КРИПТОКОМ

УТВЕРЖДЕН
СЕИУ.00009-01 99 - ЛУ

СРЕДСТВО КРИПТОГРАФИЧЕСКОЙ ЗАЩИТЫ ИНФОРМАЦИИ
МагПро КриптоПакет вер. 1.0

Протокол TLS и его практическая реализация
Общие сведения

СЕИУ.00009-01 99
Листов 19

Литера О

2009

Аннотация

Настоящий документ содержит общие сведения о протоколе TLS и его реализации в продукте МагПро КриптоПакет.

Содержание

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

1 ПОНЯТИЕ О ПРОТОКОЛЕ TLS

1.1 Протоколы защиты соединений

Существуют различные протоколы защиты соединений, реализующие шифрование на различных уровнях: на сетевом, на транспортном или на прикладном.

Сетевые протоколы — это защита не одного конкретного соединения, а целой сети. Подключение к защищенной сети автоматически предоставляет пользователю доступ ко всей информации, циркулирующей по этой сети. Такой подход удобен, например, для корпоративного взаимодействия (локальная сеть или ее распределенный аналог), но совершенно не подходит для взаимодействия, например, с клиентами. Примеры сетевого протокола - различные протоколы, на основании которых строятся сети VPN: IPSec, PPTP, OpenVPN.

Прикладные протоколы предназначены для решения конкретных специализированных задач и требуют использования соответствующего программного обеспечения, зачастую нестандартного и непривычного для пользователя. Пример прикладного протокола — протокол ssh.

Транспортные протоколы идеально подходят для защиты соединений, предоставляющих доступ к отдельным видам сервисов, предоставляемым сервером. Именно такие клиент-серверные соединения используются в огромном большинстве в интернет-приложениях, если для доступа на сервер требуется аутентификация пользователя, а также если передача информации между сервером и клиентом должна быть закрытой. В настоящее время наиболее распространенным протоколом транспортного уровня в интернет-приложениях является протокол TLS, который является развитием более ранней версии — протокола SSL.

1.2 Протокол TLS и протоколы интернет-соединений

Протокол TLS может использоваться для защиты соединений практически по всем распространенным интернет-протоколам. Например, всем известный протокол https — это интернет-соединение по протоколу http, защищенное по протоколу TLS. Кроме того, по протоколу TLS могут быть защищены соединения по протоколу FTP, по протоколам IMAP, POP3 и SMTP (передача электронной почты) и т.д.

2 ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ, РЕАЛИЗУЮЩЕЕ ПРОТОКОЛ TLS

2.1 TLS-серверы и TLS-клиенты

В качестве TLS-серверов и TLS-клиентов используются различные приложения. Например: сервер Apache и клиент Internet Explorer реализуют протокол https, сервер Dovecot и клиент Outlook реализуют протоколы POP3S и IMAPS, почтовый сервер Postfix является одновременно и клиентом и сервером протокола SMTPS, Outlook также может выступать в качестве клиента SMTPS при отправке сообщений на почтовый сервер.

2.2 Криптобиблиотеки, реализующие протокол TLS

Протокол TLS реализуют криптографические динамические библиотеки. В современных ОС Windows этот протокол может быть реализован с помощью Microsoft CryptoAPI. Кроме того, существует ряд OpenSource-библиотек, реализующих протокол TLS, среди них библиотеки OpenSSL и GNU TLS. Наиболее распространенной является криптобиблиотека OpenSSL, работающая в большинстве существующих операционных систем.

2.3 Продукты, реализующие различные криптографические алгоритмы в TLS

В ООО «Криптоком» разработаны программные продукты, предоставляющие возможность использовать в TLS-соединениях российские криптографические алгоритмы ГОСТ.

Реализация российских криптографических алгоритмов осуществляется с помощью динамических модулей, подключаемых к криптобиблиотекам. Одна и та же криптобиблиотека может работать с различными криптографическими алгоритмами благодаря тому, что к ней при работе могут подключаться различные модули, в каждом из которых реализован свой набор алгоритмов.

