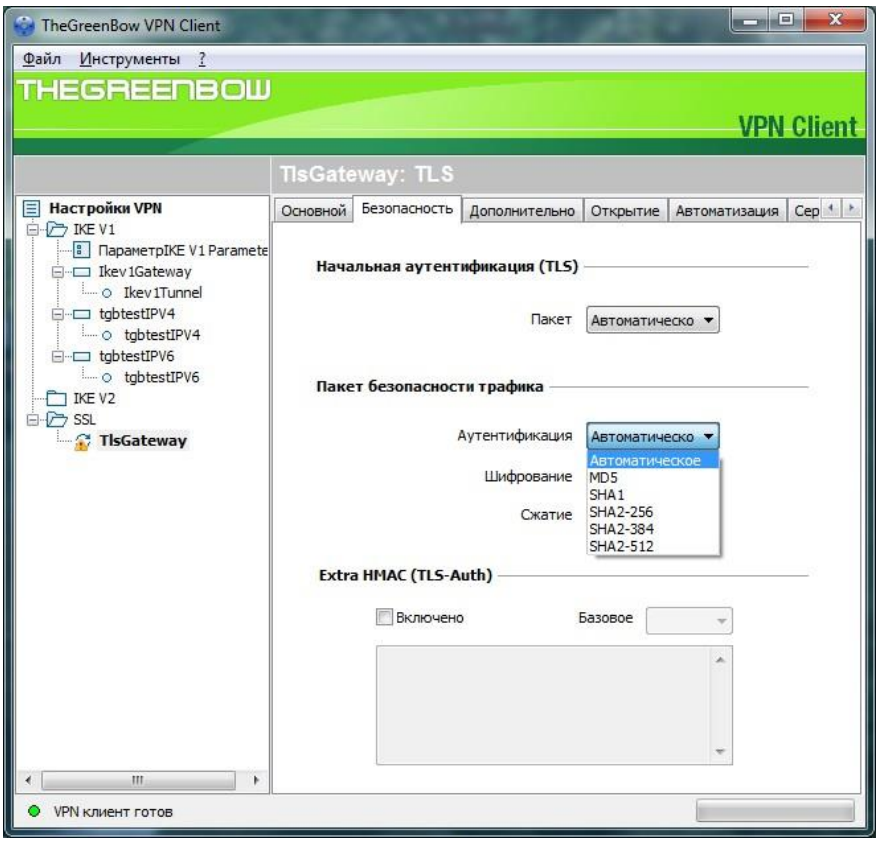


Рисунок 1.81 - Параметры аутентификации трафика

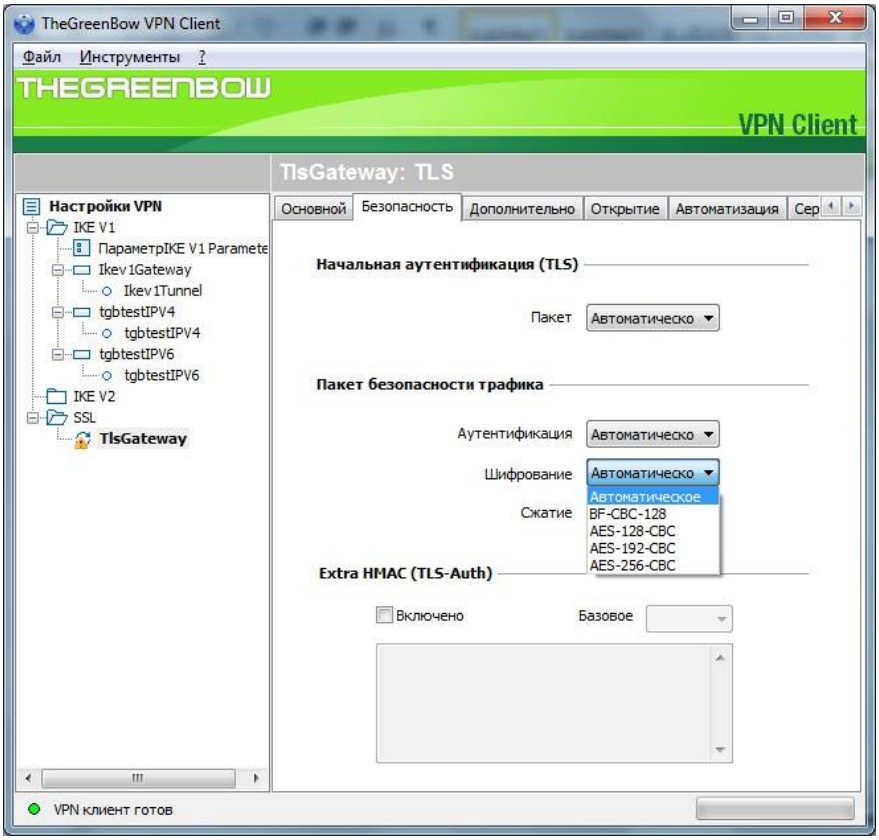


Рисунок 1.82 - Параметры шифрования

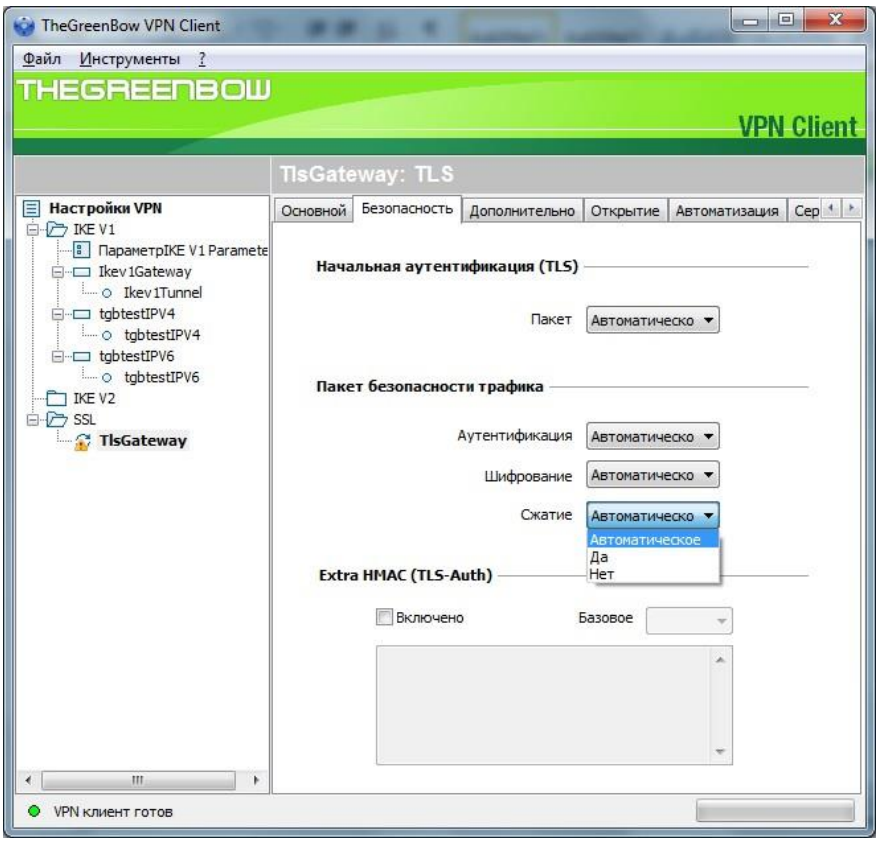


Рисунок 1.83 - Параметры сжатия

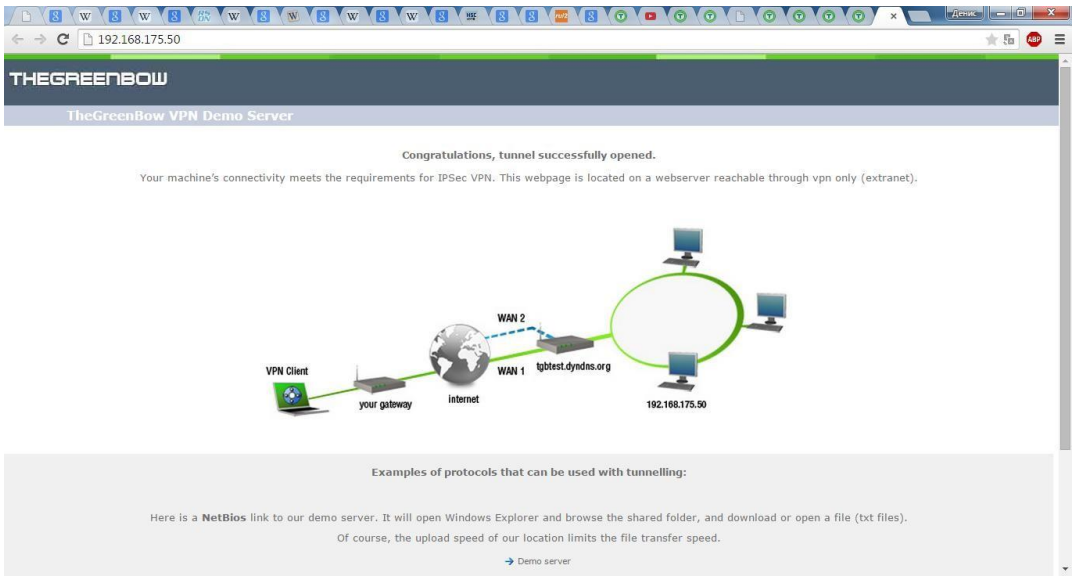


Рисунок 1.84 - Окно в браузере об успешном соединении

Выводы

Протокол безопасности транспортного уровня обеспечивает услугибезопасности из конца в конец для приложений, которые пользуются протоколами транспортного уровня, такими как, например, TCP. На сегодняшний день преобладает применение двух протоколов: *протокол "Уровень Безопасных Розеток" (SSL - Secure Sockets*

Layer) и *протокол "Безопасность Транспортного уровня"* (TLS - Transport Layer Security).

- SSL или TLS обеспечивают такие услуги, как фрагментация, сжатие, целостность сообщения, конфиденциальность и создание кадра данных, полученных от прикладного уровня. Как правило, SSL (или TLS) может получить прикладные данные от любого протокола прикладного уровня, но работает протокол обычно с *HTTP*.
- Комбинация алгоритмов смены ключей, хэширования и алгоритм шифрования определяют *набор шифров* для каждого сеанса.
- Для того чтобы обмениваться заверенными и конфиденциальными сообщениями, клиенту и серверу необходимо иметь шесть единиц криптографической секретности (четыре ключа и два вектора инициализации).
- В SSL (или TLS) отличают *подключение* и сеанс. В сеансе одна сторона играет роль клиента, а другая - роль сервера; при *подключении* обе стороны играют одинаковые роли, на равном подуровне.
- SSL (или TLS) определяет четыре протокола на двух уровнях: *протокол установления соединения*, *протокол изменения параметров шифрования, аварийный протокол* и *протокол передачи записей*. *Протокол установления соединения* использует несколько сообщений, чтобы договориться о *наборе шифров*, подтвердить подлинность сервера для клиента и клиента для сервера, если это необходимо, и обмениваться информацией для организации криптографической секретности.

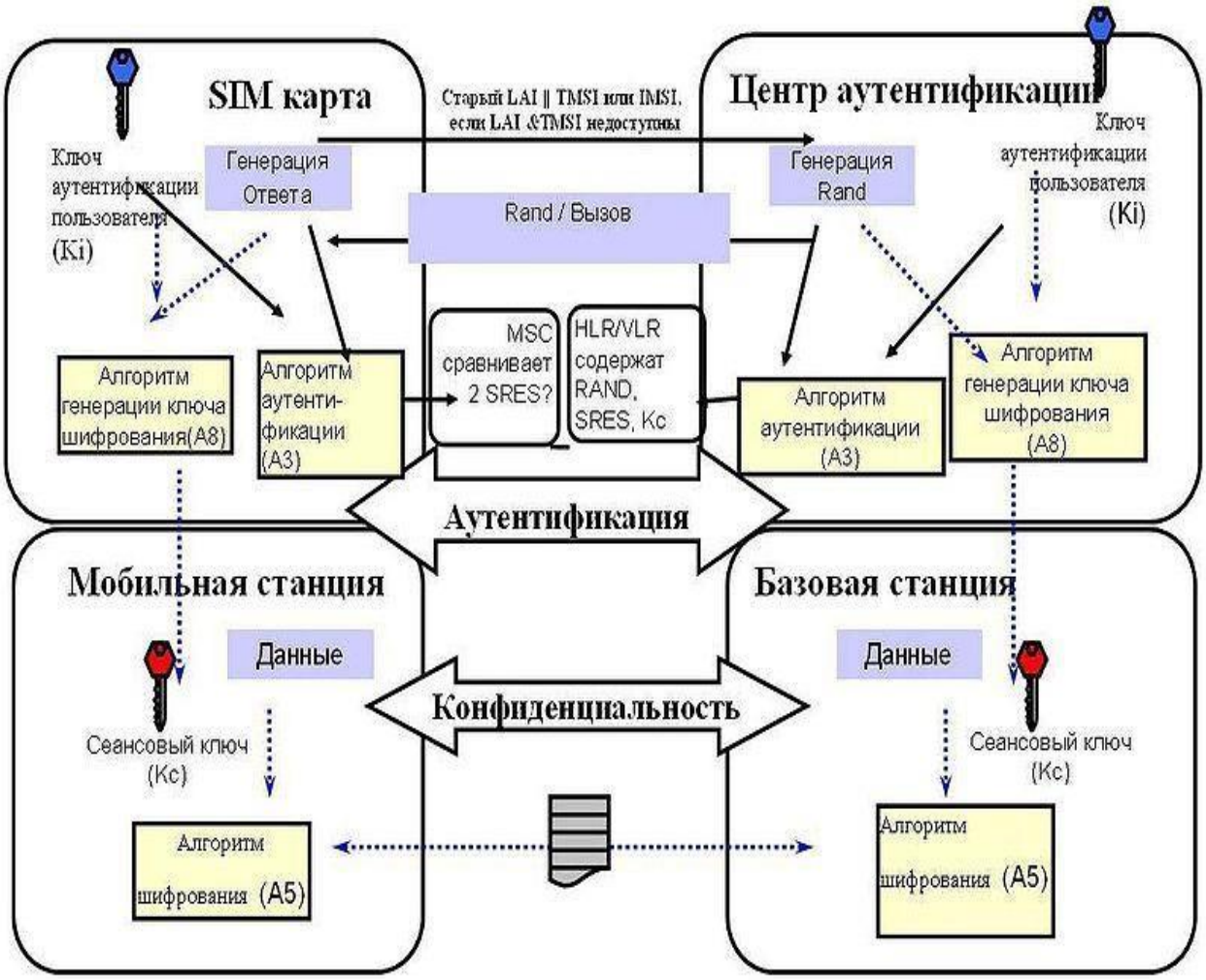
Протокол изменения параметров шифрования определяет процесс перемещения информации между состоянием ожидания и активным состоянием. *Аварийный протокол* передает извещения об ошибках и ситуациях, отклоняющихся от нормальных.

Протокол передачи записей доставляет сообщения от верхнего уровня (протокол установления соединения, аварийный протокол, ChangeCipherSpec-протокол) или прикладного уровня.

2. МЕТОДЫ ШИФРОВАНИЯ В СОВРЕМЕННЫХ СИСТЕМАХ СВЯЗИ И ПЕРЕДАЧИ ДАННЫХ

2.1 БЕЗОПАСНОСТЬ GSM СЕТЕЙ

Прежде чем приступить к описанию алгоритма шифрования используемого в GSM сетях рассмотрим каким образом происходит аутентификация пользователя и формирования ключа шифрования. Для этоговоспользуемся картинкой.



На данном рисунке схематично представлены следующие шаги:

1. Телефон оператора подключается к сети.
2. Для подтверждения своей подлинности телефон посылает специальный идентификационный код, называемый TMSI (Temporary Mobile Subscriber Identity).

3. Центр Аутентификации(ЦА) генерирует 128-битное случайное число RAND и посылает его на Мобильную Станцию(МС).

4. МС зашифровывает полученное число RAND, используясвой секретный ключ K_i и алгоритм аутентификации A3.

5. МС берет первые 32 бита из последовательности, полученной на предыдущем шаге(назовем их SRES (signed response)) иотправляет их обратно на ЦА.

6. ЦА проделывает ту же операцию и получает 32 битную последовательность XRES (expected response).

7. После чего ЦА сравнивает SRES и XRES. В случае, если оба значения равны, телефон считается аутентифицированным.

8. МС и ЦА вычисляют сессионный ключ шифрования,используя секретный ключ K_i и алгоритм формирования ключа A8 $K_c=A8_{ki}(RAND)$

9. Говоря об алгоритмах аутентификации A3 и алгоритме формирования ключа A8, следует отметить что на практике большинство сотовых операторов используют для этих целей один алгоритм, называемый COMP128(он имеет множество модификаций COMP128-COMP128-10.COMP128-3). COMP128 представляет собой обыкновенную хэш-функцию, на входе которая принимает 128-битную последовательность и на выходе возвращает 96-битную.

В криптографии, попытка сэкономить время разработчикам обернулась полным провалом. Безопасность GSM сетей изначально основывалась на принципе «безопасность за счёт неизвестности». И когда в 1998 году алгоритм был вскрыт группой исследователей состоящих из Marc Briceno, Ian Goldberg и David Wagner обранужилась одна занятная особенность: последние 10 бит секретного ключа K_i всегда равнялись нулю. Используя это любопытное свойство, а также уязвимость COMP128 к «атаке дней рождений» Marc Briceno, Ian Goldberg и David Wagner смогли извлечь секретный ключ K_i из SIM-карты. Результатом этого исследования стал повсеместный отказ от алгоритма COMP128 и его замена на более надежные модификации COMP128-2 и COMP128-3, технические детали которых держатся в тайне.

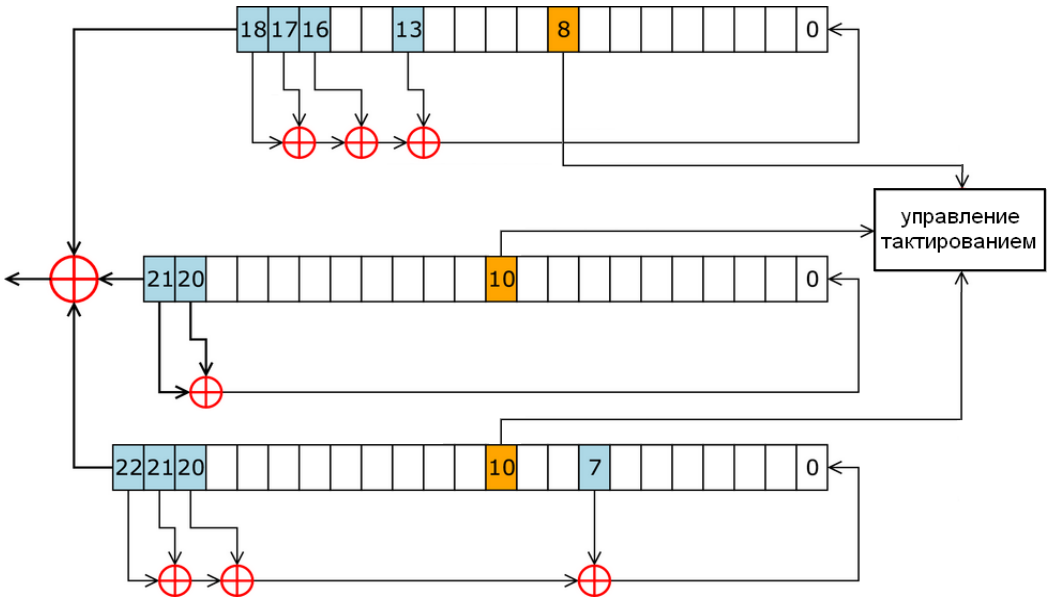
Алгоритм шифрования A5/1

В качестве алгоритма шифрования в GSM используются алгоритмы изсемейства A5. На сегодняшний день их всего 3:

- **A5/1** — поточный шифр, наиболее распространенный на сегодня.
- **A5/2**-вариант предыдущего алгоритма «для бедных». Оченьпохож на своего «старшего брата», но изначально задумывался, как сильно ослабленная версия A5/1. В настоящее время не используется
- **A5/3**-блочный шифр. Разработан в 2002 году с целью заменить устаревший A5/1. Однако в настоящее время используется только в 3GPP сетях. У алгоритма найден ряд

уязвимостей, но о практических атаках речи пока не идет.

Рассмотрим подробнее алгоритм A5/1



Внутреннее состояние шифра A5/1 состоит из трех линейных регистров сдвига с обратной связью R1, R2, R3, длиной 19, 22 и 23 бита соответственно(всего 64 бита).

Сдвиг в регистрах R1, R2, R3 происходит только при выполнении определенного условия. Каждый регистр содержит " бит управления тактированием". В R1 это 8-й бит, а в R2 и R3 — 10-й. На каждом шаге сдвигаются только те регистры у которых значение бита синхронизации равно большинству значений синхронизирующих битов всех трех регистров.

На сегодняшний день известно большое количество успешных атак на GSM шифрование и все они относятся к атакам типа known-plaintext, т.е. для восстановления ключа атакующему помимо зашифрованных фреймов необходимо знать так же незашифрованные данные, которые соответствуют этим фреймам. На первый взгляд такое требование может показаться фантастическим, однако из-за специфики стандарта GSM, в котором помимо голосового трафика передаются различные системные сообщения, такого рода атаки из разряда теоретических переходят в разряд практических.

Системные сообщения GSM содержат повторяющиеся данные и могут использоваться злоумышленником. В частности, метод предложенный Karsten Nohl в 2010 году основан как на поиске такого рода данных в шифротексте и простом переборе различных вариантов ключей, хранящихся в радужных таблицах, до тех пор пока не будет найден ключ, порождающий нужный шифротекст для известного заранее системного сообщения.

2.2. КРИПТОГРАФИЧЕСКАЯ ЗАЩИТА БЕСПРОВОДНЫХ СЕТЕЙ СТАНДАРТА 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 [2].

На рисунке 2.1 представлена структура сети стандарта LTE.

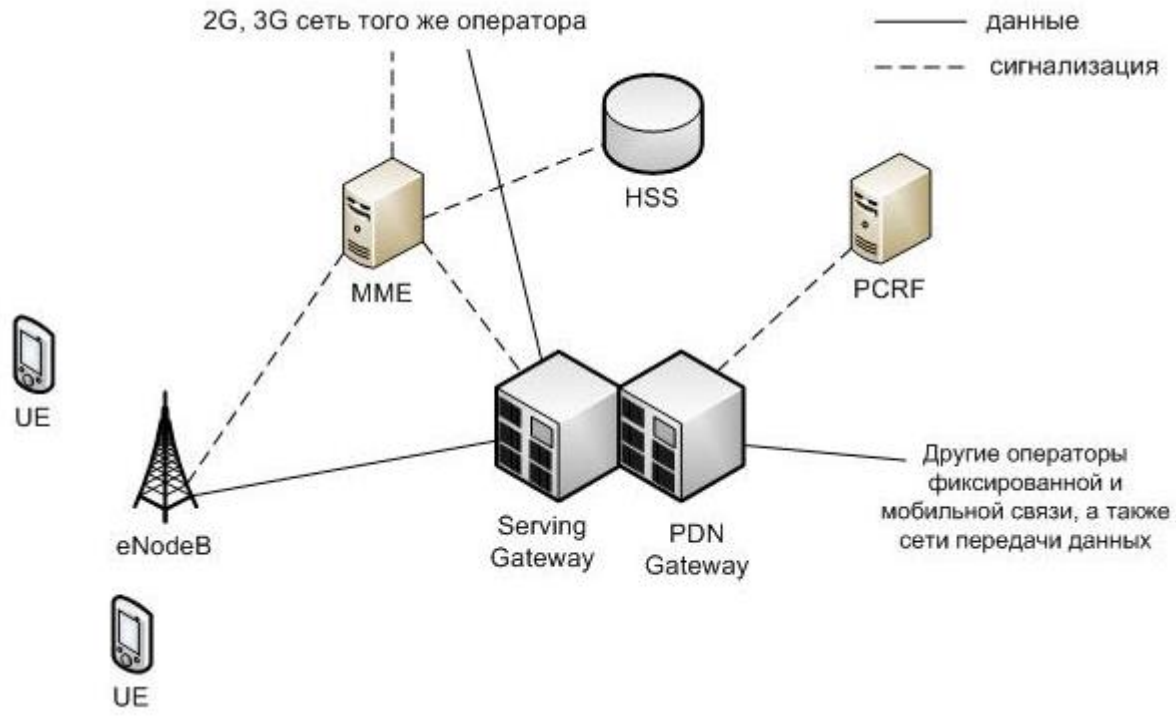


Рисунок 2.1 – Структура сети стандарта LTE

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.