2.3.1 МагПро КриптоПакет

Алгоритмы ГОСТ для библиотеки OpenSSL реализованы в подключаемом модуле engine, входящем в состав продукта МагПро КриптоПакет. Криптобиблиотека OpenSSL версии 0.9.8 не предназначена для подключения модуля, содержащего алгоритмы ГОСТ, поэтому МагПро КриптоПакет также включает в себя модифицированную версию библиотеки OpenSSL. Эта версия предоставляет всю функциональность, которую предоставляет библиотека OpenSSL версии 0.9.8, и имеет возможность работать с алгоритмами ГОСТ.

Библиотека OpenSSL версии 0.9.9 может подключать подгружаемый модуль engine, реализующий алгоритмы ГОСТ, поэтому для реализации протокола TLS на алгоритмах ГОСТ с помощью OpenSSL 0.9.9 достаточно только такого модуля.

МагПро КриптоПакет в TLS-соединении может использоваться как на стороне клиента, так и на стороне сервера.

2.3.2 МагПро CSP

Для реализации алгоритмов ГОСТ в ОС Windows с помощью Microsoft CryptoAPI в ООО «Криптоком» разработан криптопровайдер МагПро CSP. Он также может быть использован при реализации протокола TLS.

МагПро CSP в TLS-соединении может использоваться только на стороне клиента.

3 ОПИСАНИЕ ПРОТОКОЛА TLS

3.1 Аутентификация сервера

При установлении соединения, защищенного по протоколу TLS, непременно производится аутентификация сервера. Это делается для того, чтобы клиент мог быть уверен, что соединение установлено именно с тем сервером, с которым планируется обмен информацией, а не с каким-либо другим компьютером, выдающим себя за сервер.

Аутентификация сервера всегда является криптографической, т.е. выполняется с помощью сертификата сервера. У сервера обязательно должна быть ключевая пара, и клиент должен иметь возможность проверки подлинности сертификата, который сервер предоставляет ему для аутентификации. (Подробно об аутентификации сервера см. раздел 4.)

3.2 Аутентификация клиента

Аутентификация клиента в протоколе TLS не является обязательной. Необходимость и способ аутентификации клиента определяется регламентом работы сервера.

Существует достаточно много случаев использования протокола TLS, когда достаточно уверенности клиента в том, что сервер, с которым он соединился, подлинный, и сессия будет защищена.

Аутентификация клиента может быть парольной и криптографической. При этом криптографическая аутентификация клиента является частью протокола TLS, а парольная аутентификация реализуется на прикладном уровне, т.е. средствами приложения.

При принятии решения о том, требовать ли с клиента криптографической аутентификации, следует учитывать, что эта аутентификация достаточно трудоемка и сложна для пользователя. Соответственно, требуется поддержка выдачи и отзыва клиентских сертификатов, пользователь должен соблюдать регламенты работы с долговременной ключевой информацией, поэтому криптографическая аутентификация клиентов применяется достаточно редко — только в таких критических приложениях, как, например, система WebMoney. И даже в таких приложениях, как правило, предусматривается альтернативный способ аутентификации, поскольку некоторые абонентские устройства, например мобильные телефоны, могут не поддерживать клиентские сертификаты.

Поэтому чаще всего используется аутентификация клиента по имени и паролю. В сочетании с криптографической аутентификацией сервера и последующей защитой канала это обеспечивает безопасность, достаточную для работы большинства приложений.

3.3 Защита данных

Цель защиты данных при TLS-соединении — сделать так, чтобы никто из тех, кто имеет возможность просматривать поток данных между клиентом и сервером, не мог извлечь из этого потока осмысленную информацию.

Защита данных при TLS-соединении реализуется с помощью шифрования всех передаваемых данных (в обе стороны). Для этого при каждом TLS-соединении вырабатываются специальные ключи для шифрования данных. Эти ключи называются сессионными (сессия — это однократное соединение от установления до разрыва соединения) и после завершения сессии уничтожаются без возможности восстановления.