Существующие методы и стандарты 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.

Алгоритм аутентификации и генерации ключа представлен на рисунке 2.2.

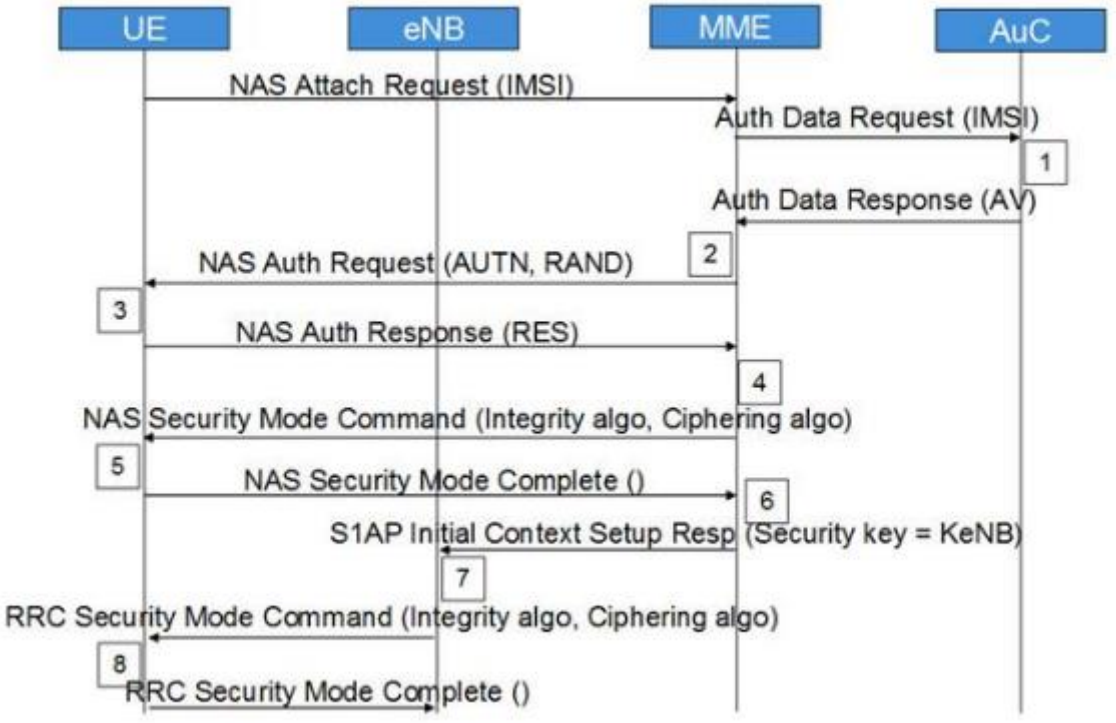


Рисунок 2.2 – Алгоритм аутентификации и генерации ключа

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. Слои безопасности представлены на рисунке 2.3.

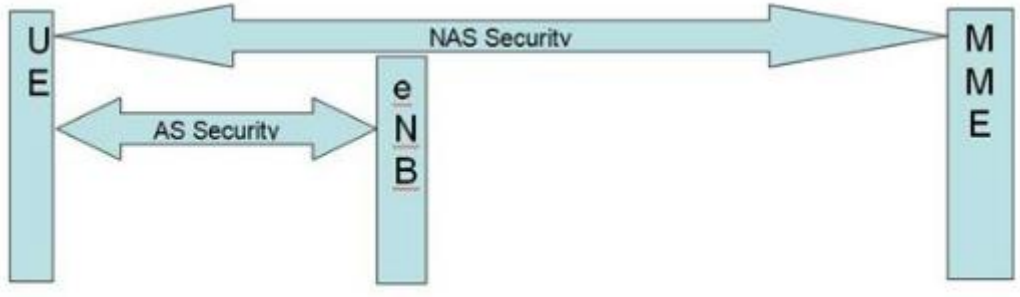


Рисунок 2.3 – Слои безопасности

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 2.4.

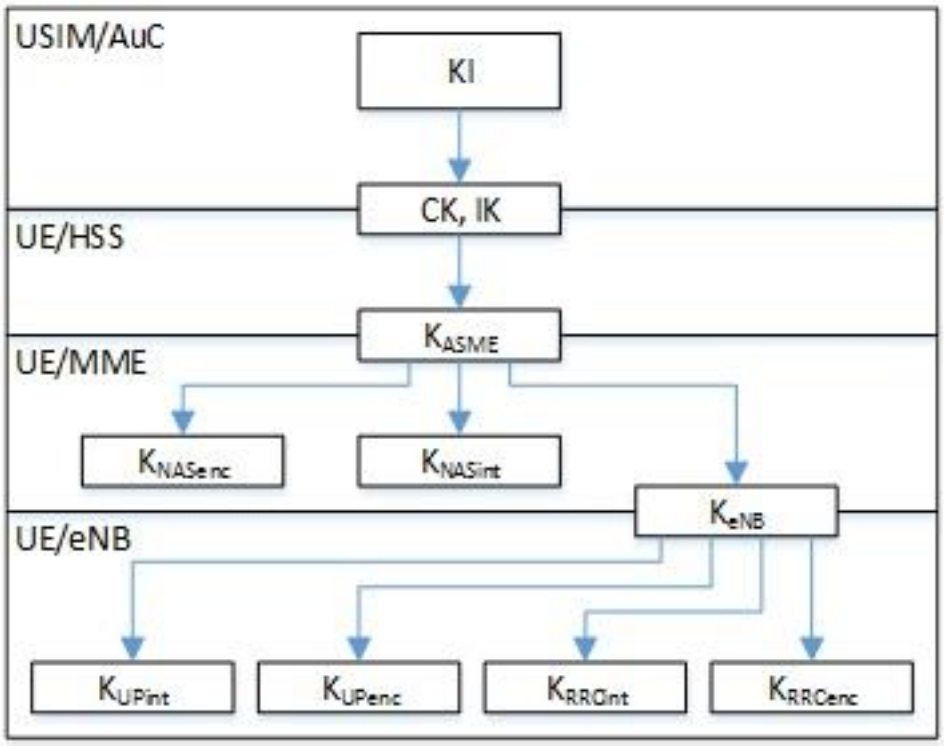


Рисунок 2.4 – Иерархия ключей в LTE

Векторы аутентификации представлены на рисунках 2.5 и 2.6:

- 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.

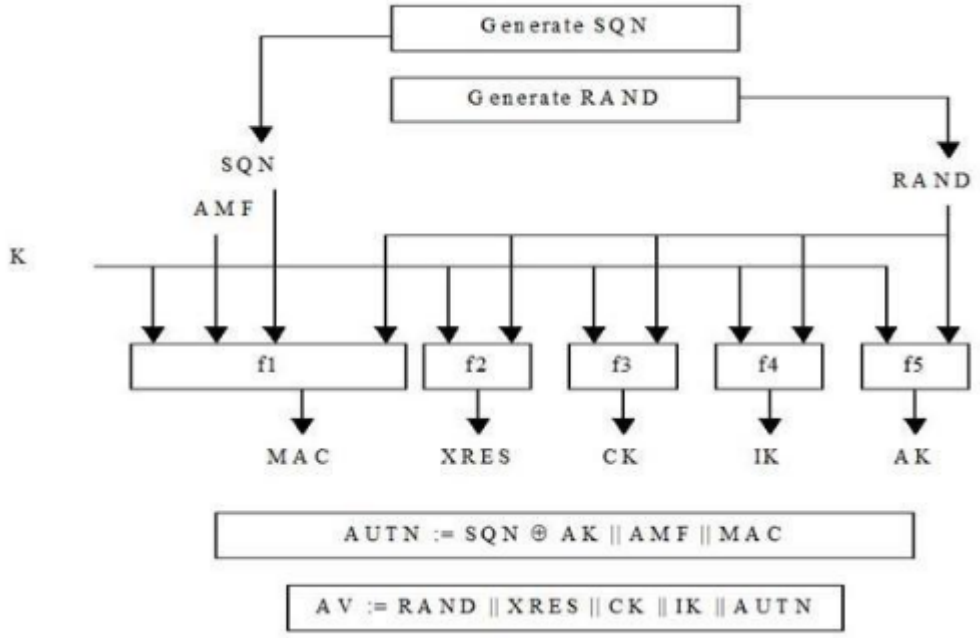


Рисунок 2.5 – Создание векторов на передающей стороне

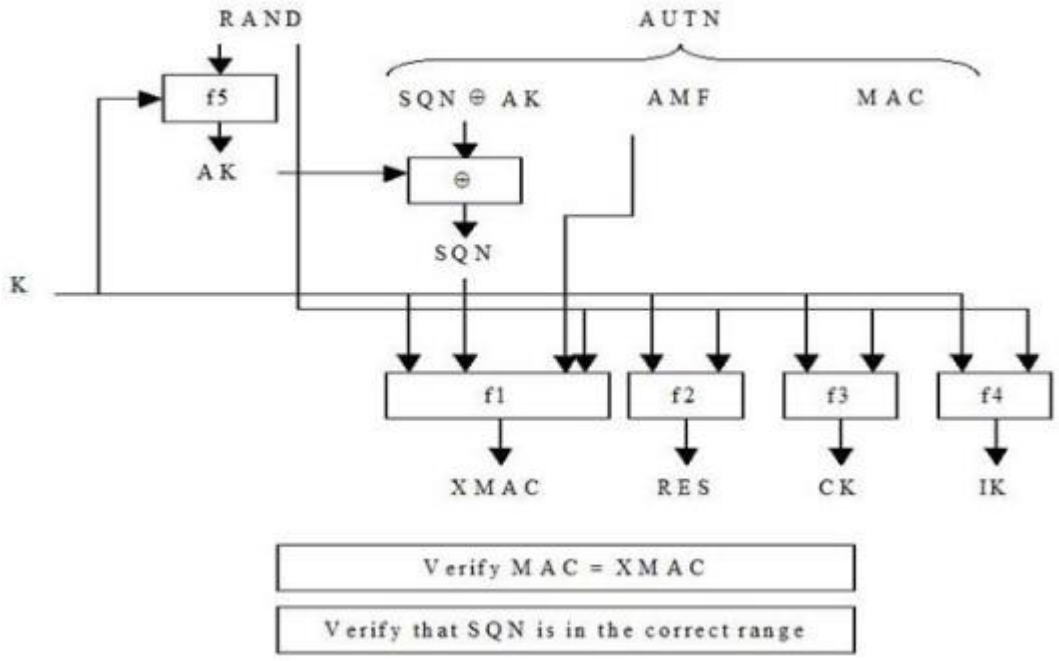


Рисунок 2.6 – Преобразование векторов на приемной стороне

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 2.7 представлена генерация ключей шифрации и целостности для NAS сигнализации.

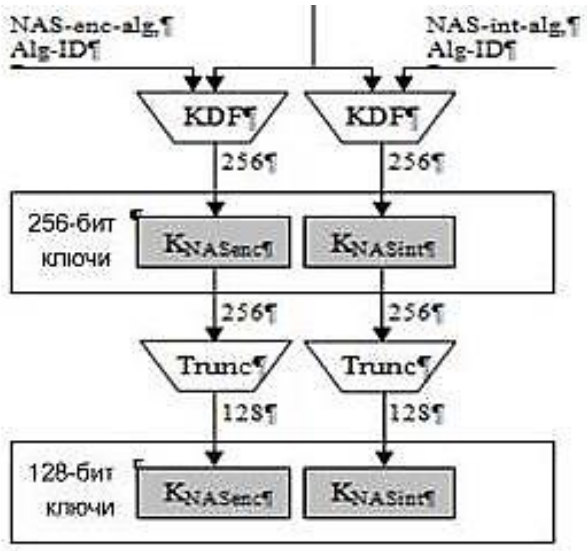


Рисунок 2.7 – Генерирование ключей шифрации и целостности для NAS сигнализации

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. 2.8.

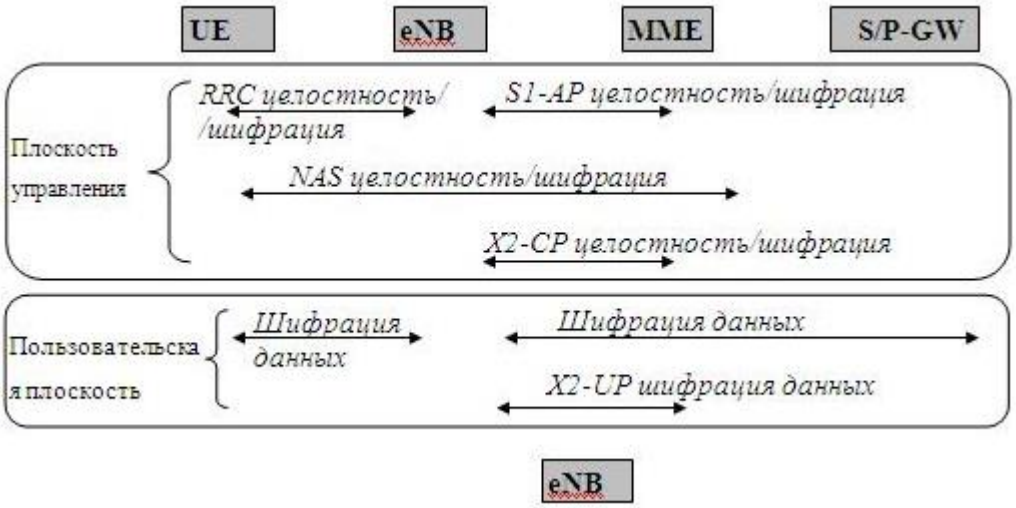


Рисунок 2.8 – Процедуры безопасности в сети LTE

Алгоритм шифрации и дешифрации сообщений представлен на рисунке 2.9.

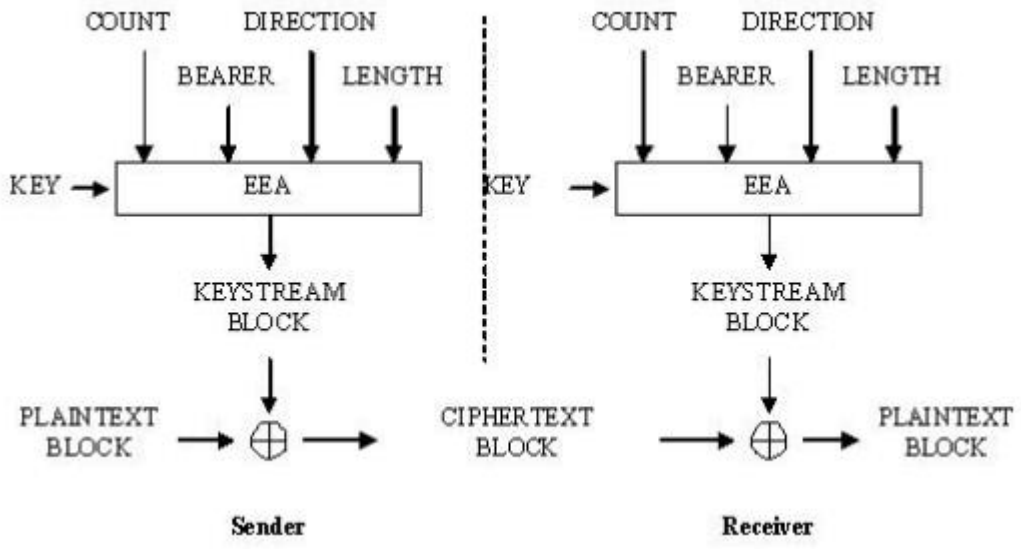


Рисунок 2.9 – Алгоритм шифрации и дешифрации в LTE

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 2.10.

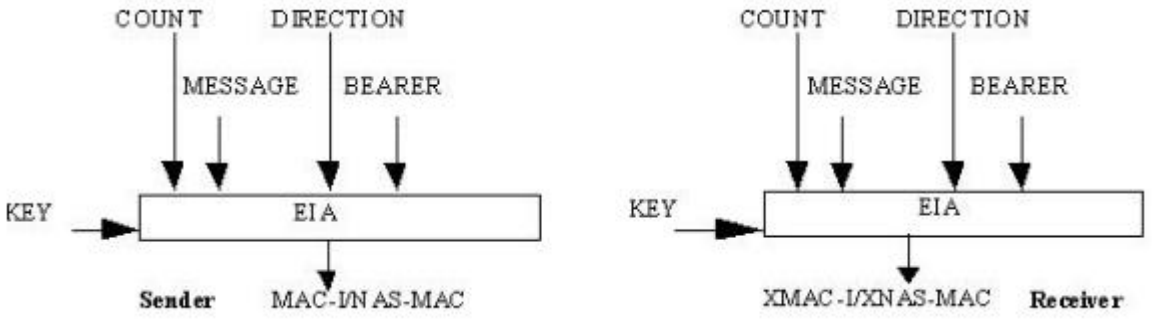


Рисунок 2.10 – Алгоритм проверки целостности

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.

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 2.11 представлен общий алгоритм построения AES шифра.

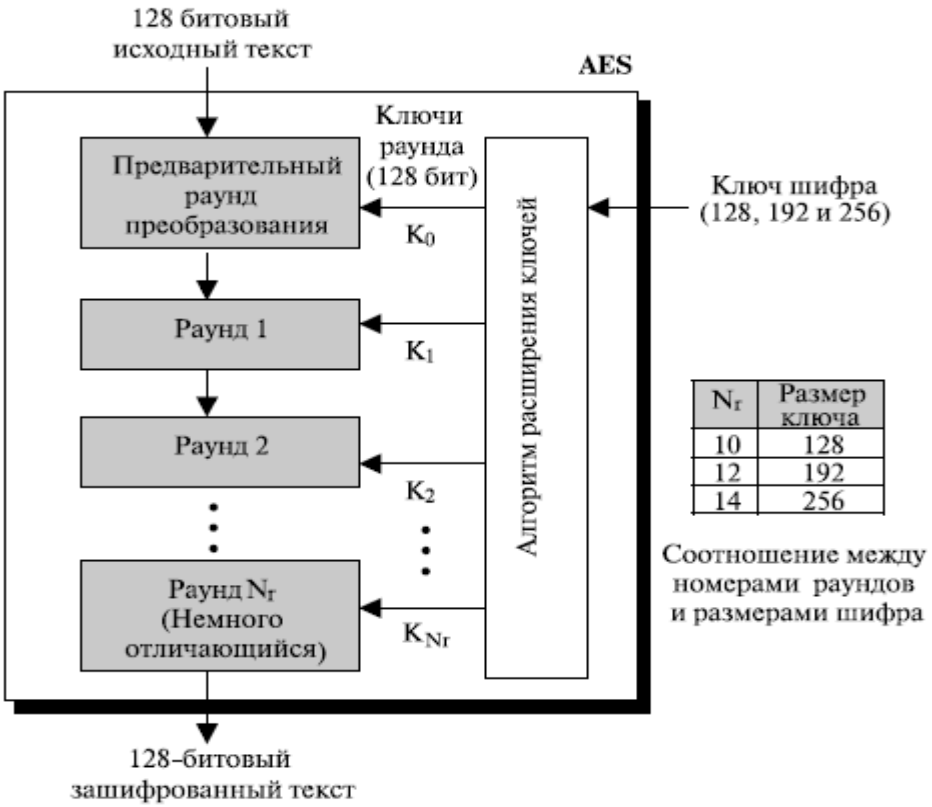


Рисунок 2.11 – Алгоритм AES шифра

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 2.12 представлено преобразование из блока данных в матрицу состояния и обратно.

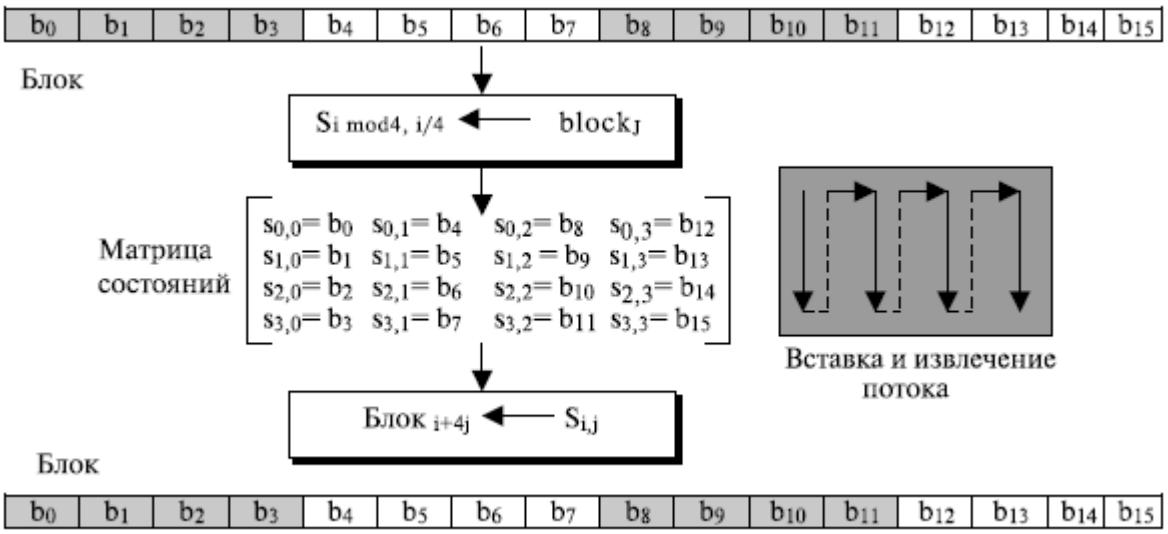


Рисунок 2.12 – Преобразование блок–матрица состояний и наоборот

Структура каждого раунда шифрования изображена на рисунке 2.13.

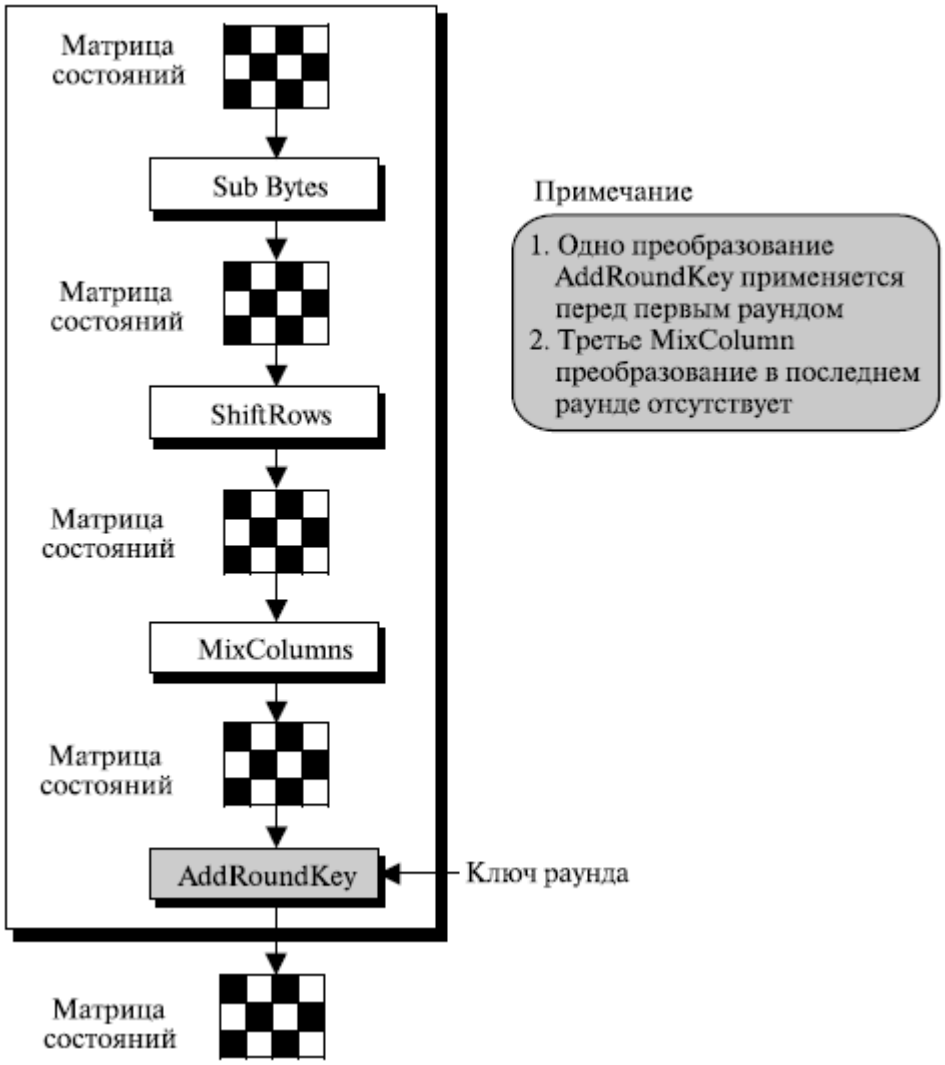


Рисунок 2.13 – Структура раунда шифрования

Общий алгоритм шифрования и дешифрования представлен на рисунке 2.14.

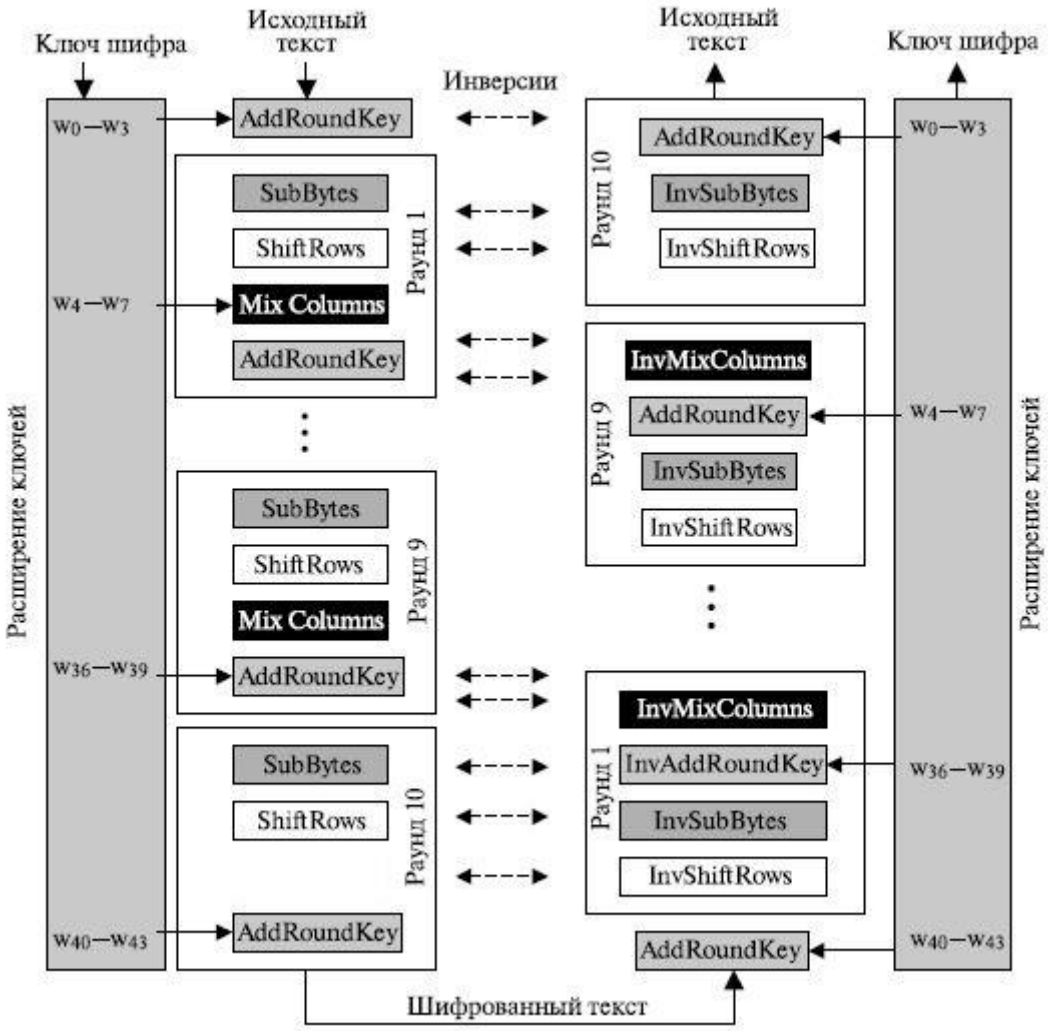


Рисунок 2.14 – Шифрование/дешифрование по алгоритму AES

Алгоритм программы шифрования и дешифрования представлен в приложении А. Основные операции раундов будут рассмотрены ниже.

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 2.15 представлен принцип работы CBC режима.

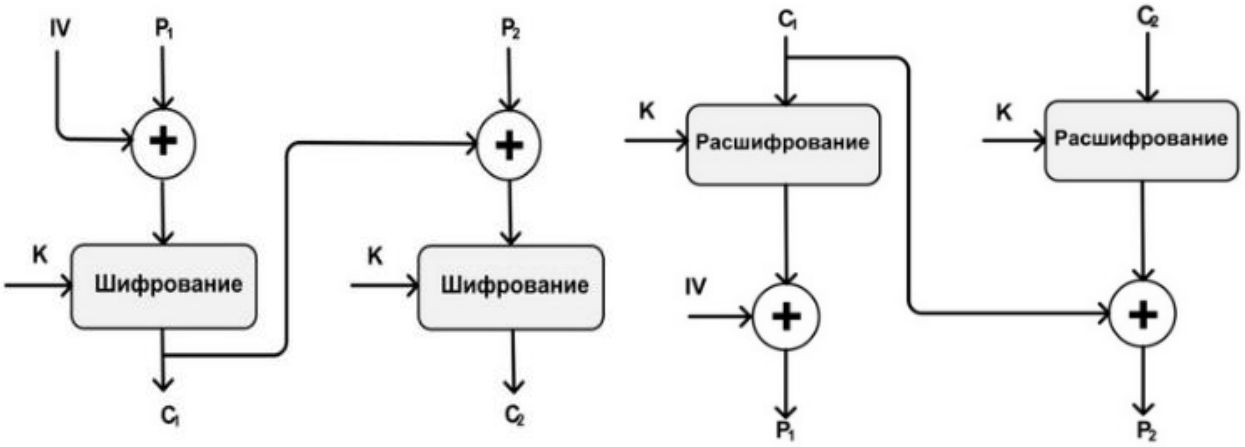


Рисунок 2.15 – Шифрование/дешифрование в режиме CBC

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 2.16 представлен принцип режима CFB.

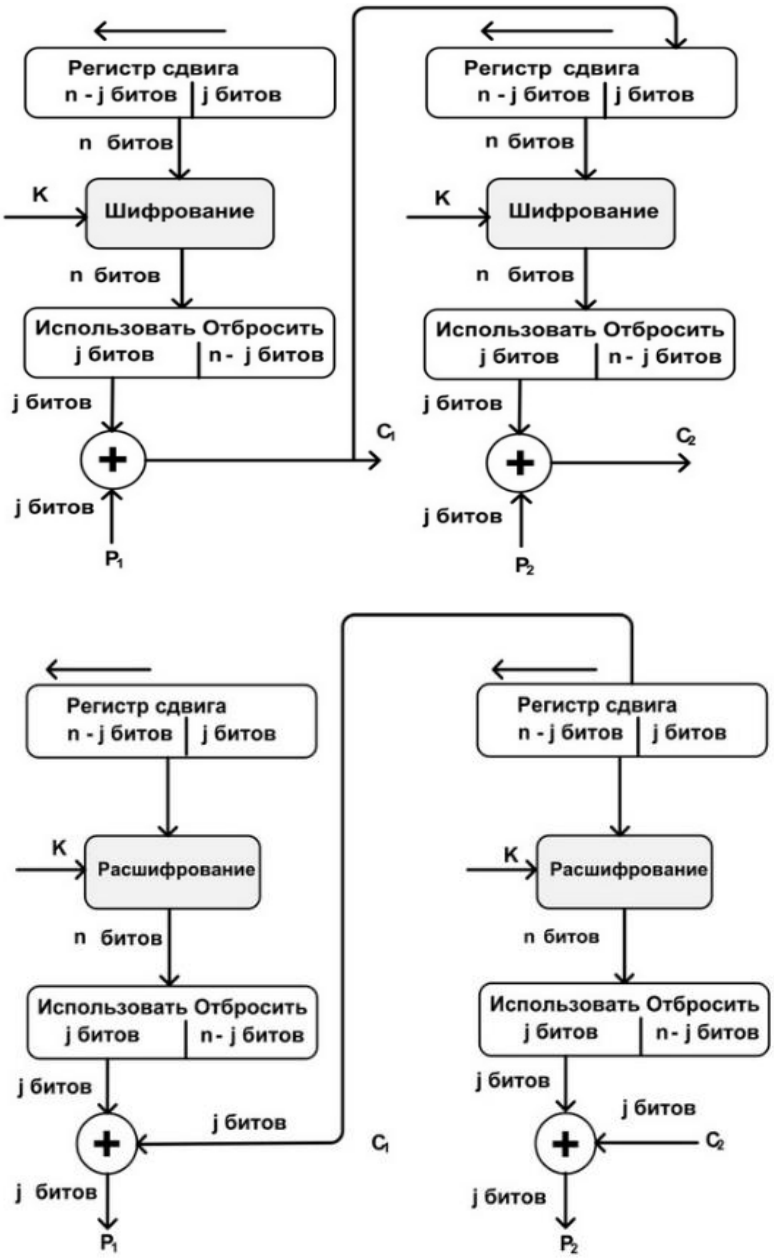


Рисунок 2.16 – Шифрование/дешифрование в режиме CFB

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 2.17 представлен принцип режима OFB.

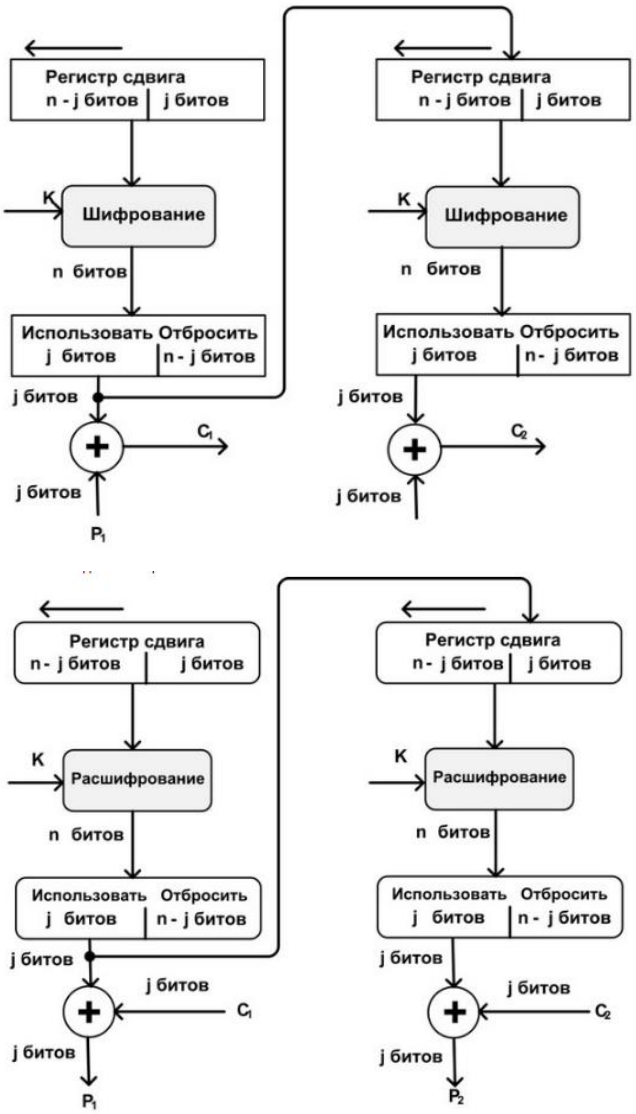


Рисунок 2.17 – Шифрование/дешифрование в режиме OFB

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

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.

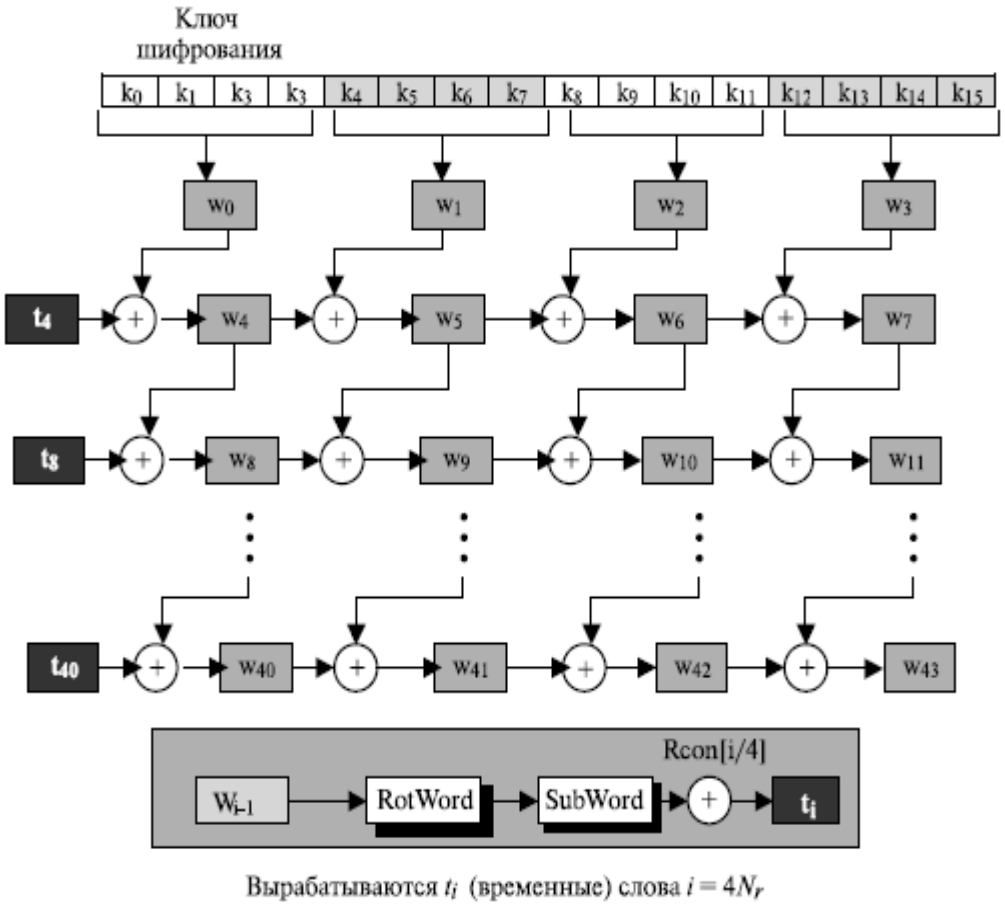


Рисунок 2.18 – Расширение ключа

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 2.19.

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 2.19 – Таблица данных $RCon$

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

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	*54*	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 2.20.

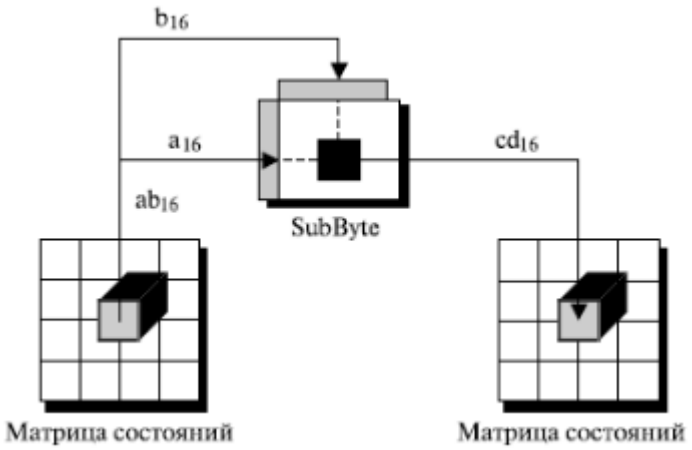


Рисунок 2.20 – Функция *SubBytes*

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*

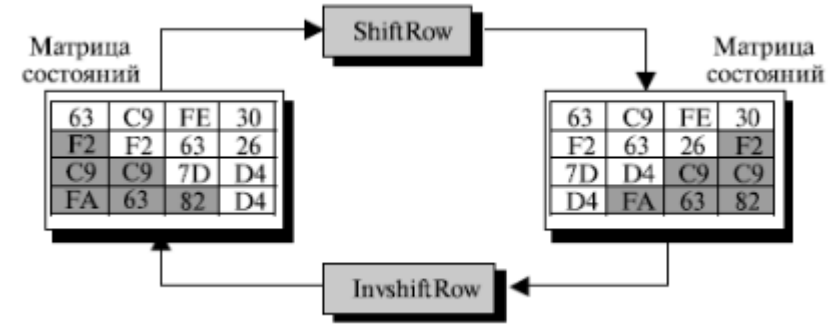
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
5	6C	70	48	50	FD	ED	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	*EA*	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 В.

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.*



c

Рисунок 2.21 – Функции *ShiftRows* и *InvShiftRows*

Листинг функций *ShiftRows* и *InvShiftRows* представлен в приложении Г.

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:

Листинг данных функций представлен в приложении Д.

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 2.22 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
6	90 35 13 60	18 OA B9 B5	86 66 C1 32
	2C FB 82 3A	64 68 6A FB	90 1C 03 ID
	9E *FC* 61 ED	*5A* EF D7 79	OB 8D OA 82
	49 39 CB 47	8E B2 10 4D	95 23 38 D9
7	18 OA B9 B5	01 63 F1 96	62 04 C5 F7
	64 68 6A FB	55 24 3A 62	83 9F 9C 81
	5A EF D7 79	F4 8A DE 4D	3E B3 B9 3B
	8E B2 10 4D	CC BA 88 03	B6 95 AD 74
8	01 63 F1 96	2A 34 D8 46	EE *EA* 2F D8
	55 24 3A 62	2D 6B A2 D6	61 *FE* 62 E3
	F4 8A DE 4D	51 64 CF *5A*	AC IF A6 9D
	CC BA 88 03	87 A8 F8 28	DE 4B E6 92
9	2A 34 D8 46	OA D9 Fl 3C	E4 OE 21 F9
	2D 6B A2 D6	95 63 9F 35	3P Cl A3 40
	51 64 CF *5A*	2A 80 29 00	E3 *FC 5A* C7
	87 A8 F8 28	16 76 09 77	BF F4 12 80
10	OA D9 Fl 3C	BC EO 55 E6	*DB* D5 F4 OD
	95 63 9F 35	02 E3 OD Fl	F9 38 9B *DB*
	2A 80 29 00	8B Bl 6D 82	2E D2 88 4F
	16 76 09 77	D3 95 F8 41	26 D2 CO 40

Рисунок 2.22 – Пример шифрования

На рисунке 2.23 представлен пример 7-го раунда.

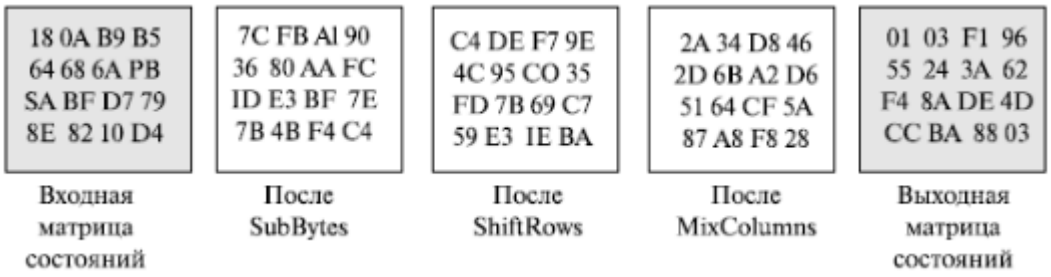


Рисунок 2.23 – 7-й раунд шифрования из примера

Моделирование алгоритма AES-128

Согласно разработанному программному коду алгоритма (приложение А), была разработана соответствующая модель в *Simulink*. На рисунке 2.24 представлена модель AES-128 в режиме ECB.

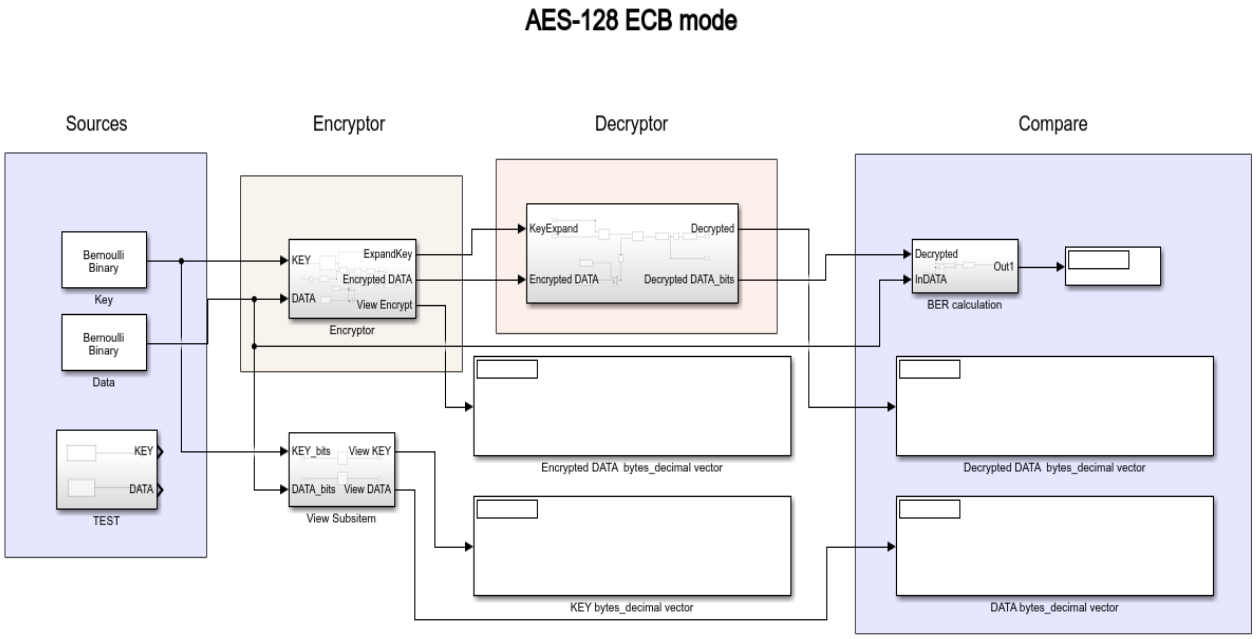


Рисунок 2.24 – Модель AES-128

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 2.25 представлены параметры блока *Bernoulli Binary Generator*.

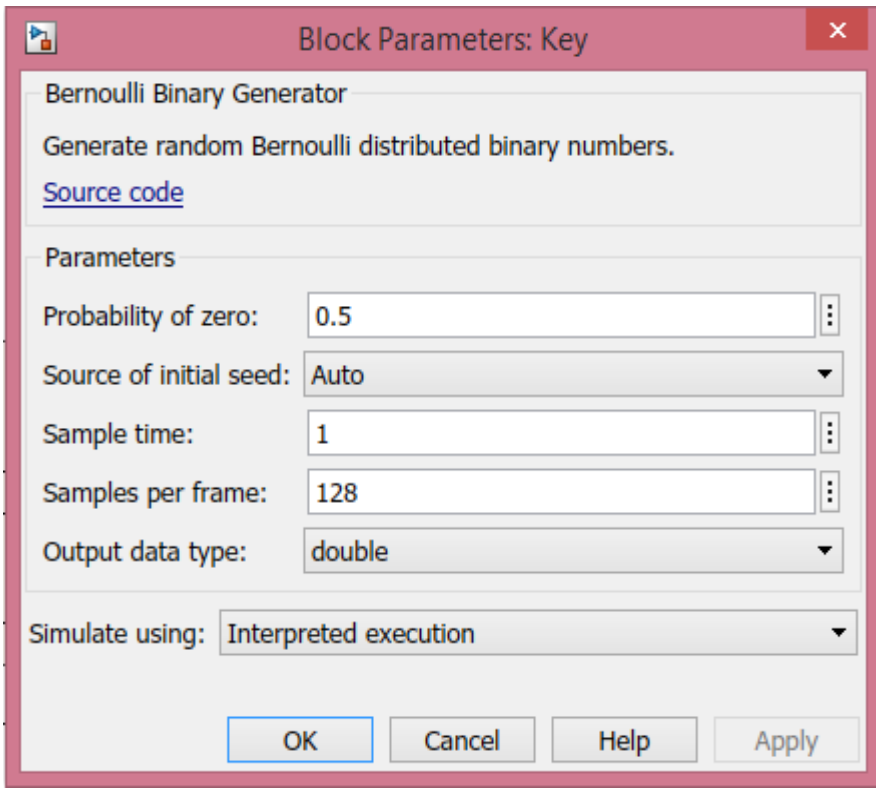


Рисунок 2.25 – Параметры блока Bernoulli Binary Generator

Подсистема шифратора (*Encryptor*) представлена на рисунке 2.26.