Сразу после аутентификации сторон (или только сервера) происходит выработка и согласование ключей, которые будут использоваться для шифрования сессионных ключей. Для этого

используются ключи сервера и клиента. Если у клиента нет долгосрочной ключевой пары (для соединения с сервером не требуется криптографической аутентификации), программа-клиент вырабатывает эфемерную ключевую пару, которая используется так же, как использовалась бы долгосрочная ключевая пара в этом случае.

Согласование ключей происходит без их передачи по каналу связи, т.к. защита канала еще не установлена.

Следует иметь в виду, что возможно TLS-соединение без защиты данных, только с взаимной аутентификацией или даже только с аутентификацией сервера. Такое соединение имеет смысл при передаче несекретных данных, особенно в больших объемах. При TLS-соединении без защиты данных крайне нежелательно использовать парольную аутентификацию клиента, т.к. в этом случае пароль передается в незашифрованном виде и может быть подсмотрен. Поэтому аутентификация клиента, если она применяется, в таком соединении должна быть криптографической.

3.4 Шифрсьюты

При использовании протокола TLS есть возможность пользоваться различными криптографическими алгоритмами и использовать их различными способами. Эта возможность реализуется в виде различных наборов алгоритмов и особенностей протокола, которые можно использовать в TLS-соединениях. При каждом TLS-соединении используется один такой набор, причем он должен быть одинаковым у обеих сторон.

Такие наборы называются шифрсьютами (cipher suites).

У каждого программного продукта, реализующего протокол TLS, имеется некоторый определенный набор шифрсьютов, зависящий как от доступных алгоритмов, так и от собственно реализации протокола.

Такой набор шифрсьютов может быть расширяемым. То есть существует некоторый набор шифрсьютов, подключаемый при запуске приложения, использующего библиотеку OpenSSL, без дополнительных указаний. Этот набор включает не все доступные шифрсьюты. Если необходимы дополнительные шифрсьюты, они должны быть явно указаны в конфигурационном файле OpenSSL, и приложение должно вызывать их через обращение к этому файлу с помощью соответствующей функции (см. документ «Средство криптографической защиты информации МагПро КриптоПакет. Библиотека libssl. Руководство программиста (СЕИУ 00009-01 33 02)»).

В продукте МагПро КриптоПакет могут быть дополнительно подключены шифрсьюты без шифрования и без авторизации (анонимные шифрсьюты — используются в редчайших случаях).

Список доступных шифрсьютов можно получить командой openssl ciphers.

Продукт МагПро КриптоПакет содержит все шифрсьюты, включенные в библиотеку OpenSSL, а также шифрсьюты, включающие алгоритмы ГОСТ:

- Шифрсьюты, включаемые по умолчанию:
  GOST2001-GOST89-GOST89 и GOST94-GOST89-GOST89;
- Дополнительные шифрсьюты без шифрования:
  GOST94-NULL-GOST94 и GOST2001-NULL-GOST94

В данном случае в названиях шифрсьютов указываются соответственно по порядку: алгоритм выработки подписи и согласования ключей, алгоритм симметричного шифрования данных и алгоритм контроля целостности.

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

Ниже приводится расшифровка данных обозначений.

Таблица 1.
Расшифровка обозначений в шифрсьютах МагПро КриптоПакет
с использованием алгоритмов ГОСТ

Обозначение алгоритма	**В позиции алгоритма аутентификации**	**В позиции алгоритма шифрования**	**В позиции алгоритма контроля целостности**
GOST2001	ГОСТ Р 34.10-2001. Для обмена ключами в этих шифрсьютах используется алгоритм VKO GOST R 34.10-2001 (см. RFC 4357).		
GOST94	ГОСТ Р 34.10-94. Поскольку данный стандарт уже устарел, этими шифрсьютами не следует пользоваться.		HMAC на базе алгоритма хэширвания ГОСТ Р 34.11-94 (подробно об алгоритме HMAC см. в RFC 2104).
GOST89		ГОСТ 28147-89	ГОСТ 28147-89 в режиме имитозащиты.
NULL		Шифрование не производится	

(Подробно о том, какие шифрсьюты включены в OpenSSL и какие алгоритмы включены в какой шифрсьют, можно узнать из man-руководства к команде ciphers.)

3.5 Понятие о хендшейке

«Хендшейк» (рукопожатие) — это фаза (часть) сессии TLS-соединения, во время которой обе стороны «представляются друг другу».

Хендшейк обязательно включает в себя аутентификацию сервера, при необходимости — аутентификацию клиента, согласование и выработку сессионных ключей.

Хендшейк обязательно выполняется в начале сессии TLS-соединения.

При установлении TLS-соединения программа-клиент и программа-сервер в первую очередь обмениваются информацией о доступных шифрсьютах и «договариваются» о том, какой шифрсьют они будут использовать в данном соединении. Обычно выбор шифрсьюта производится по критерию наибольшей защищенности.

С того момента, как шифрсьют выбран, дальнейший хендшейк зависит от выбранного шифрсьюта. В частности, шифрсьютом определяется способ аутентификации сервера (и клиента).

Можно повторить хендшейк, заново провести аутентификацию и согласовать новые ключи, не прерывая сессии. Эта возможность используется, например, в тех случаях, если в момент начального хендшейка не проводится аутентификация клиента, а потом в ходе сессии становится понятным, что аутентификация клиента требуется, и какая именно аутентификация клиента требуется — парольная или криптографическая.

Поскольку фаза хендшейка достаточно трудоемка, для протокола TLS существует техническая возможность сохранения сессии и последующего ее возобновления. Это особенно актуально для протоколов, которые часто создают и закрывают TCP/IP соединения, например http.

В продукте МагПро КриптоПакет это не реализовано, потому что в нем используется дополнительная защита keymeshing (см. RFC 4357) — сессионные ключи меняются через каждый килобайт переданных данных. Поэтому сохранение сессии (т.е. взаимосогласованных ключей) технически крайне затруднено.

3.6 Строение сессии TLS-соединения

Таким образом, типичная сессия TLS-соединения устроена так:

- Хендшейк. Обмен шифрсьютами. Аутентификация сервера. Если необходимо — аутентификация клиента, выработка и согласование сессионных ключей для защиты данных.
- Обмен данными в обе стороны. Данные зашифровываются программой-отправителем на сессионных ключах, передаются, принимаются программой-получателем и расшифровываются. Выполняются проверки целостности данных.
  В ходе сессии могут выполняться также повтоные хендшейки.
- При разрыве соединения автоматически происходит закрытие сессии (возможно, с ее сохранением).

4 АУТЕНТИФИКАЦИЯ СЕРВЕРА

4.1 Получение серверных сертификатов

Для того, чтобы сервер мог быть аутентифицирован, ему необходимо иметь сертификат и соответствующий ему закрытый ключ.

Для получения сертификата администратору сервера необходимо:

1. с помощью команды openssl req создать заявку на сертификат и соответствующий закрытый ключ;
2. отправить созданную заявку в удостоверяющий центр.
   В удостоверяющем центре заявка подписывается на ключах данного удостоверяющего центра. Полученный сертификат сервера отправляется обратно администратору сервера.

4.2 Процедура аутентификации сервера

Процедура аутентификации сервера представляет собой всестороннюю проверку клиентом сертификата сервера, чтобы клиент мог быть уверен, что установил связь именно с тем сервером, с которым ему нужно установить связь, а не с каким-то злоумышленником, подменившим сервер.

Аутентификация сервера включает:

1. Проверку корректности сертификата. Это позволяет клиенту убедиться, что сертификат не искажен (раздел 4.3).
2. Проверку статуса сертификата. Это позволяет клиенту убедиться, что сертификат действителен (раздел 4.4).
3. Проверку принадлежности сертификата. Это позволяет клиенту убедиться, что предъявленный сертификат действительно принадлежит тому серверу, с которым клиент пытается установить соединение, т.е. не подменен (раздел 4.5).