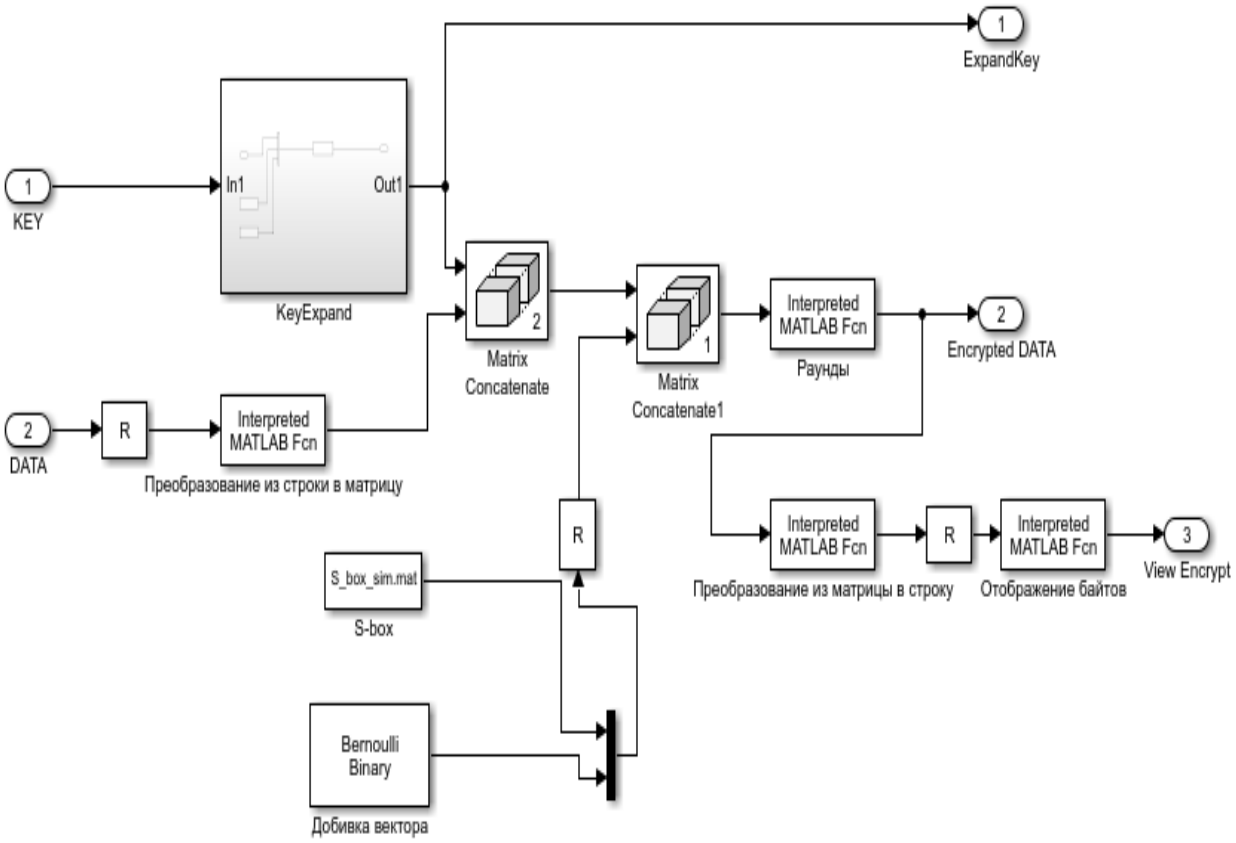


Рисунок 2.26 – Подсистема Encryptor

Подсистема дешифратора (*Decryptor*) представлена на рисунке 2.27.

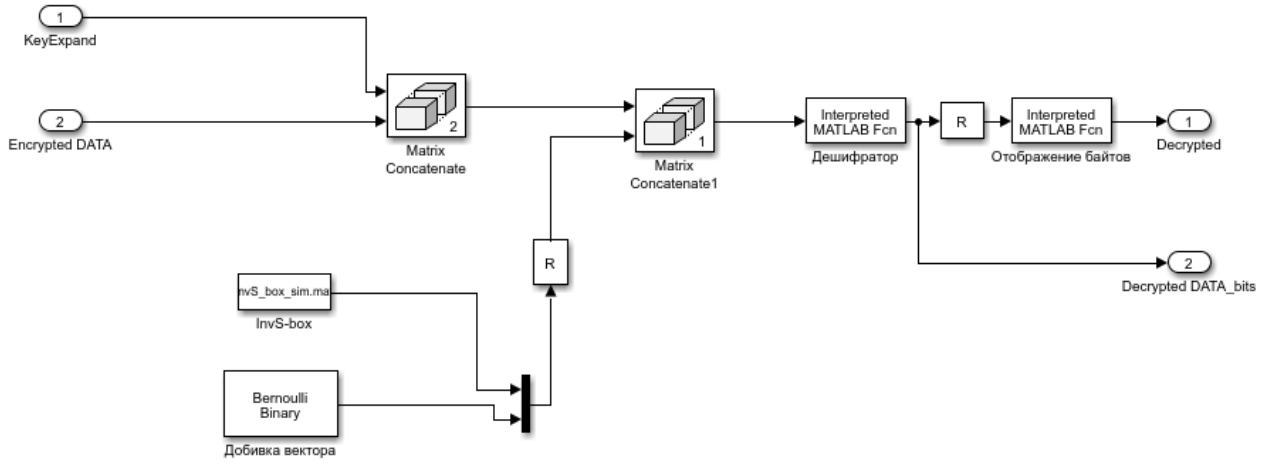


Рисунок 2.27 – Подсистема Decryptor

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 2.28 представлены результаты работы модели.

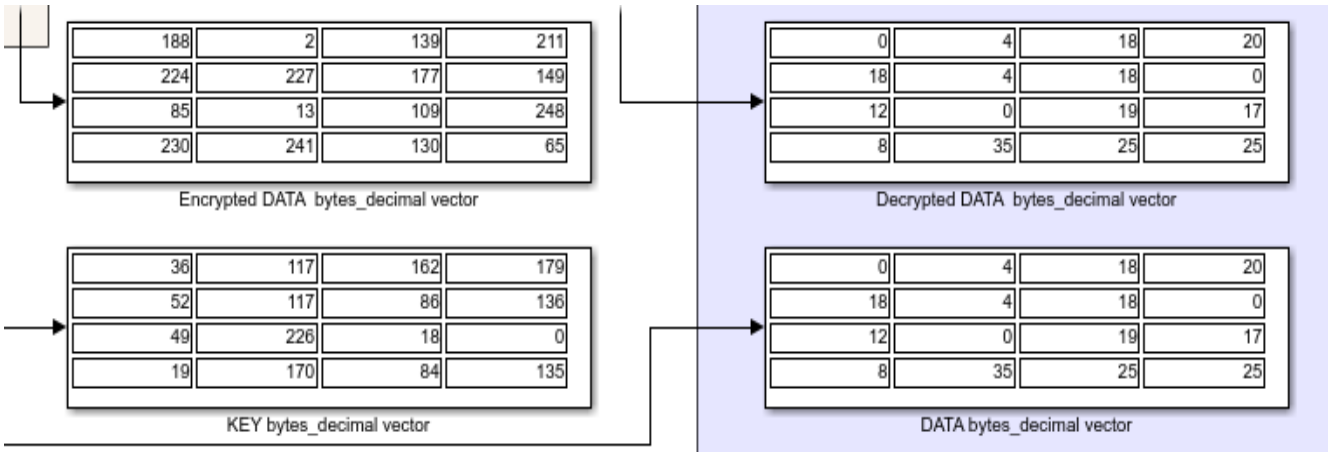


Рисунок 2.28 – Результаты моделирования

В таблице 2.5 приведены результаты измерений зависимость времени шифрования и дешифрования от объема файла для расшифровки:

Таблица 2.5 Результаты измерений

Объем файла, Б	1	2	5	10	65	180	400	500
Время шифрования, с	0,043	0,038	0,042	0,038	0,112	0,28	0,67	0,78
Время дешифрования, с	0,034	0,033	0,034	0,03	0,132	0,29	0,63	0,76

На рисунке 2.29 представлена зависимость времени шифрования и дешифрования от объема файла для расшифровки:

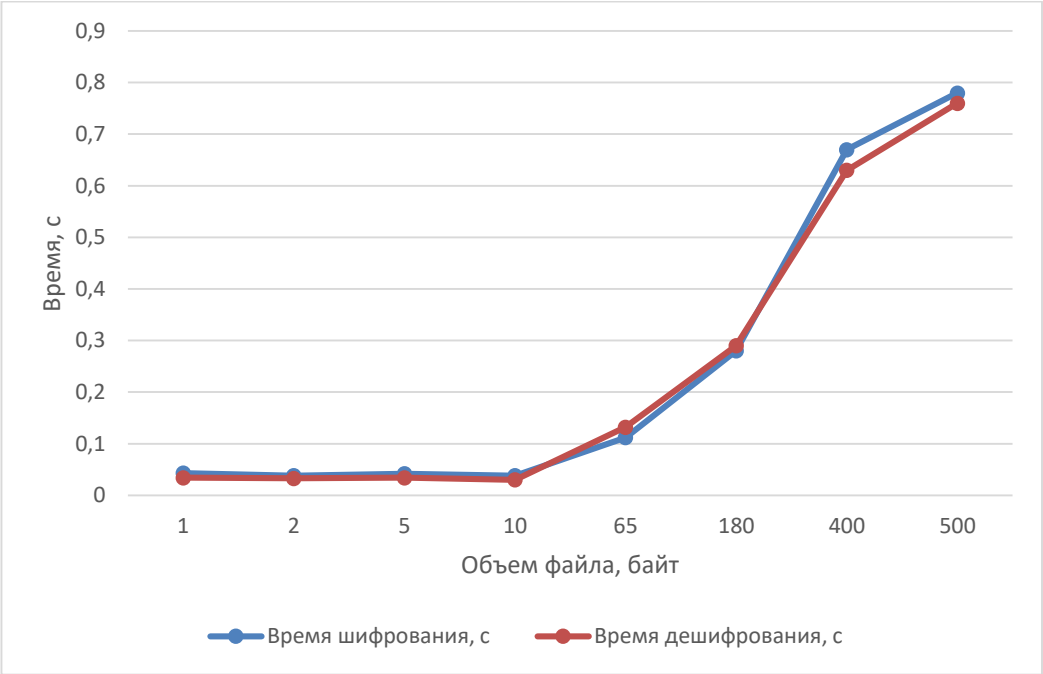


Рисунок 2.29 – зависимость времени шифрования и дешифрования от объема файла для расшифровки

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.

ЗАКЛЮЧЕНИЕ

В учебном пособии представлена защита информации в цифровых системах связи, рассмотрены шифры с открытым ключом, криптографические протоколы в сетях передачи данных на транспортном уровне - SSL и TLS, на сетевом уровне - IP SEC и прикладном уровне – PGP. Представлен компьютерный практикум для шифров с открытым ключом и задание на самостоятельную работу по шифрам с открытым ключом. Рассмотрено шифрование в современных системах связи стандартов GSM и LTE и компьютерный практикум для исследования стандарта LTE в MATLAB.

ЛИТЕРАТУРА

1. Б. Шнайер «Прикладная криптография. 2-е издание. Протоколы, алгоритмы и исходные тексты на языке С». - М.: Изд-во "Триумф", 2002. - 816 с.

2. Романец Ю.В. Защита информации в компьютерных системах и сетях / Под ред. В.Ф. Шаньгина. - 2-е изд., перераб. и доп. - М.: Радио и связь, 2001.

- 376 с.

3. Diffie, D. New directions in Cryptography / D. Diffie, M.Hellman // IEEETransactions on information theory. November. 1976.

4. Rivest, R. A Method for obtaining digital signatures and public keyCryptosystems / R. Rivest, A. Shamir, L. Adleman // Communications of the ACM.February. 1978.

5. Столингс, В. Криптография и защита сетей: принципы и практика / В. Столингс ; пер. с англ. . – 2-е изд. – М. : Изд. дом «Вильямс», 2001. – 672 с.

ПРИЛОЖЕНИЕ

Задания на самостоятельную работу

по криптографическим протоколам

PGP кодированине

Мы разберем подробно схему использования асимметричного шифрования на примере вымышленной консалтинговой компании Forest inc. Она занимается подбором руководителей высшего звена, то есть находит бизнес с потребностью в кадрах и подходящих кандидатов из других компаний. Классический хедхантинг:

- Любая утечка информации на любом из этапов приведет к весьма драматическим последствиям для всех сторон.
- В Forest inc. работают три консультанта: Винни Пух, Сова и Пятачек.
- Винни Пух узнал, что Тигра ищет руководителя по развитию бизнеса и уже едет к нему на встречу, чтобы познакомиться, снять запрос и провести пресейл.
- В это время Сова в офисе изучает информацию о Tigra Co. в надежде найти информацию, которая поможет Винни Пуху заключить сделку.
- Винни Пух приезжает в офис к Тигре, а Сова к тому времени выяснила, что Тигра очень любит прыгать – это очень важный факт, который нужно безопасно сообщить Винни.
- Необходимо проинформировать Пятачка, который находится в командировке в дальнем лесу: он занимается подбором топ-менеджеров и это очень поможет ему в поиске.
- У Винни телефон на Android, Сова в офисе за ноутбуком с Windows, а Пятачек, как порядочная свинья, пользуется айфоном.

Настройка программ

Начнем с ноутбука Совы. Будем использовать программу gpg4win: для начала скачайте с сайта (https://www.gpg4win.org/)дистрибутив.

Рисунок П.1 – Сайт для загрузки ПО

Запустите установку.

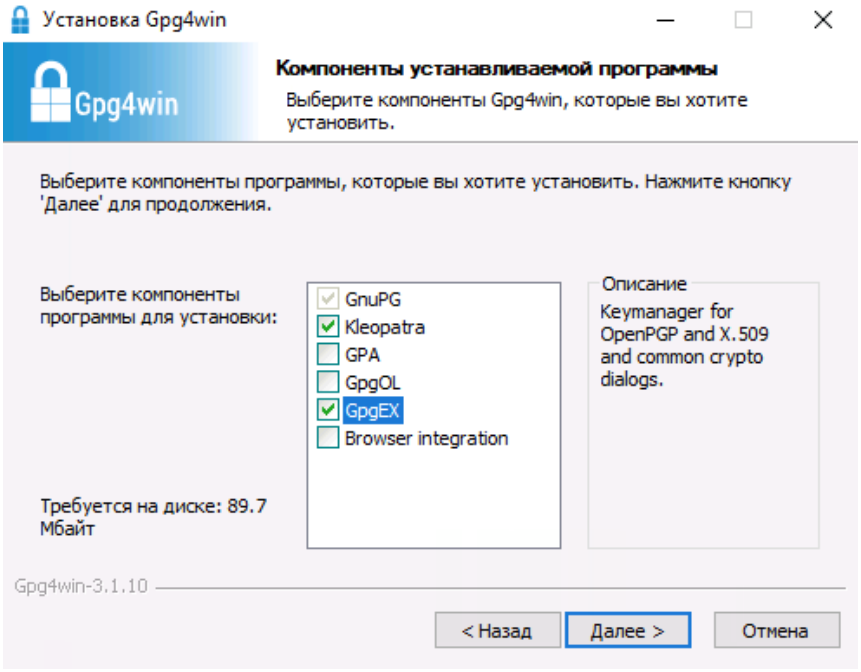


Рисунок П.2 – Установка gpg4win

Нам важны только GnuPG и Kleopatra. Уберите GpgOL, а GpgEX можно оставить.

На рабочем столе появится иконка Kleopatra, а после запуска программы вы увидите такое окно:

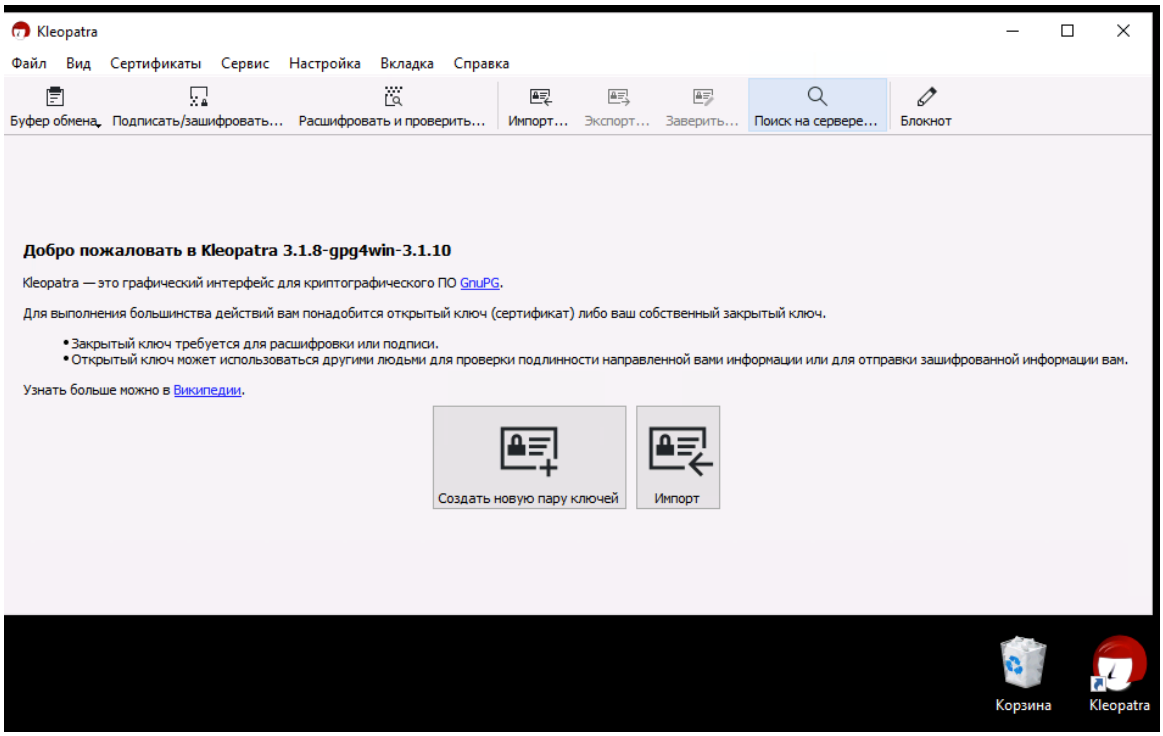


Рисунок П.3 – Окно программы Kleopatra

Сразу сделайте две вещи: добавьте кнопку Буфер обмена на панель и включите постоянное отображение ярлыка в трее у часов.

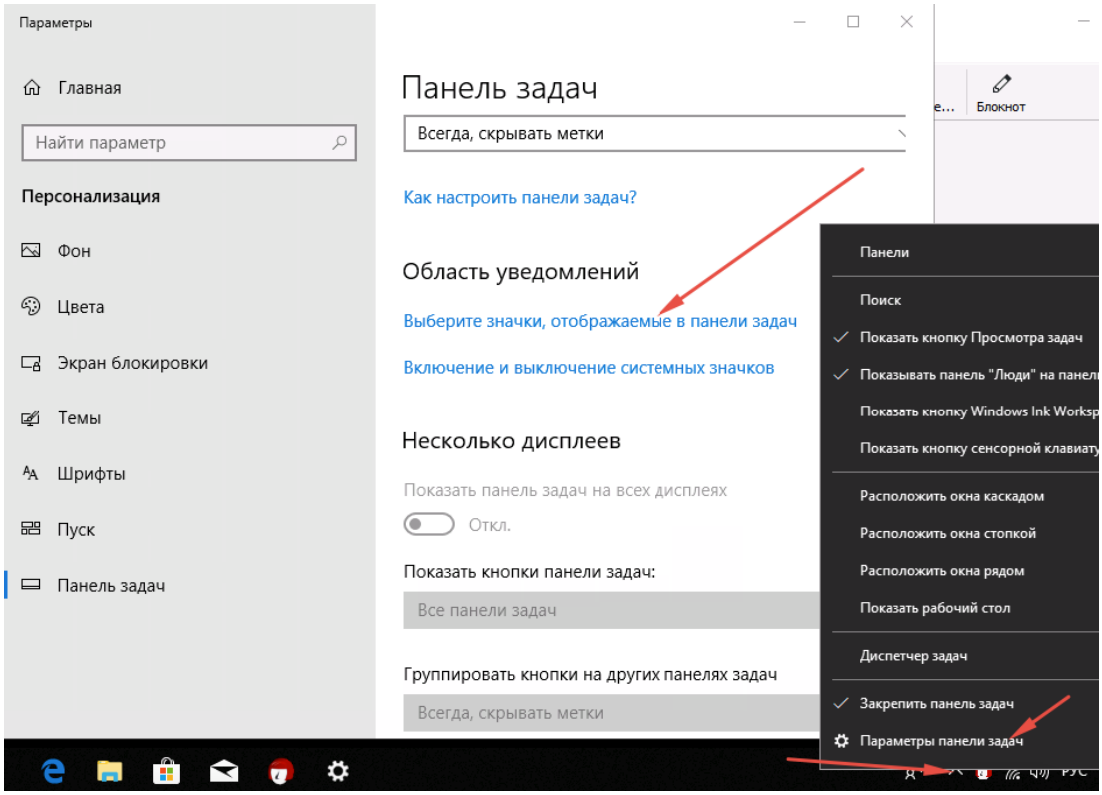


Рисунок П.4 – Правой кнопкой на стрелочке

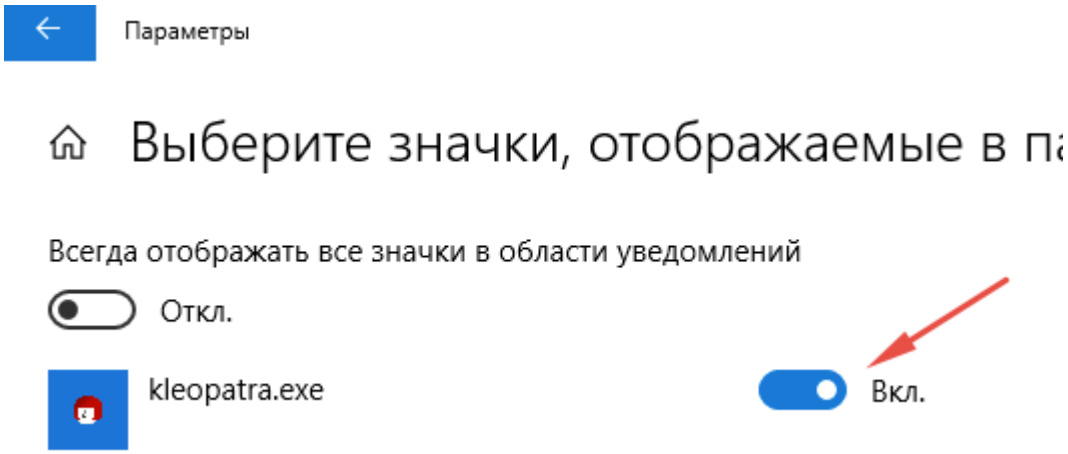


Рисунок П.5 – Включите ползунок

Теперь идите в Клеопатру и вытаскивайте кнопку Буфер обмена на панель:

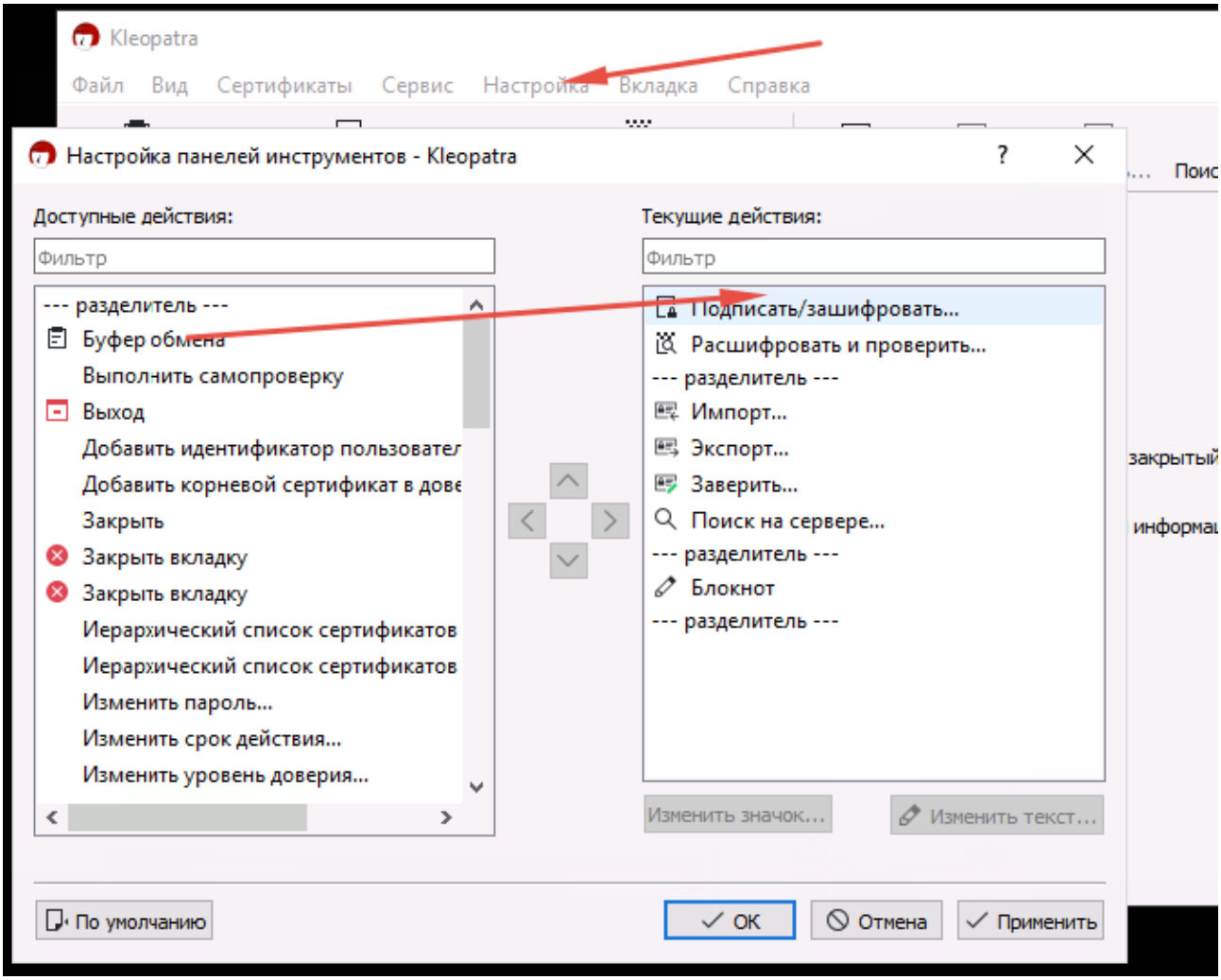


Рисунок П.6 – Перетащите Буфер обмена из правой колонки в левую

Можно добавить Клеопатру в автозагрузку: для этого нажмите Пуск → Выполнить или используйте сочетание клавиш Win+R, а потом напишите shell:startup и нажмите Enter:

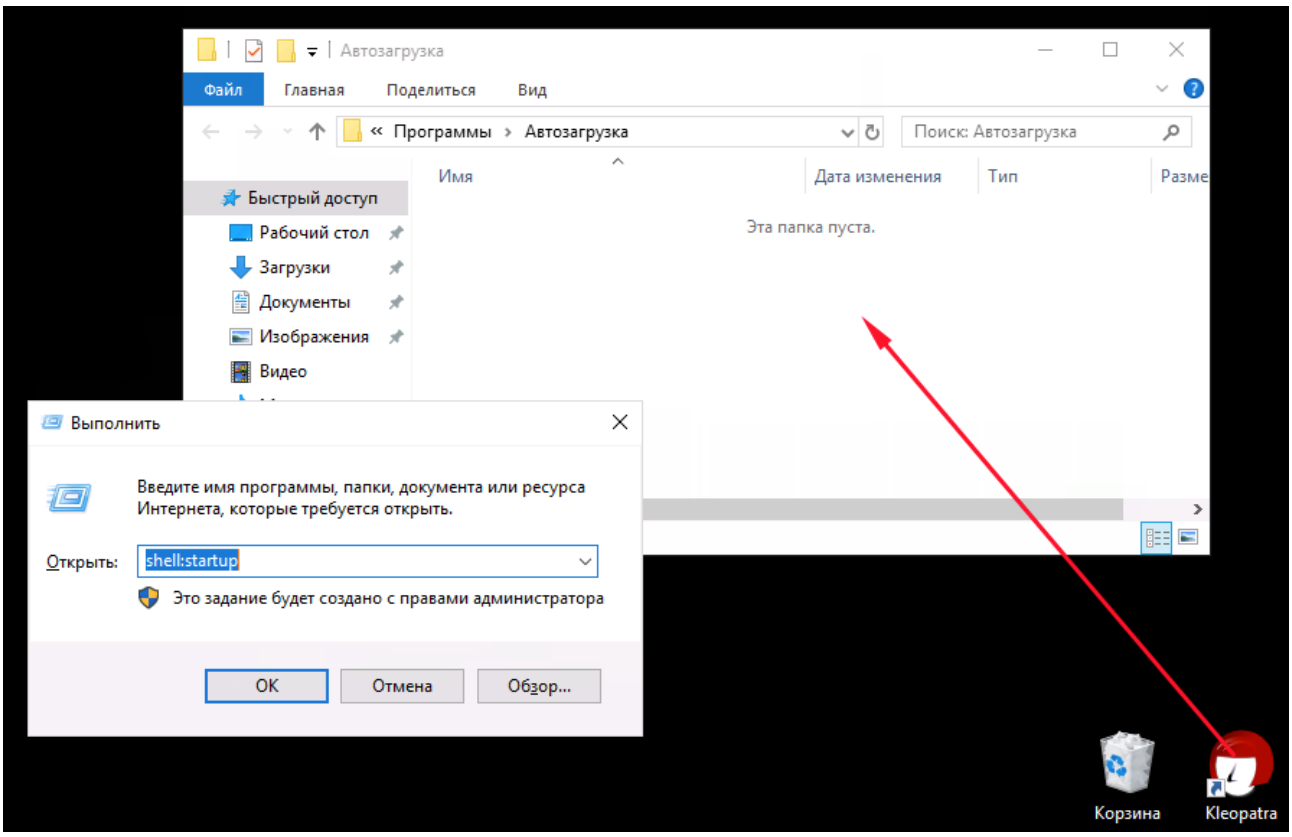


Рисунок П.7 – Правой кнопкой перетащите ярлык Клеопатры в папку Автозагрузка и выберите Копировать

Настройка программы закончена. Теперь время создать ключи и обменяться ими с коллегами.

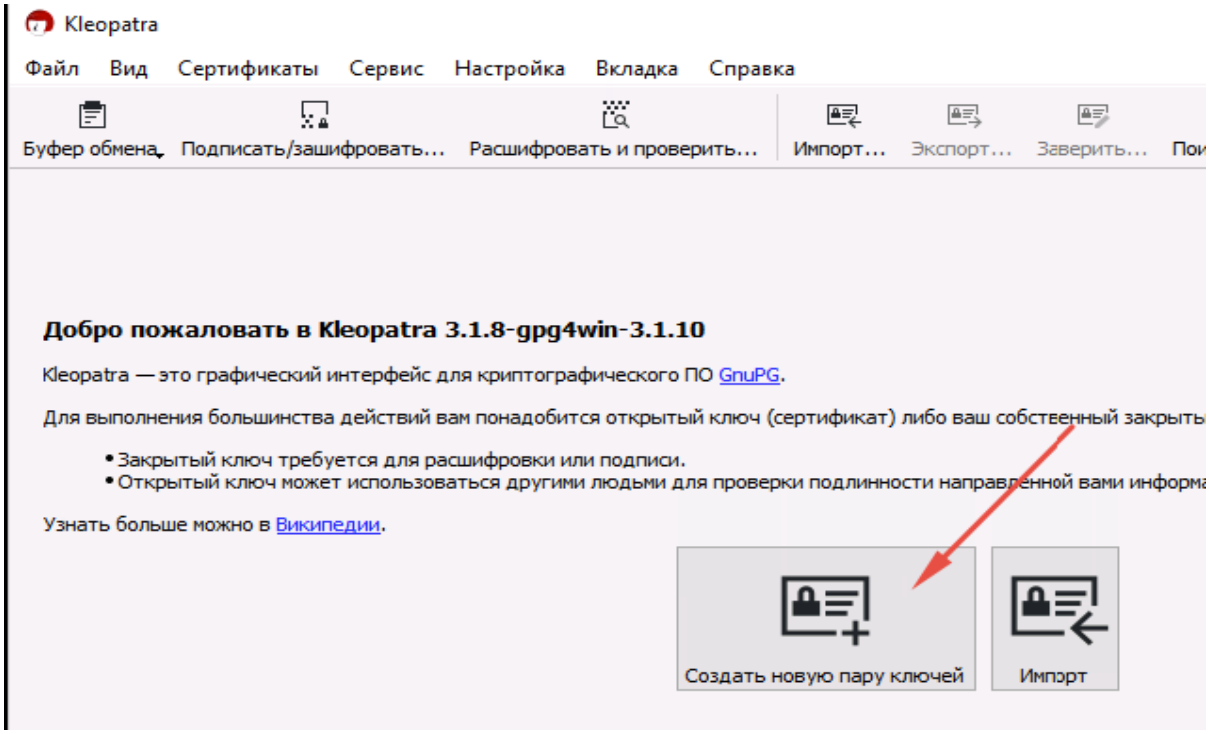


Рисунок П.8 – Нажмите Создать пару ключей

Важно понимать, что публичный ключ можно публиковать в открытом виде. Это безопасно и позволяет вам начать шифрованную переписку с кем угодно, однако в публичном ключе содержатся поля Имя и E-mail. Лучше всего написать там вымышленные данные. Электронная почта никакого значения не имеет, а имя ключа ваши адресаты увидят в собственных списках. Достаточно, чтобы они опознали владельца ключа.

Перейдите в Дополнительные параметры:

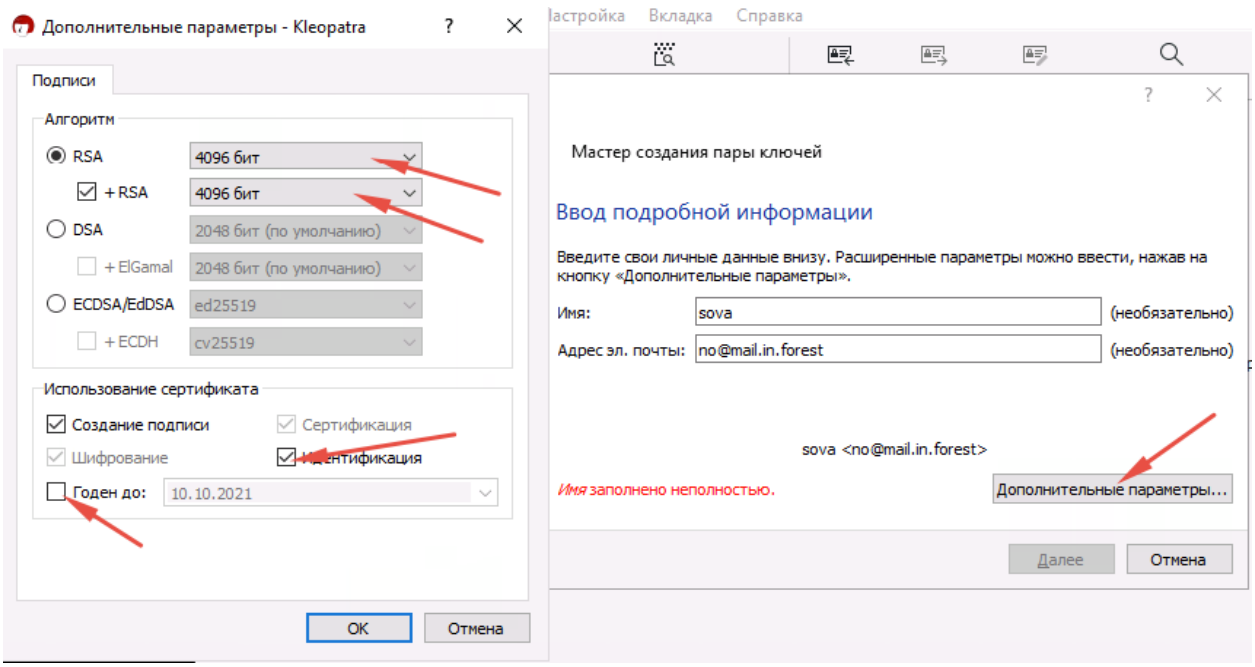


Рисунок П.9 – Укажите максимальный размер ключа, уберите срок его годности и поставьте галочку на Идентификации

Для создания ключа все готово. Нажмите «Далее», далее и задайте пароль:

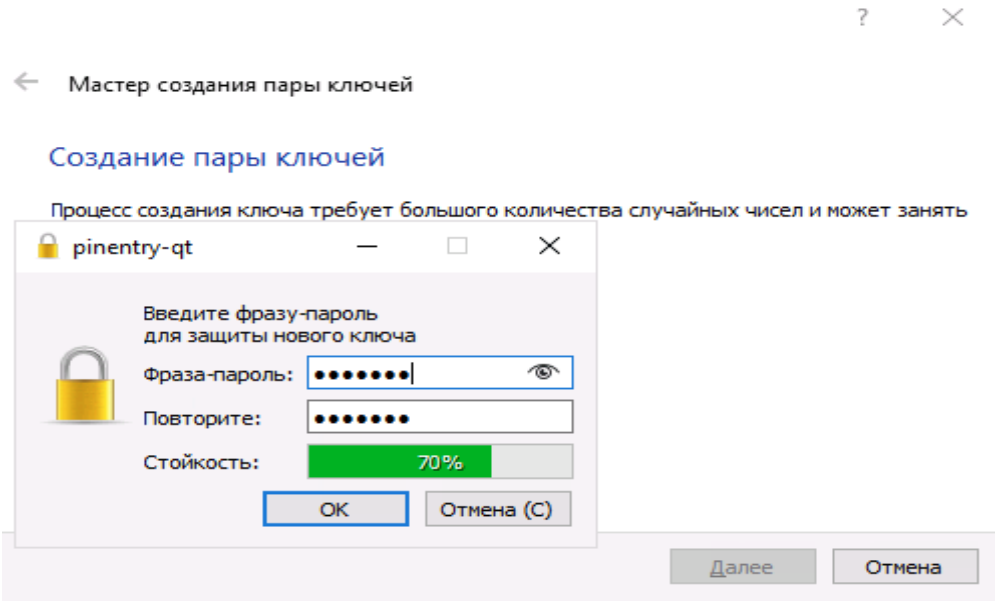


Рисунок П.10 – Мастер создания пары ключей

Пароль нужен для импорта и экспорта секретного ключа – обязательно запомните его или запишите. Сам пароль никаких сообщений не расшифровывает, но он потребуется для переноса ключа на другие устройства.

Подождите:

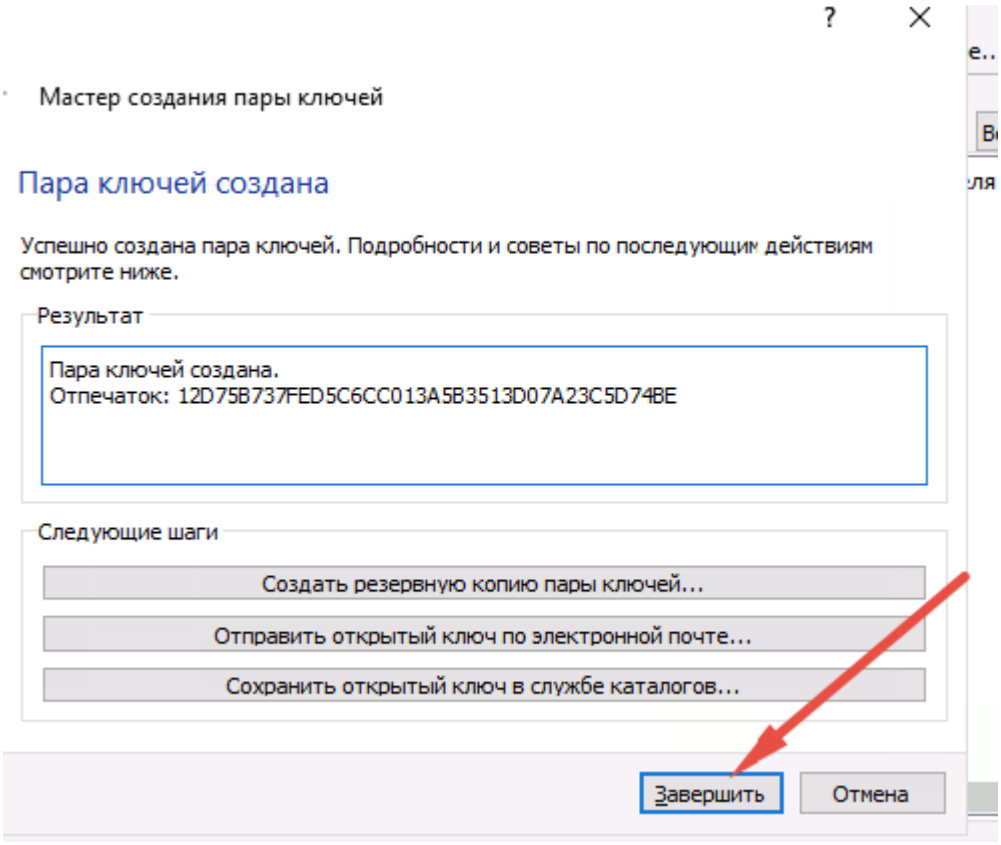


Рисунок П.11 – Нажмите Завершить

Все, ключи созданы:

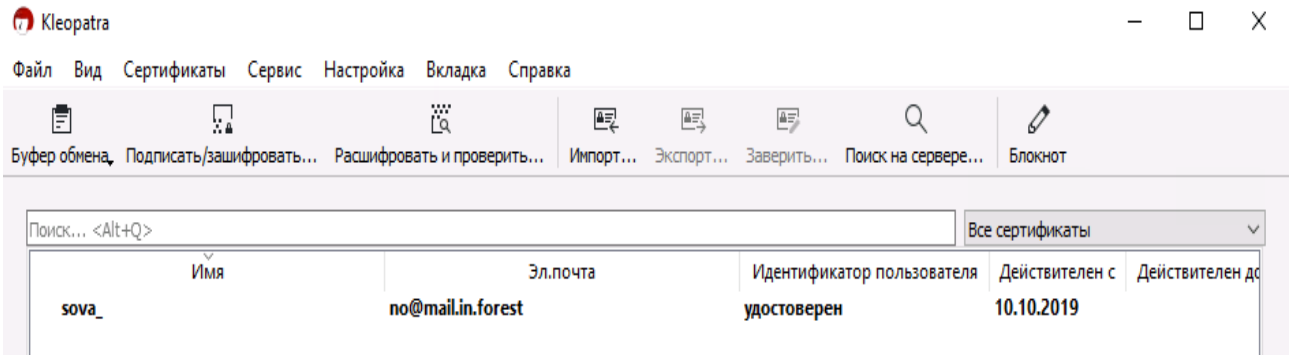


Рисунок П.12 – Окно программы Kleopatra

Теперь самое время обменяться ключами с Винни Пухом. Для этого нужно послать ему наш публичный ключ, получить ключ от него и импортировать в Клеопатру.

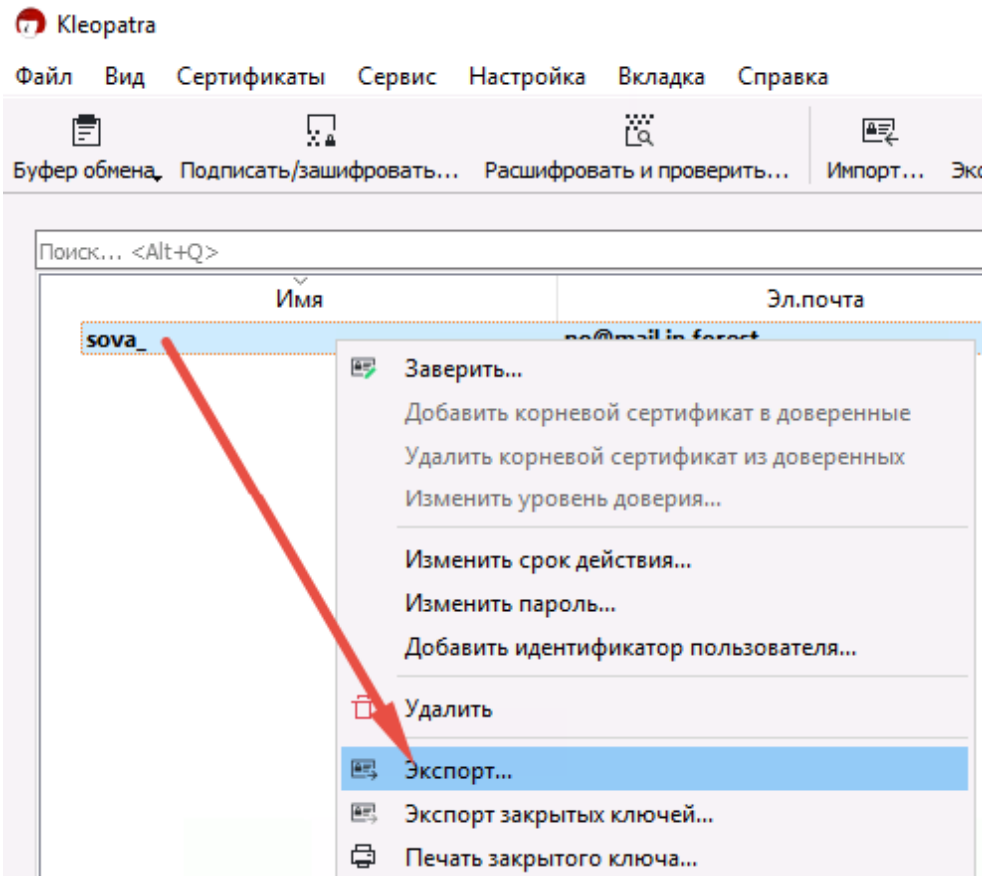


Рисунок П.13 – Правой кнопкой на ключе – Экспорт

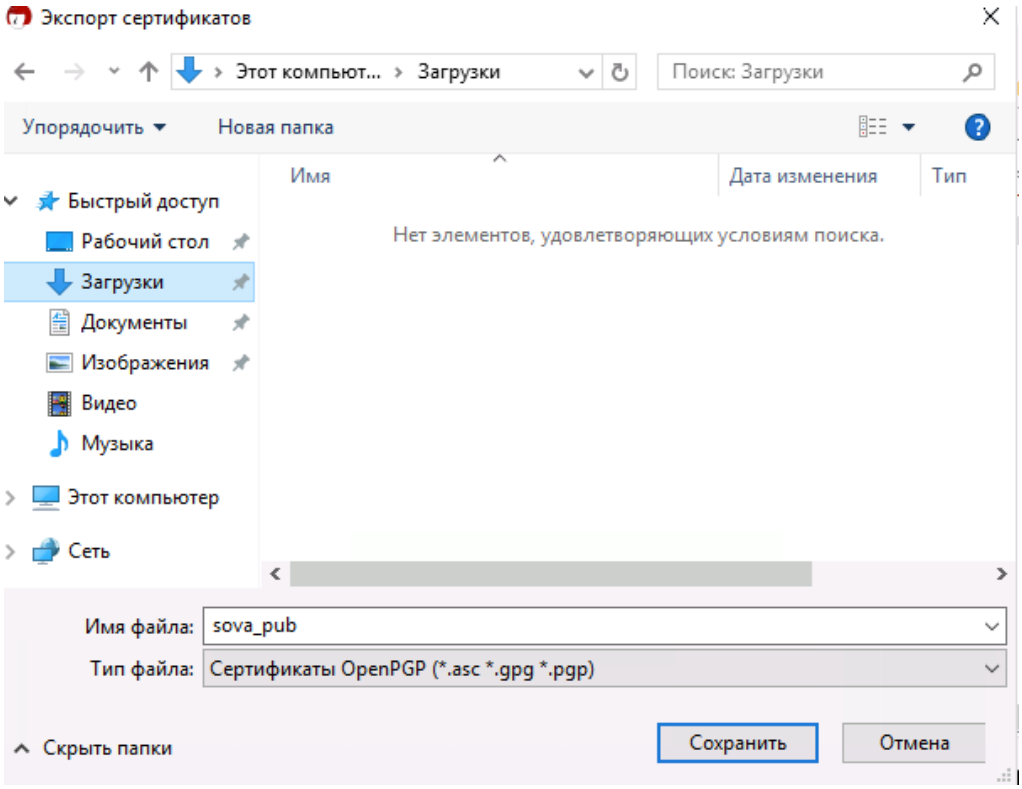


Рисунок П.14 – Сохраните файл куда-нибудь, например, в загрузки

Теперь в папке Загрузки появился текстовый файл с расширением .asc, который содержит наш публичный ключ.