4.3 Проверка корректности серверных сертификатов

Чтобы проверить корректность сертификата, необходимо иметь возможность проверить подпись под ним.

Для того, чтобы иметь возможность проверить подпись под серверным сертификатом, клиент должен иметь сертификат корневого удостоверяющего центра и доверять этому сертификату.

Подпись под сертификатом сервера проверяется с использованием сертификата удостоверяющего центра, выдавшего этот сертификат. Если этот удостоверяющий центр не является корневым, то необходимо проверять и подпись под сертификатом удостоверяющего центра. Т.е. для проверки сертификата сервера необходимо выстроить цепочку доверия от сертификата сервера до доверенного корневого сертификата.

Доверенный корневой сертификат — понятие, означающее, что человек, который пользуется данным сертификатом для проверки других сертификатов, уверен, что этот сертификат является подлинным и неискаженным, при том, что технической возможности проверить этот сертификат на другом сертификате нет, так как он является началом, «корнем» цепочек доверия.

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

Чтобы корневой сертификат удостоверяющего центра был доверенным, необходимо либо получить его по надежному каналу (по какому-либо закрытому каналу связи либо непосредственно в удостоверяющем центре), либо получить по аналогичному надежному каналу его контрольную сумму (т.наз. fingerprint) и при установке сертификата удостоверяющего центра сравнить продемонстрированную контрольную сумму с имеющейся.

4.4 Статус сертификата

Статус сертификата определяется двумя параметрами: сроками действия сертификата и фактом его отзыва/неотзыва.

4.4.1 Срок действия

У каждого сертификата имеется ограниченный срок действия, в сертификате указываются даты начала и конца срока действия. Если текущее время на машине, проверяющей сертификат, не попадает в период между этими датами, сертификат считается некорректным.

4.4.2 Списки отзыва сертификатов

Удостоверяющий центр может отозвать сертификат. Отзыв сертификата производится, например, если скомпрометированы ключи. Возможны и другие причины отзыва, например прекращение деятельности организации, которой выдан сертификат.

Для отзыва сертификатов удостоверяющий центр выпускает и распространяет по всем своим пользователям список отзыва сертификатов (certificate revocation list — CRL).

Процедура проверки корректности сертификата может включать или не включать проверку списка отзыва сертификатов. Как правило, решение об использовании списка отзыва сертификатов определяется регламентом удостоверяющего центра.

Библиотека OpenSSL предоставляет возможность проверки списка отзыва сертификатов, но по умолчанию такая проверка не проводится. Для того, чтобы такая проверка производилась, необходимо явным образом это указать.

Срок действия списка отзыва, как правило, заметно меньше, чем срок действия сертификата. Поэтому если списки отзыва используются, они должны регулярно обновляться.

Если в конфигурации указан устаревший список отзыва, то все сертификаты, выданные удостоверяющим центром, которому принадлежит данный список отзыва, будут считаться некорректными.

4.4.3 Онлайн-протокол проверки статуса сертификатов

Существует альтернативный способ проверки отзыва сертификатов — так называемый online-протокол проверки статуса сертификата (OCSP).

Для использования этого протокола необходимо наличие сервера, обычно принадлежащего удостоверяющему центру, который располагает актуальной информацией о статусе сертификатов, выпущенных этим удостоверяющим центром, и имеет специальный ключ, область применения которого — подпись OCSP-ответов.

В случае использования OCSP-сервера нет необходимости копировать списки отзыва на все машины, где может понадобиться проверка сертификатов. Статус сертификата запрашивается с сервера непосредственно в момент проверки.

К сожалению, не все программное обеспечение, использующее OpenSSL, поддерживает работу с OCSP.

4.5 Проверка принадлежности серверных сертификатов

Поскольку сертификаты серверов передаются открыто, то существует возможность, что злоумышленник, притворяющийся сервером, просто возьмет сертификат соответствующего сервера и отдаст его для аутентификации. Поэтому при аутентификации необходимо убедиться, что сертификат действительно принадлежит именно этому серверу.

Для этого следует убедиться, что у сервера есть не только сертификат, но и соответствующий ему закрытый ключ.