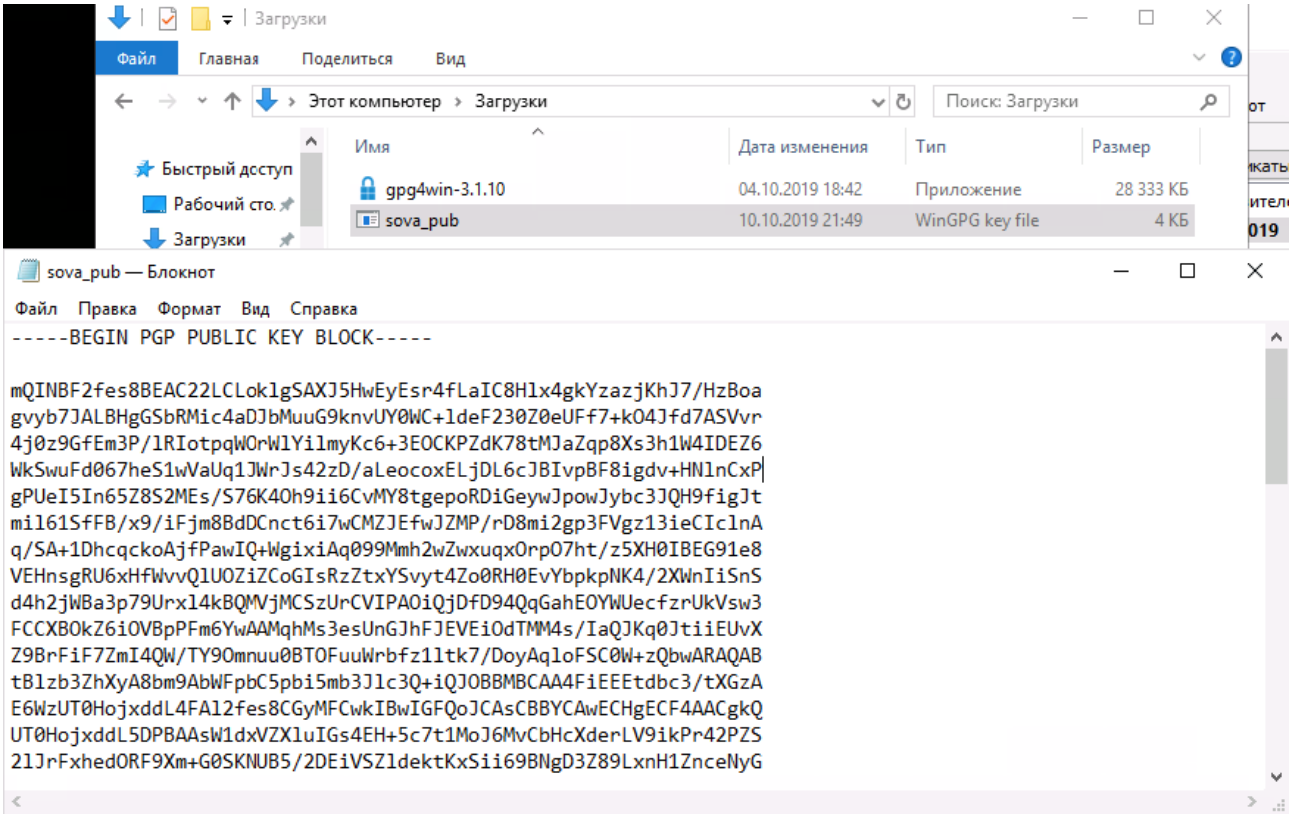


Рисунок П.15 – Так выглядит содержимое ключа

Мы помним, что Винни Пух сидит на Андроиде. Сова сообщает ему, что необходимо установить из Google Play приложение PGPtools.

Как правильно отправлять зашифрованные данные

Для обмена зашифрованными текстами лучше всего использовать сервисы одноразовых записок.

Это хорошо по нескольким причинам:

- Они анонимны.
- Записка удаляется сразу после открытия, так что, если кто-то решит прочесть ссылку раньше адресата, тот сразу узнает об утечке.
- В истории мессенджера сохраняется не переписка, а ссылки на уничтоженные записки. Если ключи украдут, расшифровать старые сообщения будет невозможно, к тому же это удобная страховка от взлома мессенджеров.

- Хранить зашифрованные данные на сервисе безопасно, поскольку ключей у сервиса нет (в отличие от мессенджеров, которые предлагают свое шифрование).Обмен публичными ключами между Android и Windows

Откройте файл с ключом в текстовом редакторе и скопируйте его содержимое в буфер:

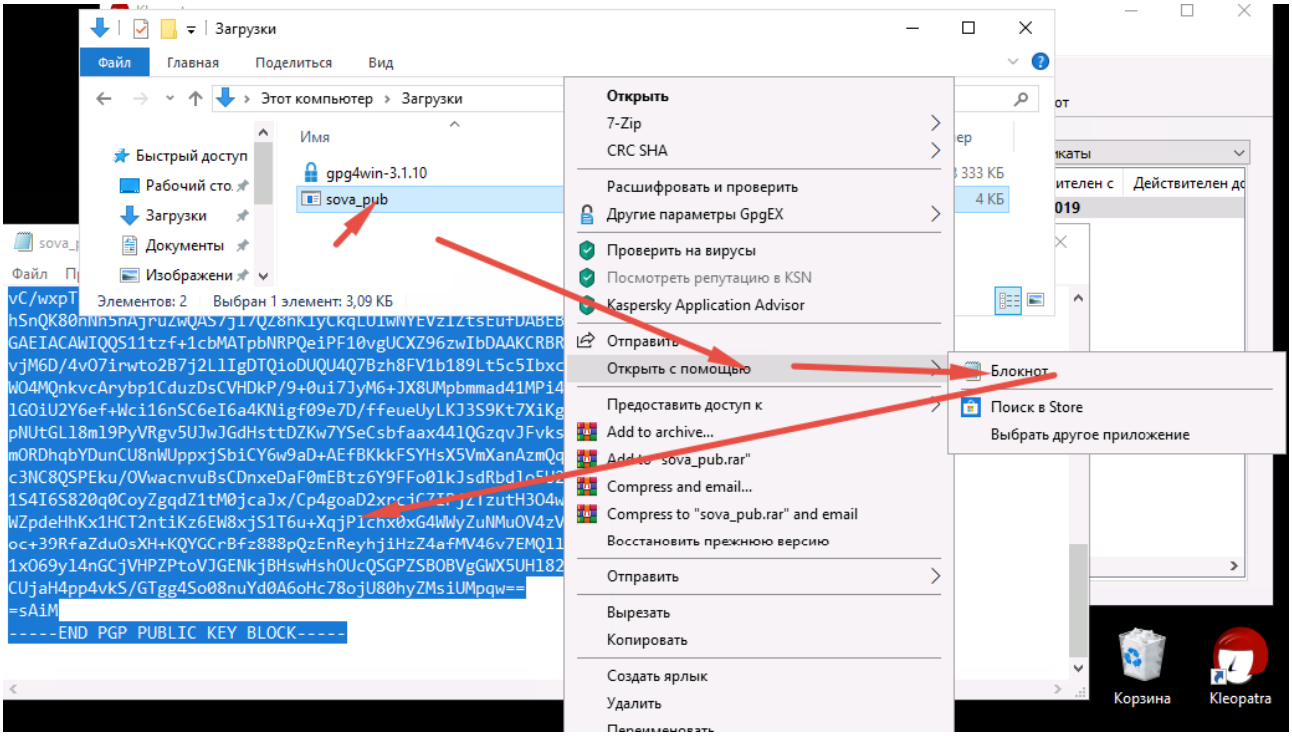


Рисунок П.17 – Окна программ

Затем на сайте privnote.com вставьте текст ключа в записку и отправьте ссылку на нее Винни Пуху по WhatsApp. Винни Пух ее получает, далее скриншоты с его мобильника:

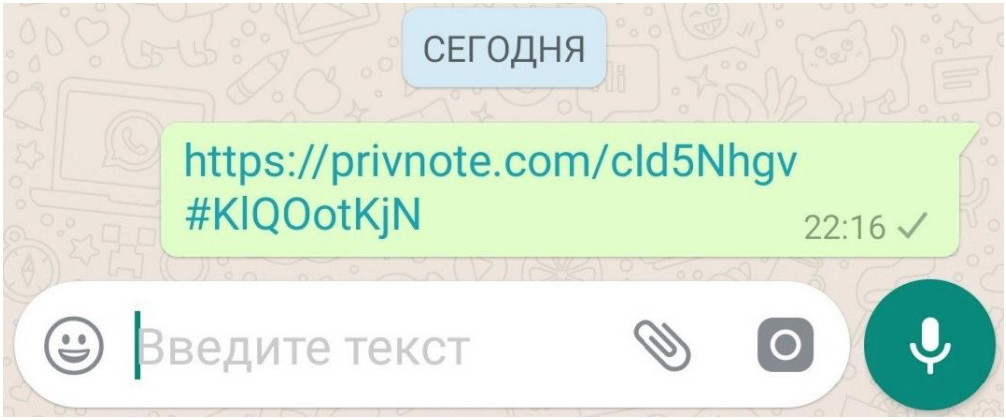


Рисунок П.18 – Перейдите по ссылке

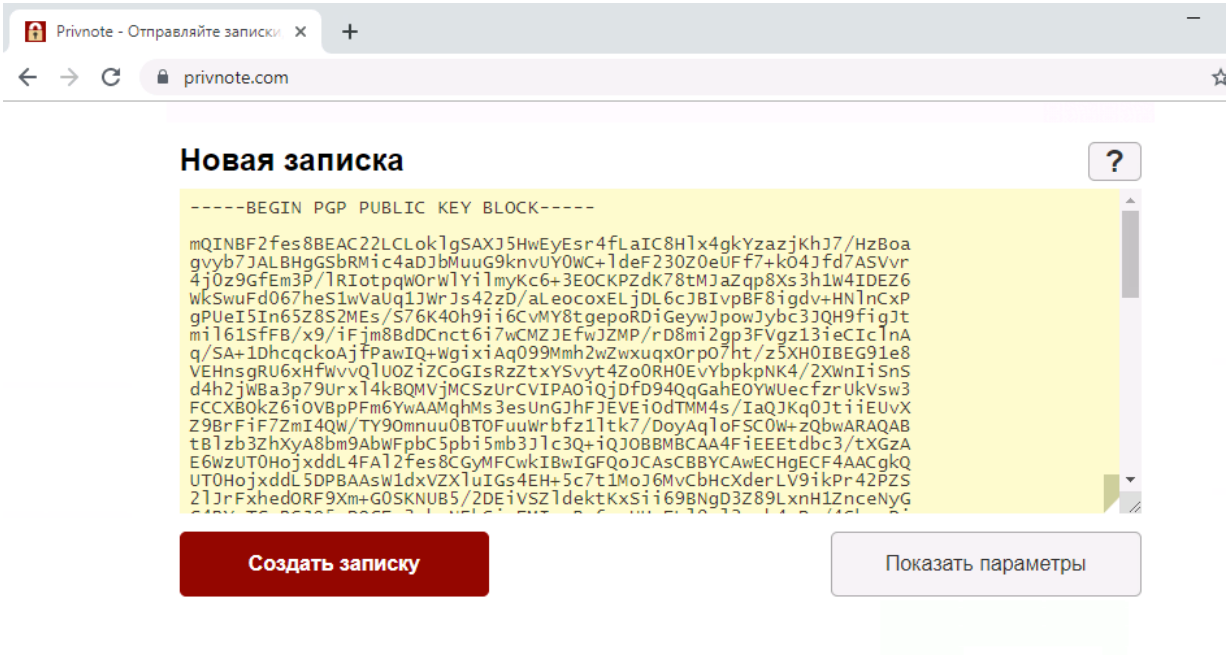


Рисунок П.19 – Ключ

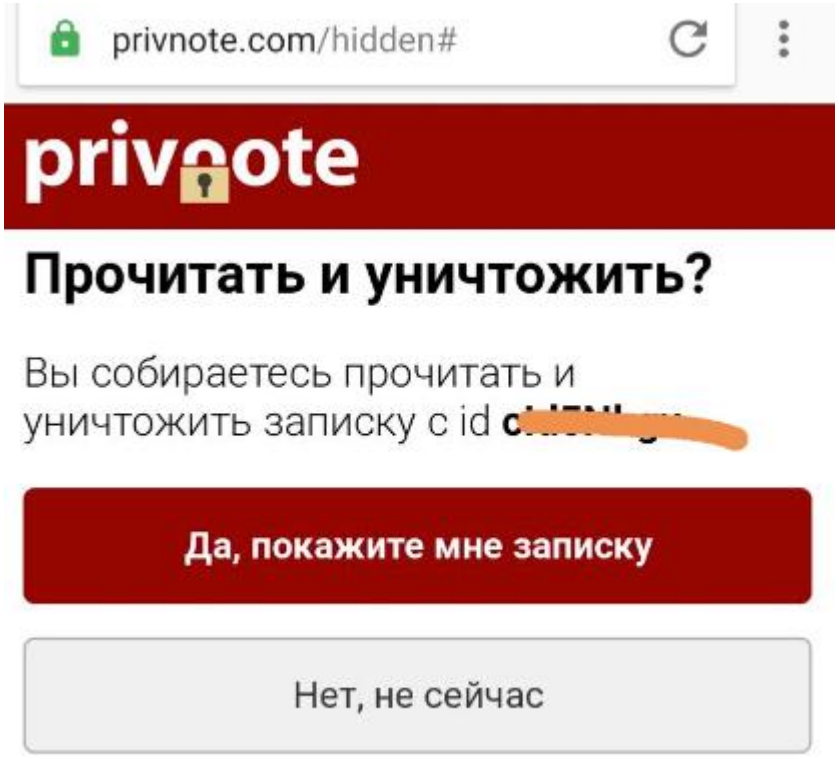
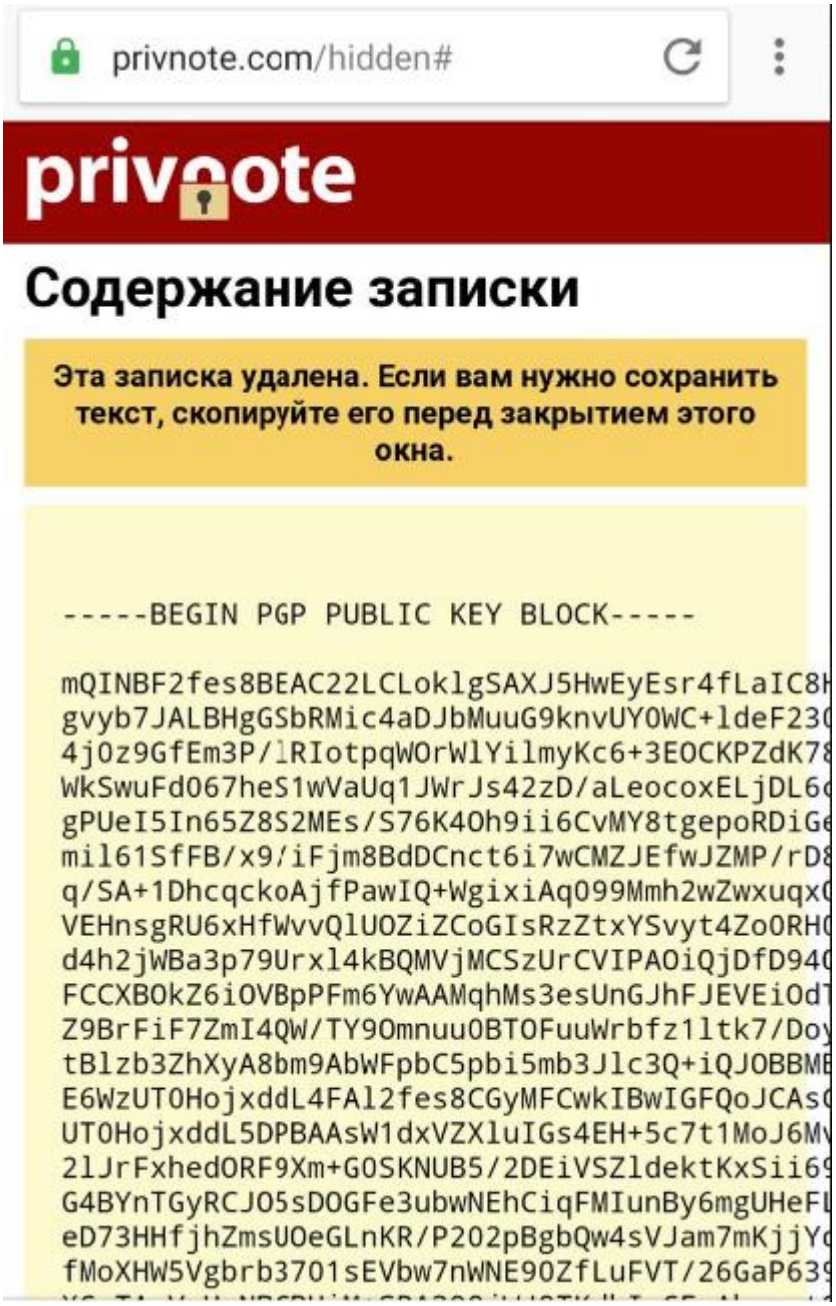


Рисунок П.20 – Откройте записку



Откройте PGPtools и нажмите Import:

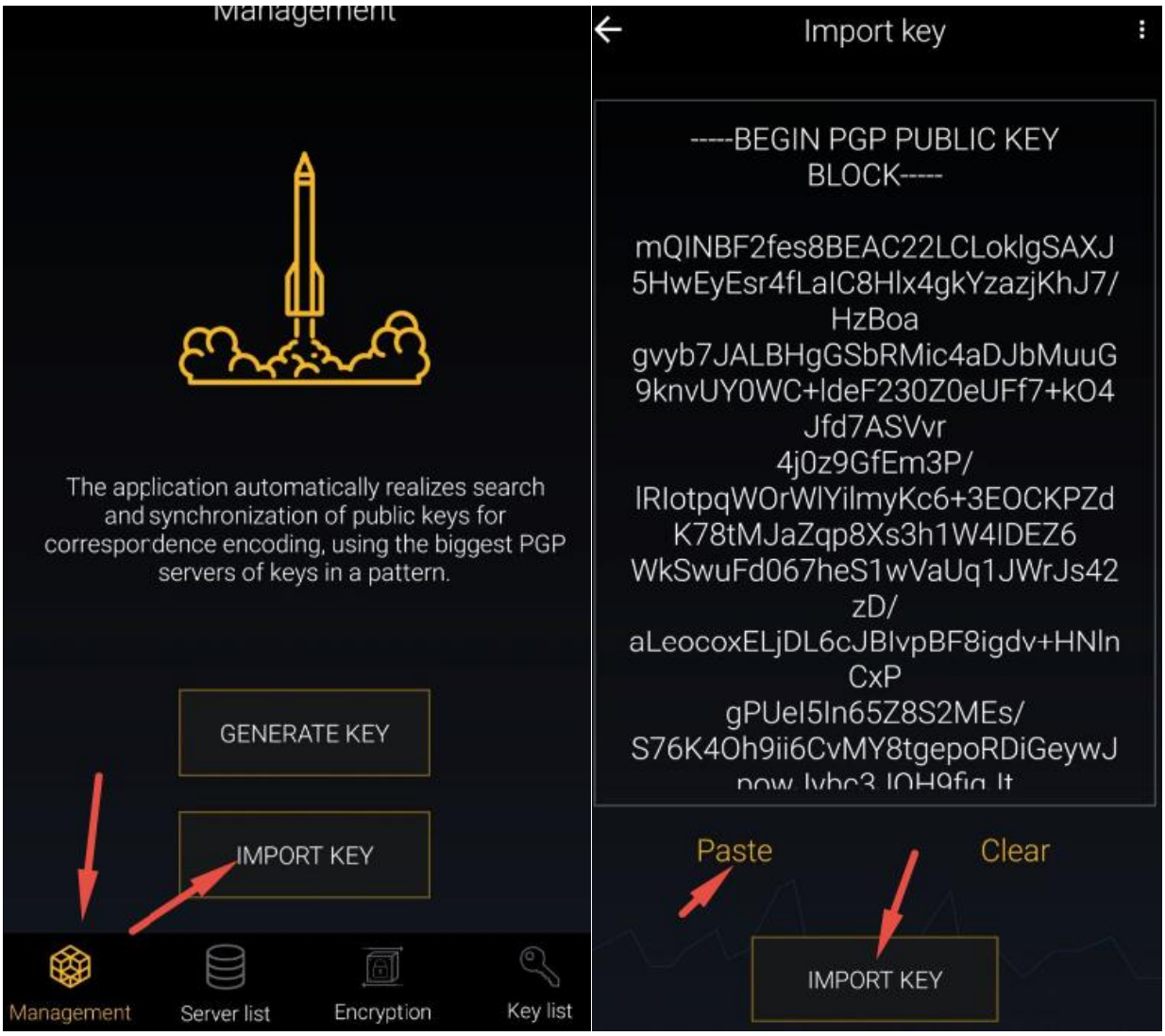


Рисунок П.30 – Окна программы PGP

Вставьте ключ в окно через кнопку Paste и нажмите Import.

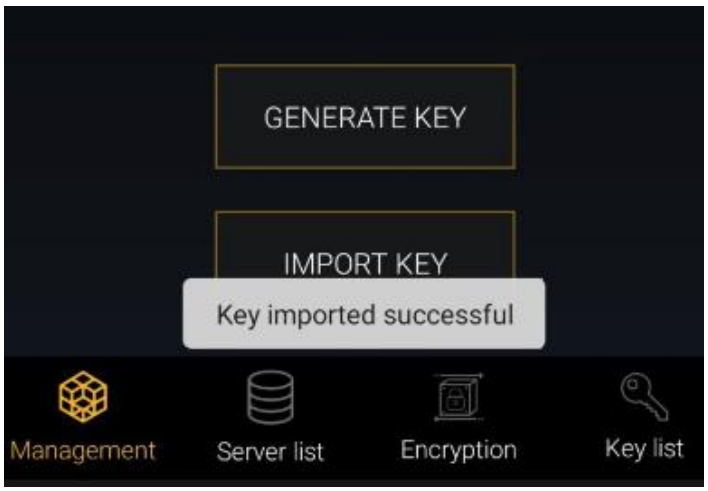


Рисунок П.31 – Импортирование ключа

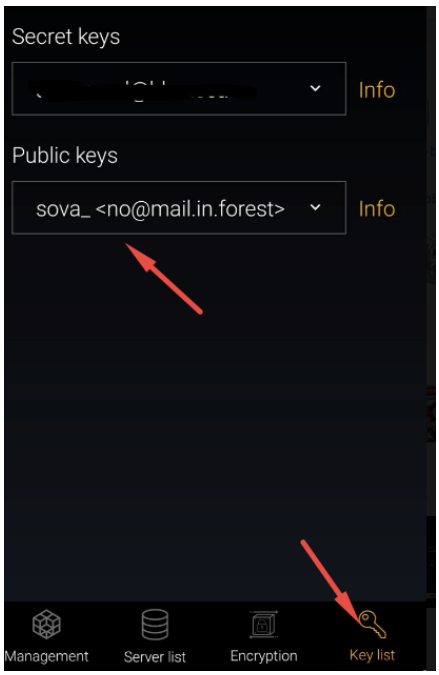


Рисунок П.32 – Ключ успешно импортирован

Убедитесь в этом, нажав Key list (вы должны увидеть в списке ключ Совы).

Теперь Винни Пух должен создать свою пару ключей и отправить Сове публичный.

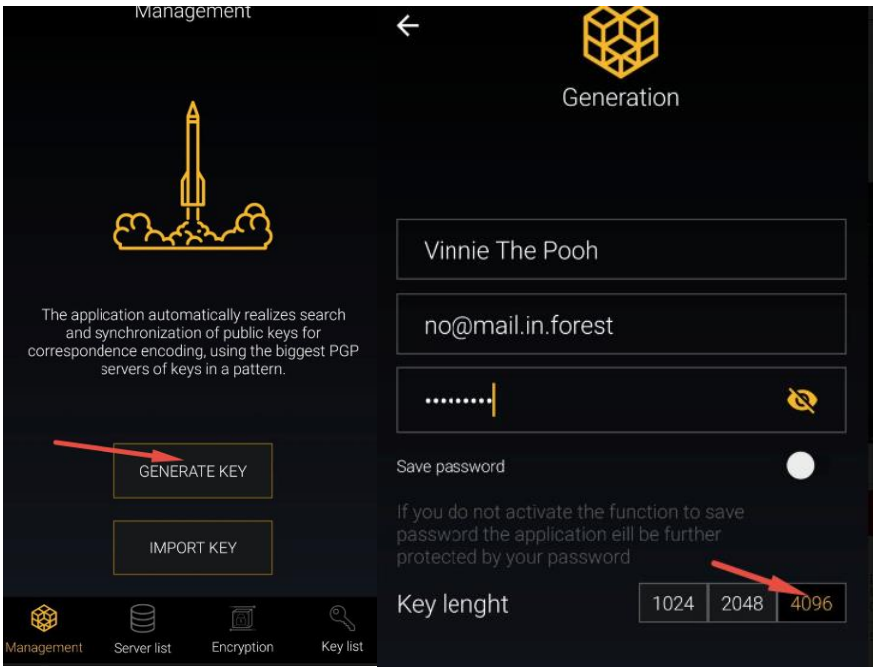


Рисунок П.33 – Нажмите Generate

Выберите максимальную длину ключа, имя и e-mail, задайте пароль, нажмите Generate и подождите секунд 15-20.

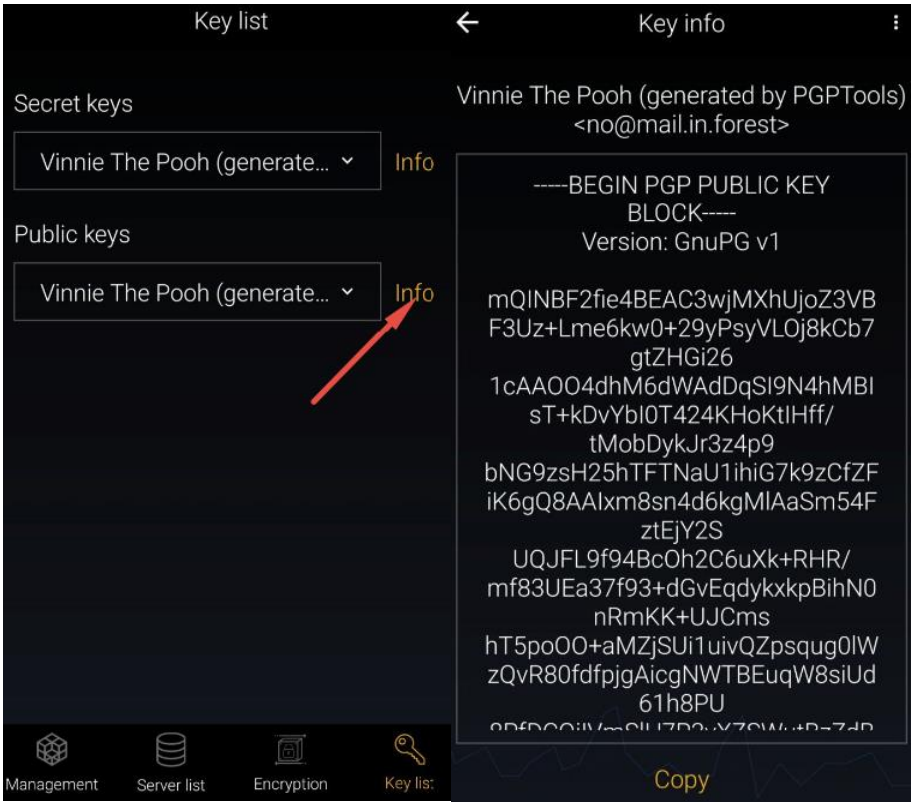


Рисунок П.34 – Окна программы PGP

Ключ создан. Теперь в Key

List нажмите Info на публичном ключе. Нажмите Copy, и ключ будет скопирован в буфер.

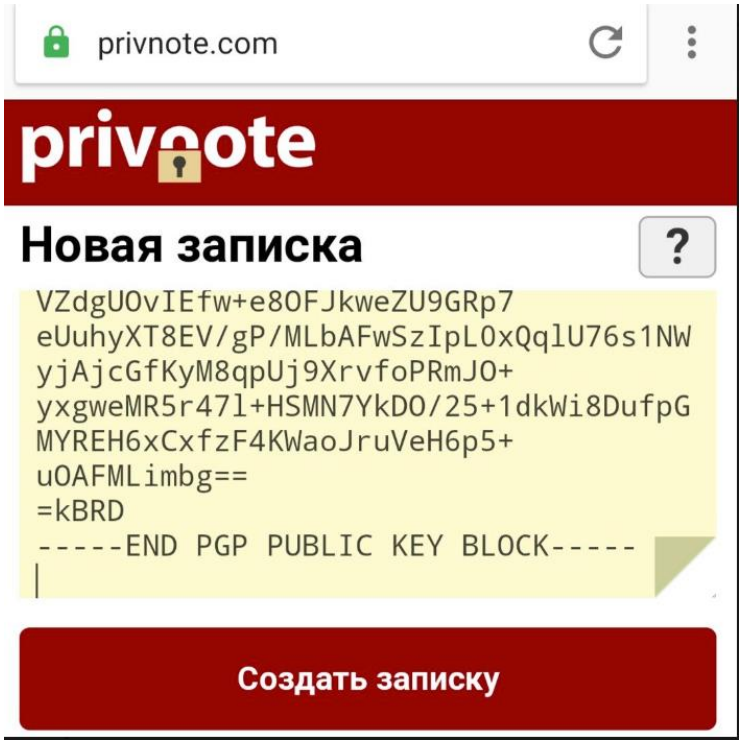


Рисунок П.35 – Вставьте ключ в окно, создайте записку и отправьте Сове

Далее Сова на ноутбуке откроет ссылку с ключом (следующие скрины с ноутбука Совы):

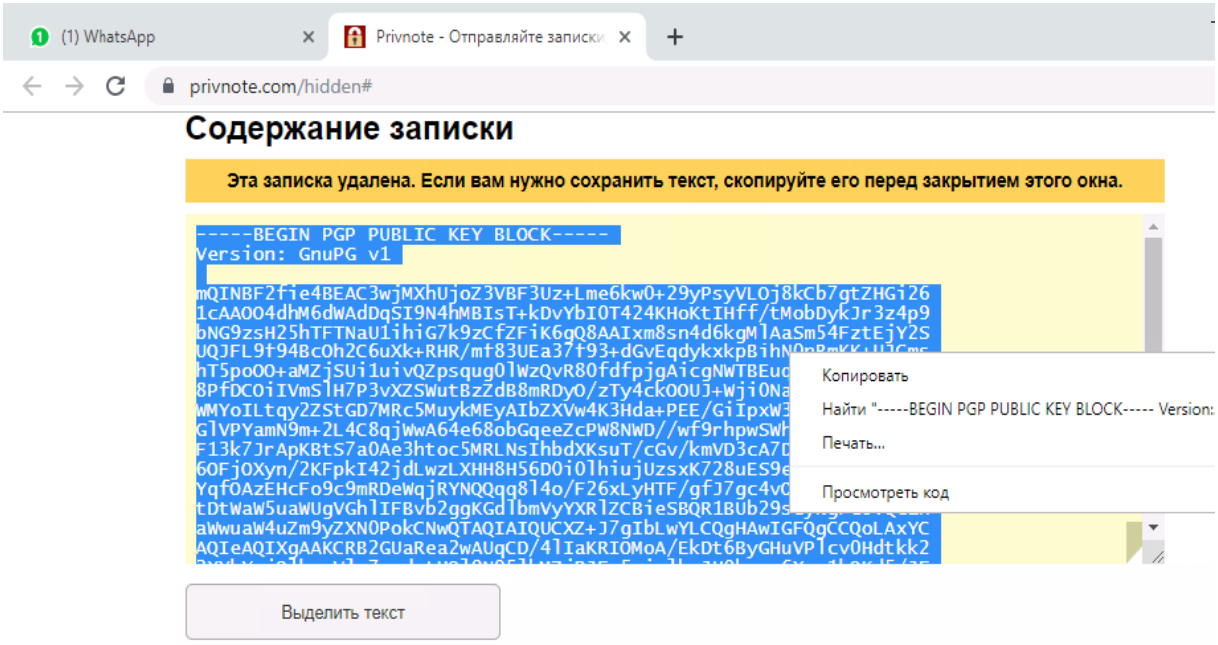


Рисунок П.36 – Скопируйте текст ключа

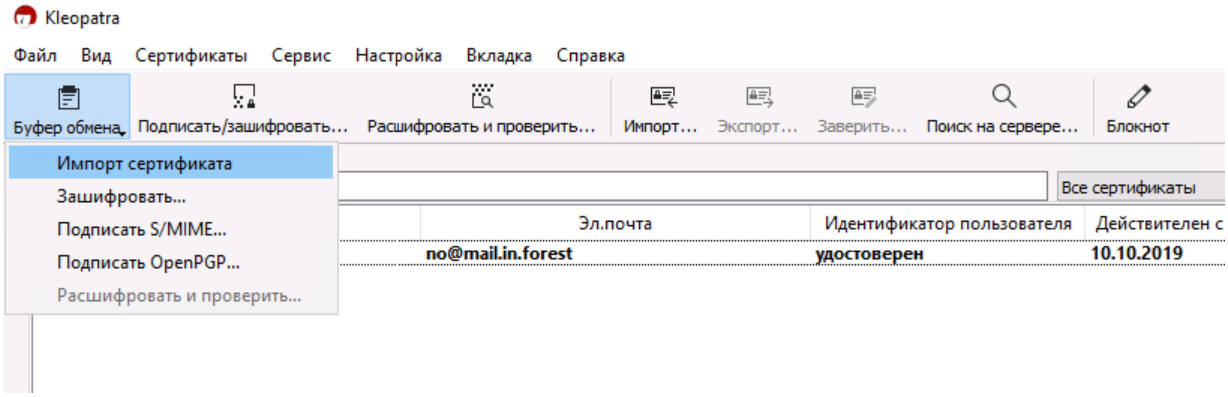


Рисунок П.37 – Откройте Клеопатру и нажмите Буфер обмена → Импорт Сертификата

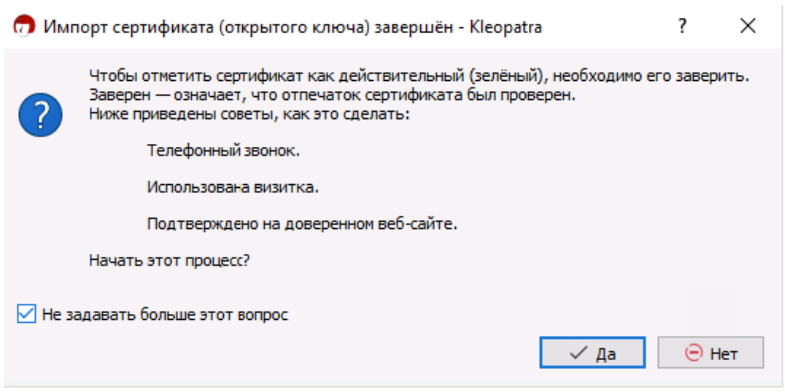


Рисунок П.38 – Нажмите “Да”

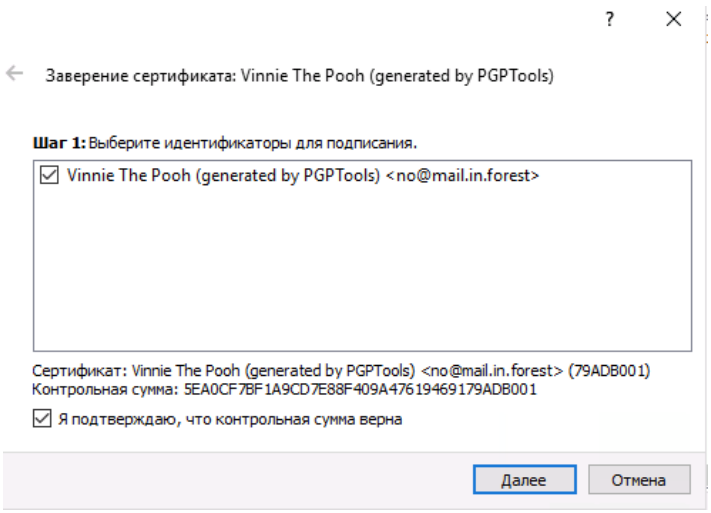


Рисунок П.39 – Поставьте две галки и нажмите Далее

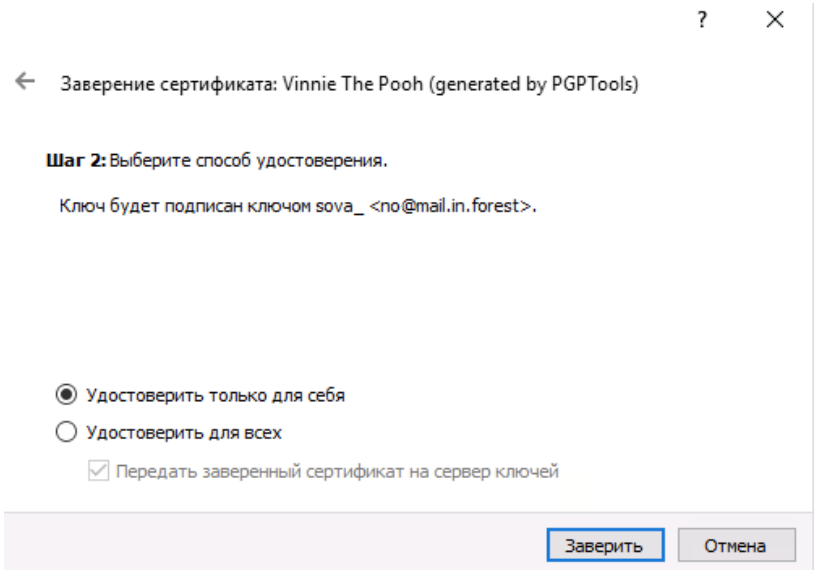


Рисунок П.40 – Нажмите “Заверить”

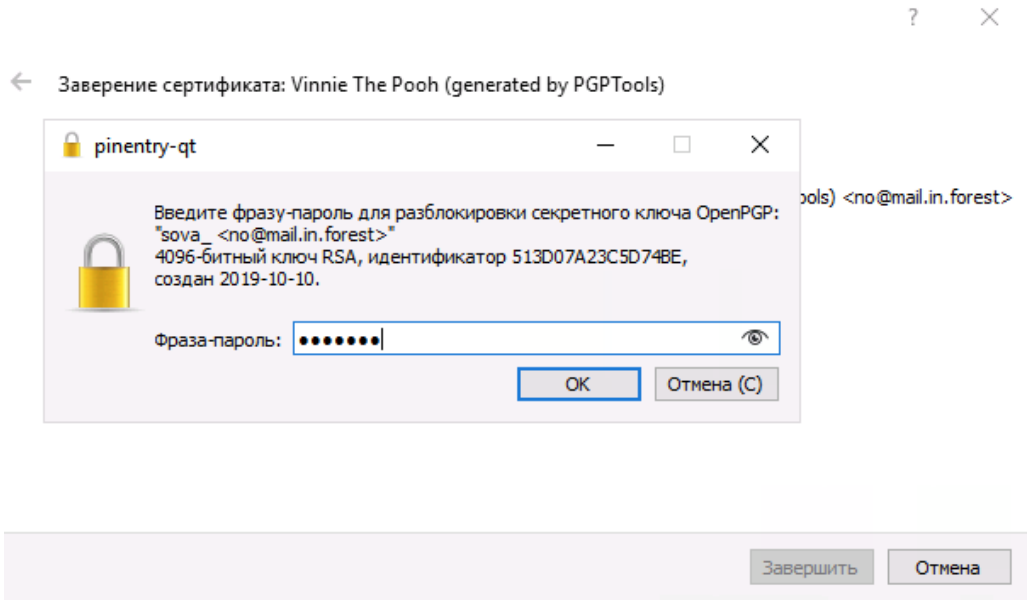


Рисунок П.41 – Сова вводит свой пароль чтобы заверить ключ Винни Пуха

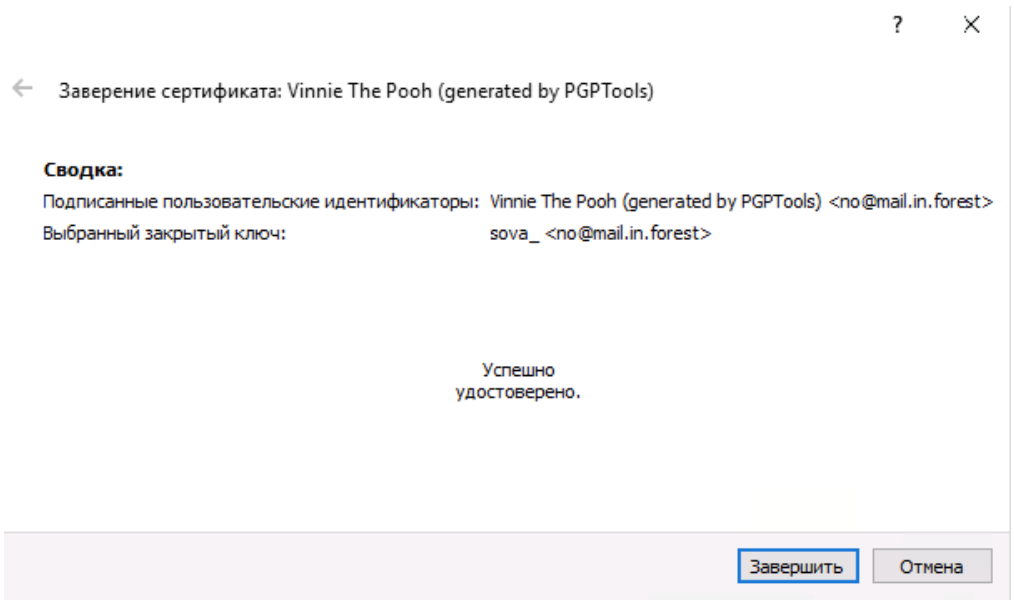


Рисунок П.45 – Нажмите “Завершить”

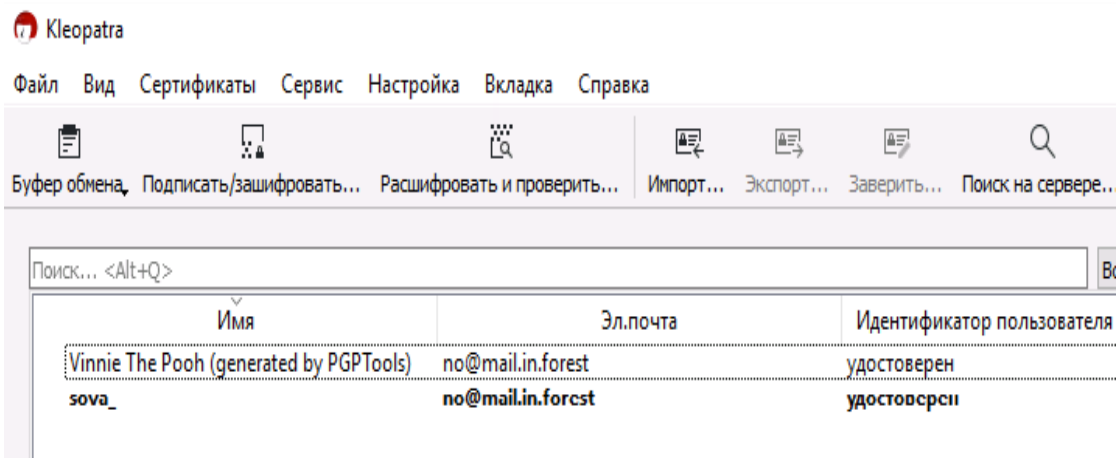


Рисунок П.46 – Теперь в списке Совы появился публичный ключ Винни

Обмен зашифрованными сообщениями

Самое сложное позади. Сова и Винни Пух обменялись ключами: это нужно сделать один раз и повторять только при смене ключей.

Теперь Сова должна послать Винни Пуху важную информацию для переговоров с Тигрой. Для этого достаточно открыть любой текстовый редактор и написать текст:

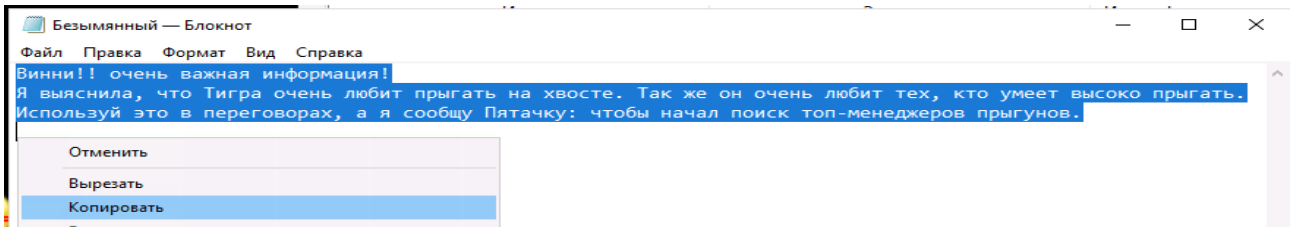


Рисунок П.47 – Скопируйте текст в буфер

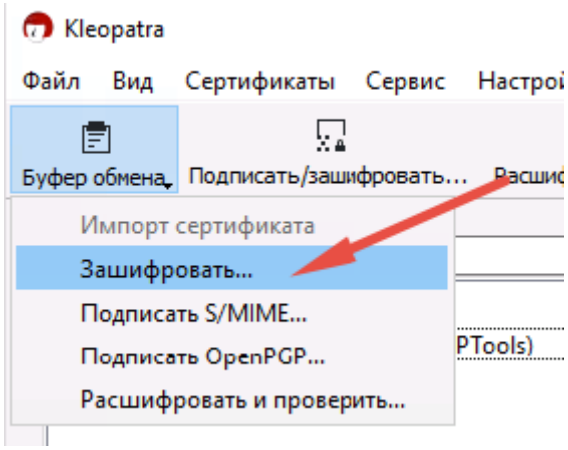


Рисунок П.48 – Откройте Клеопатру, Буфер обмена → Зашифровать

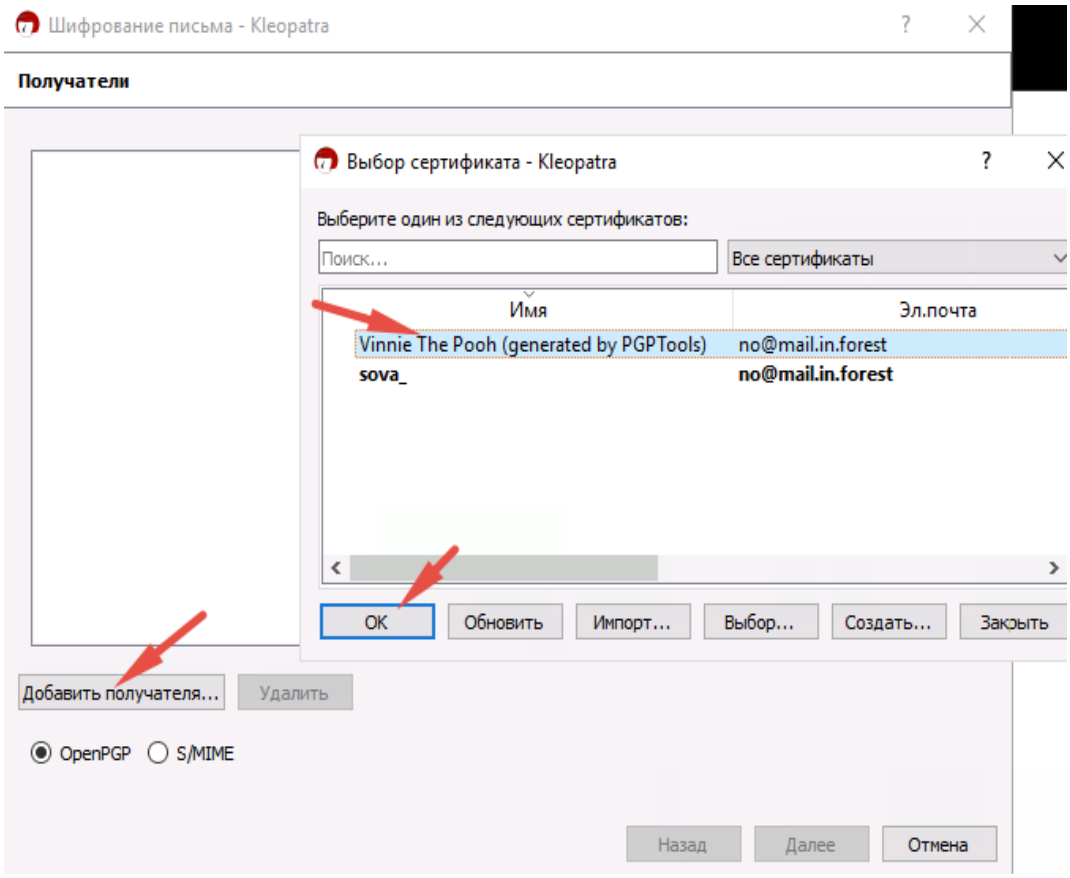


Рисунок П.49 – Выберите получателя

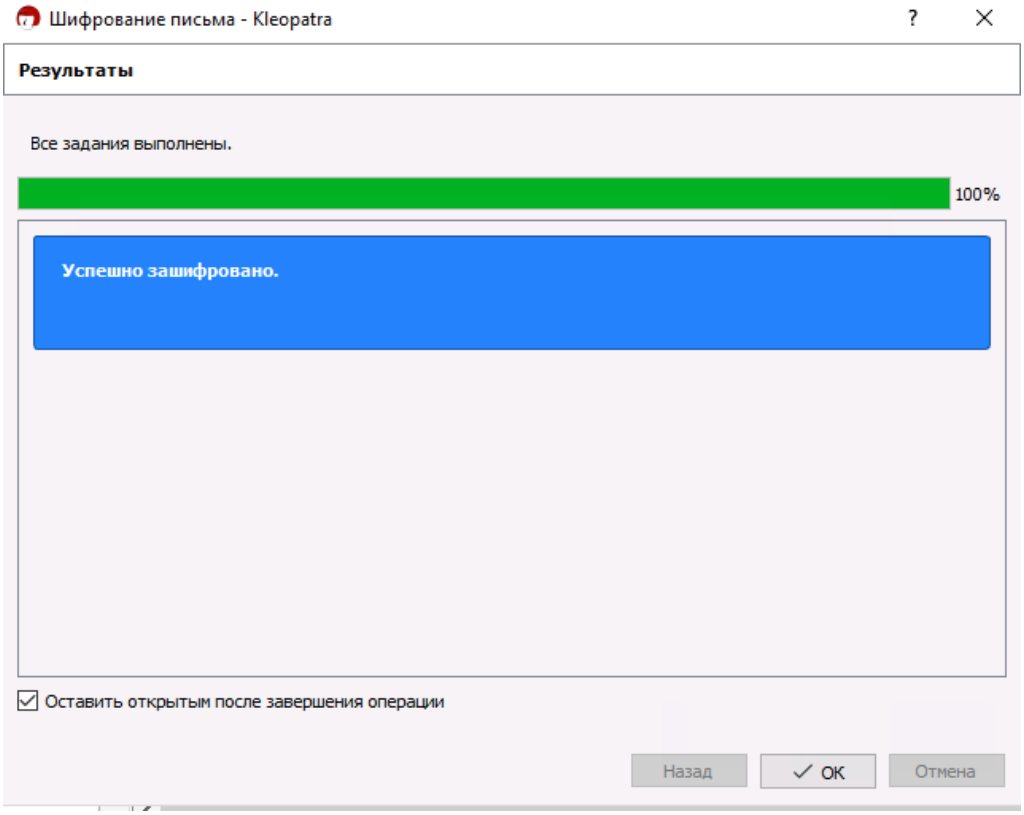


Рисунок П.50 – Сообщение из буфера зашифровано и уже скопировано в буфер. Остается только вставить его в записку

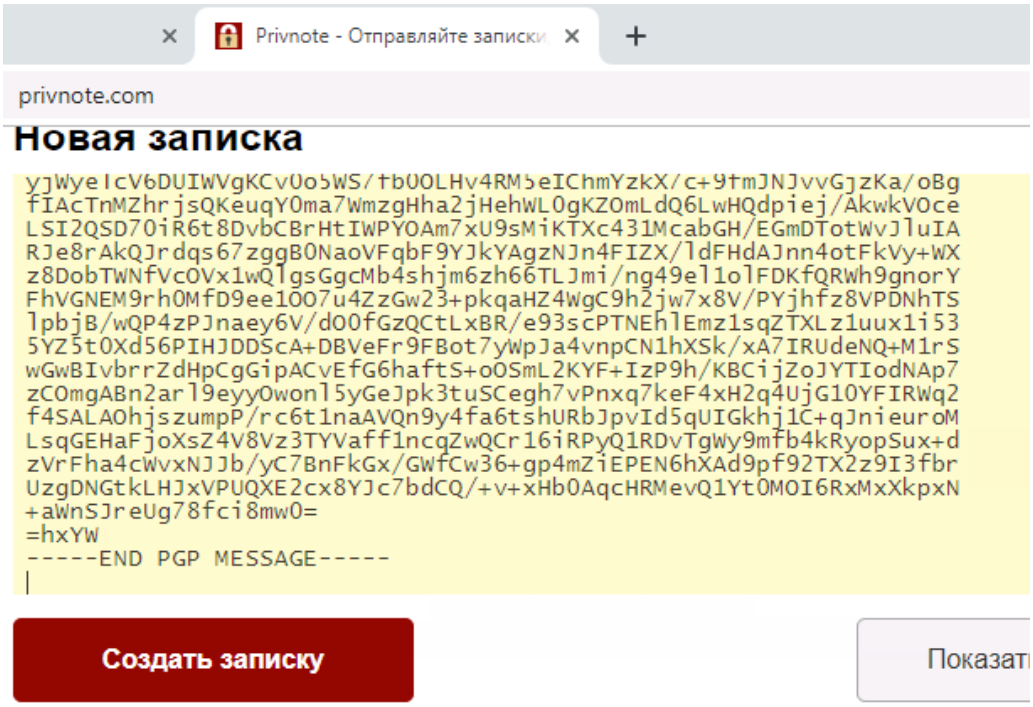


Рисунок П.51 – Вставьте зашифрованное сообщение в записку и отправьте ссылку Винни Пуху

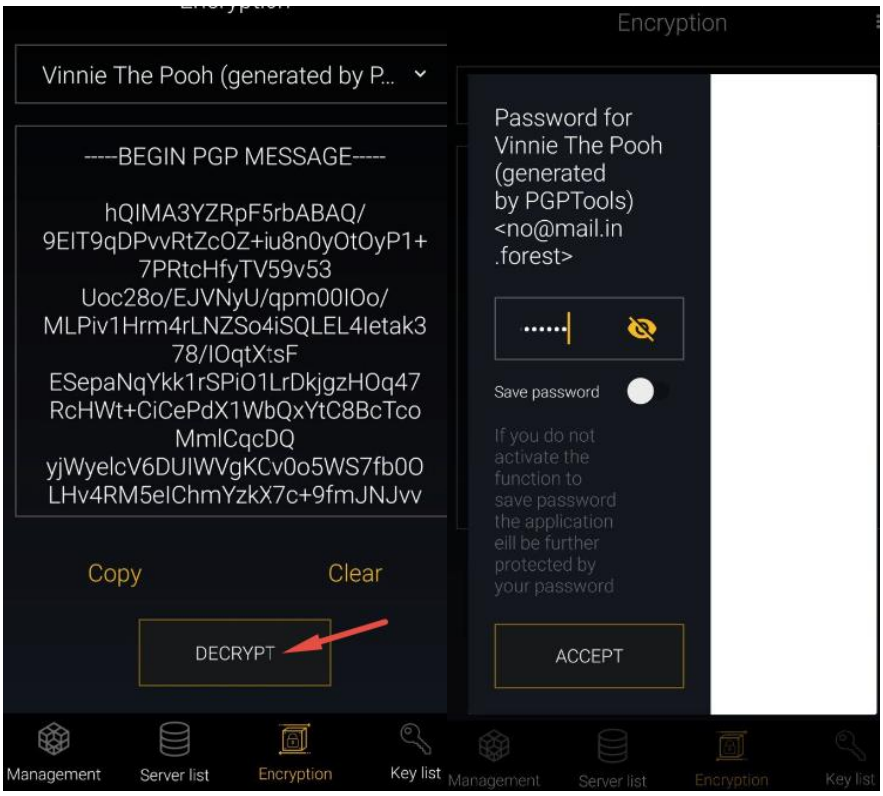


Рисунок П.52 – Винни вставляет зашифрованное сообщение кнопкой Paste и жмет Decrypt

Введите пароль ключа

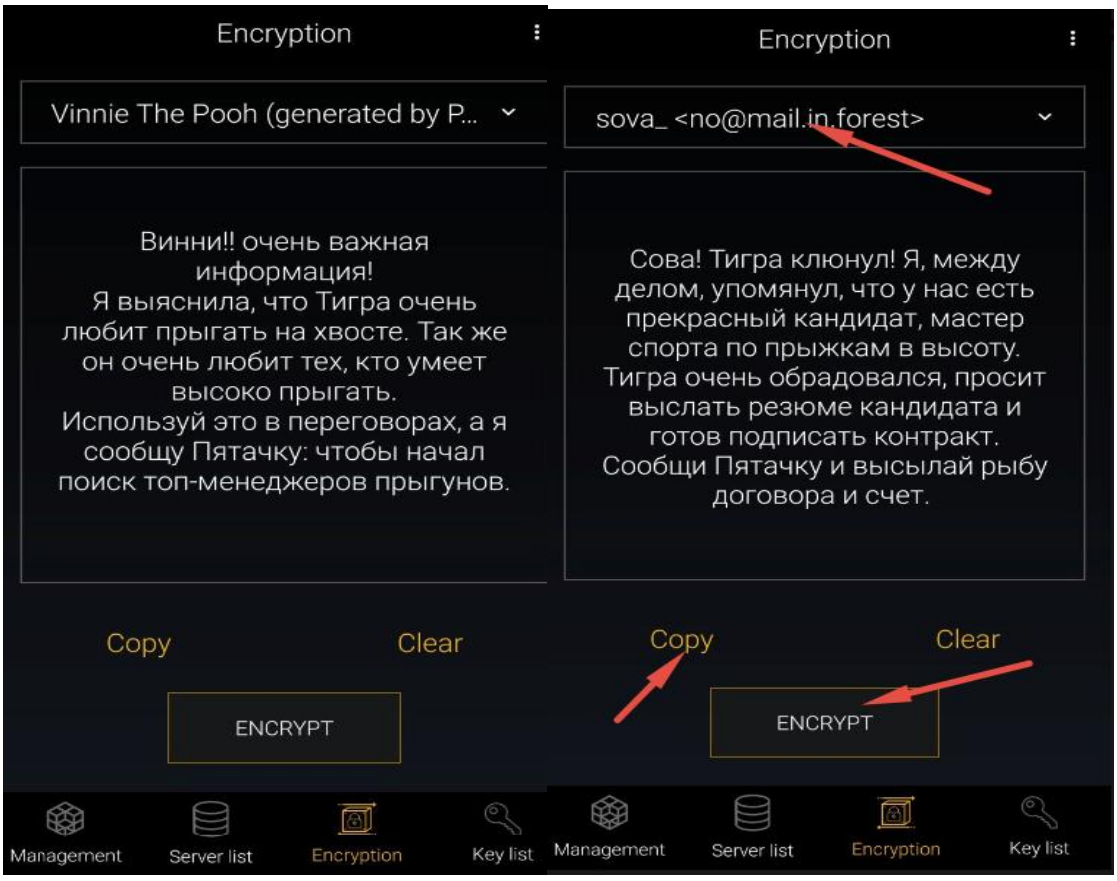


Рисунок П.53 – И прочтите сообщение

Теперь Винни пишет ответ Сове в этом же окне: просто очищает все кнопкой Clear, печатает текст, выбирает в адресатах Сову и жмет Encrypt.

Если в адресатах будет стоять кто-то другой, то сообщение зашифруется другим ключом и Сова не сможет его прочитать.

Винни копирует зашифрованное сообщение, вставляет его в privnote и отправляет ссылку Сове.

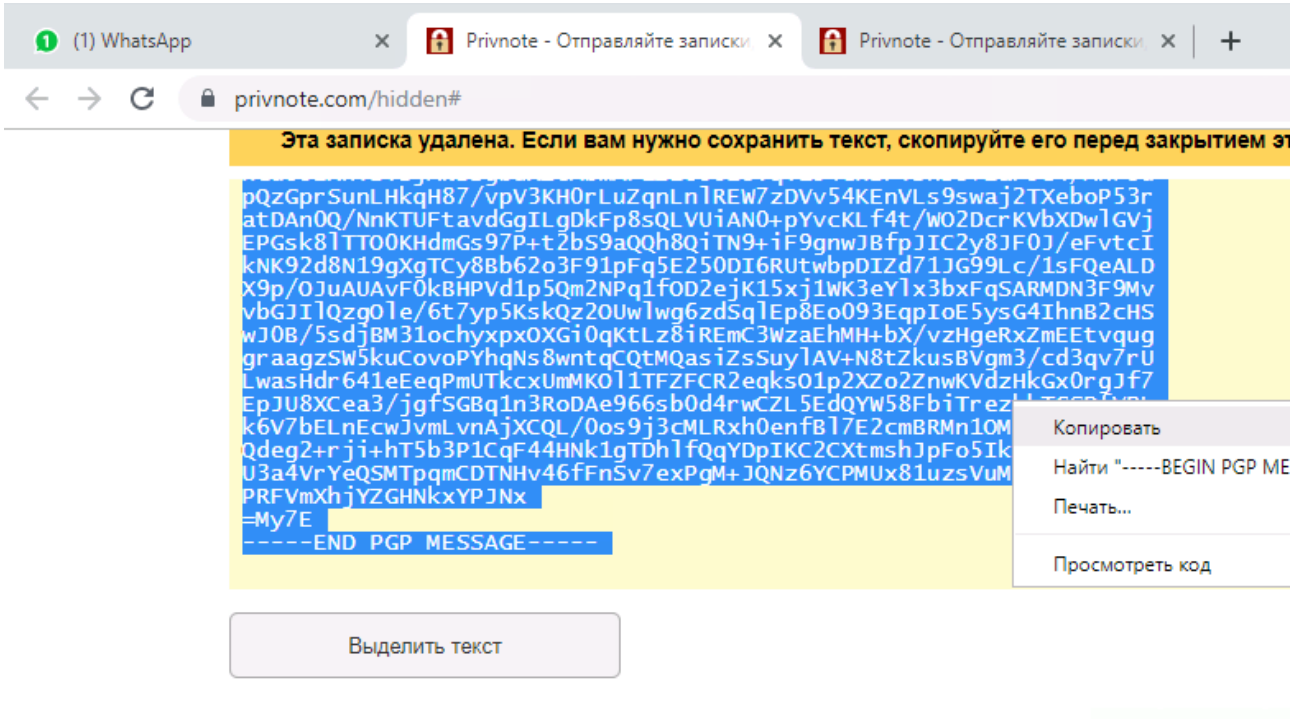


Рисунок П.54 – Сова открывает ссылку из WhatsApp и копирует текст в буфер

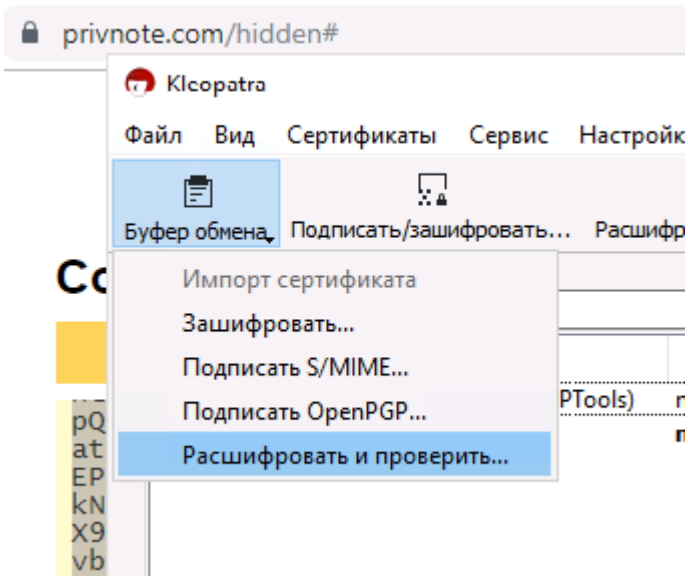


Рисунок П.55 – Открывает Клеопатру, Буфер обмена → Расшифровать, вводит пароль

Второй вариант без открытия окна Клеопатры: кликнуть правой кнопкой на иконку в трее и выбрать Расшифровать:

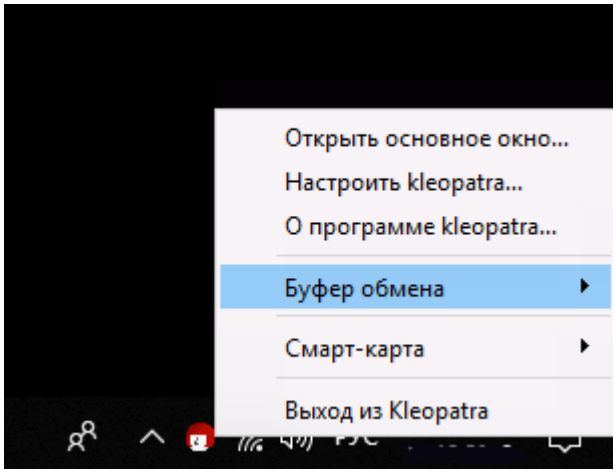


Рисунок П.56 – Результат будет тем же самым. Выбирайте наиболее удобный

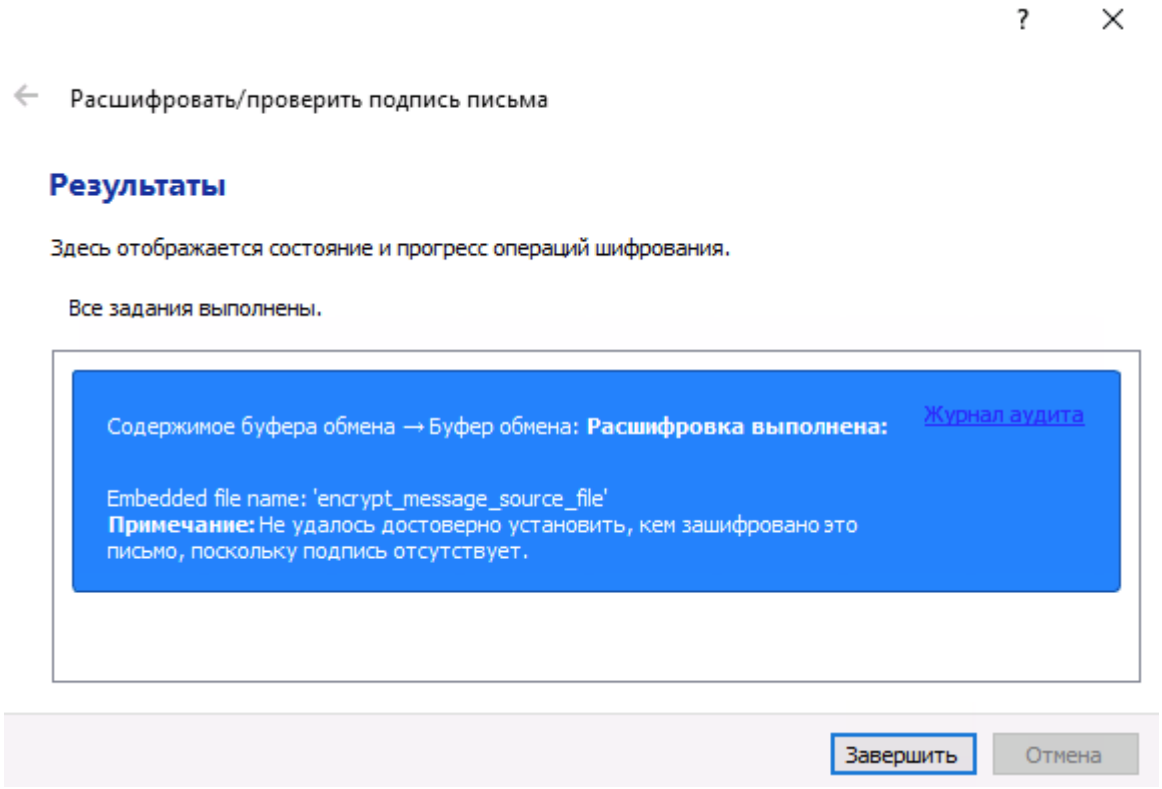


Рисунок П.57 – Сообщение успешно расшифровано и находится в буфере обмена. Просто вставьте текст в Блокнот

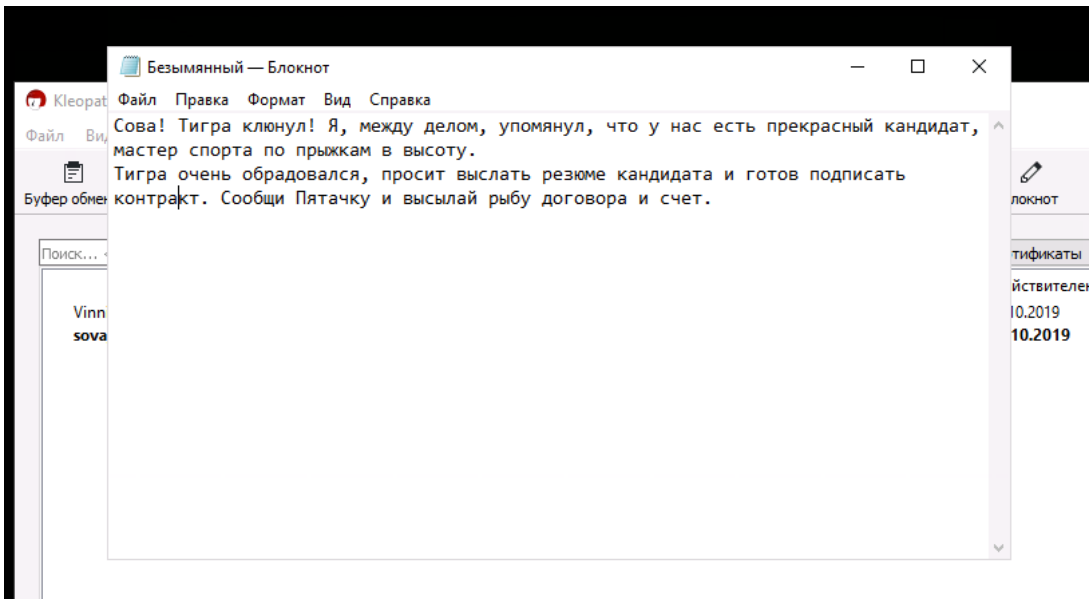


Рисунок П.58 – Результат

Получили зашифрованный тест.

- Сова и Винни Пух обменялись ключами и теперь могут за пару кликов обмениваться зашифрованными сообщениями.
- Винни Пух в мобильнике использует PGPTools, а Сова на ноутбуке – Kleopatra.
- Принцип один: оба пишут текст в редакторе, копируют его в буфер и шифруют с помощью своей программы.
- Зашифрованный результат они заливают на privnote.com и высылают ссылку адресату.
- При получении записки адресат копирует ее в буфер и расшифровывает с помощью своей программы.

Чтобы всем членам команды переехать на шифрованное общение, каждому придется установить PGPTools на смартфон и gpg4win на компьютер. Потом нужно будет создать пару ключей и выслать всем членам команды публичный. Важно помнить, что ключ должен быть продублирован на компе или телефоне, иначе пользоваться им можно будет только на одном устройстве.

Использование вместе с мессенджерами сторонних программ и сервиса одноразовых записок может показаться неудобным, но безопасность – антоним удобства. Если вы работаете с данными, которые ни при каких обстоятельствах не должны попасть в чужие руки, надеяться на шифрование мессенджера нельзя – придется настроить собственное. Данный отчет поможет сделать это без лишних сложностей, даже если вы не обладаете глубокими познаниями