Эта проверка выполняется следующим образом:

1. Клиент высылает серверу некоторый набор данных, заранее неизвестный серверу.
2. Сервер подписывает этот набор данных на своем закрытом ключе и пересылает результат обратно клиенту.
3. Клиент, которому исходные данные известны, проверяет подпись под ними, пользуясь сертификатом сервера.

Таким образом, клиент убеждается, что сервер располагает не только данным сертификатом, но и соответствующим сертификату закрытым ключом.

4.6 Сертификат сервера и DNS

Любой сертификат на открытый ключ включает информацию о том, кто является его владельцем. В случае сертификатов серверов TLS это выполняется с помощью указания сетевого имени (DNS) сервера в поле CommonName.

Если сервер имеет несколько сетевых имен, то аутентификация считается успешной только в том случае, если предъявленный сервером сертификат соответствует именно тому сетевому имени, которое клиент использовал для установления соединения.

Необходимо иметь в виду, что если при аутентификации происходит ошибка (сетевое имя не совпадает с указанным в сертификате), большинство приложений дает возможность все-таки установить TLS-соединение. При этом выводится диалоговое окно, содержащее предупреждение следующего содержания:

Вы попытались установить соединение с [сетевое имя сервера]. *Однако представленный сертификат безопасности принадлежит* [сетевое имя, указанное в сертификате]. *Имеется вероятность, хотя и небольшая, что кто-то может попытаться перехватить ваше соединение с этим вебсайтом.*

Если вы подозреваете, что показанный сертификат не принадлежит [сетевое имя сервера], *пожалуйста, откажитесь от соединения и уведомите администратора сайта.*

Пользователю предоставляется возможность выбора из четырех вариантов действий:

– Просмотреть сертификат
– Ок (установить TLS-соединение)
– Отмена (отказаться от TLS-соединения)
– Помощь (справка)

В этом случае ответственность за безопасность устанавливаемого TLS-соединения полностью берет на себя пользователь.

Такая ситуация может возникнуть, например, если в сертификате указано сетевое имя вида `http://www.name.com`, а пользователь пытается установить соединение, используя сетевое имя вида `http://name.com`.

Следует также обратить внимание, что аутентификация сервера производится до того, как клиент передал какие-то данные прикладного протокола (например протокола http — скажем, на какую страницу он хочет попасть). Поэтому в момент аутентификации серверу ничего не известно о клиенте и о том, что клиент будет запрашивать у сервера. Известен только IP-адрес, на который пришел клиент (даже в том случае, если у сервера их несколько, к началу аутентификации уже известно, к какому IP-адресу обратился клиент).

Соответственно, если на одном и том же IP-адресе находятся несколько серверов с разными сетевыми именами (DNS), то необходимо использовать один из трех следующих вариантов:

- либо эти сервера должны использовать сертификат, соответствующий всем этим именам сразу (т.е. в CommonName указать *.domain);
- либо указать все сетевые имена этих серверов в поле сертификата SubjectAltName, которое позволяет указать несколько имен;
- либо использовать TLS только на одном из сетевых имен.

4.7 Типичные ошибки при работе с сертификатами

Многие операционные системы и криптографические программы уже включают в себя готовые сертификаты и либо предлагают их к установке, либо устанавливают по умолчанию. В данном разделе объясняется, к каким проблемам может привести использование таких сертификатов.

4.7.1 Самоподписанный сертификат сервера

Многие программы-сервера при установке предлагают установить самоподписанный сертификат сервера. Следует обязательно иметь в виду, что такой сертификат не предоставляет никакой гарантии безопасности соединения. При проверке отличить такой сертификат от любого другого сертификата с тем же доменным именем невозможно, поэтому его очень легко подменить. Самоподписанный сертификат сервера можно использовать только для проверки работоспособности установленного сервера.

Любой удостоверяющий центр, даже простейший (например на базе утилиты openssl), предоставляет гораздо более надежную гарантию защиты. Потому что:

1. Установка корневого сертификата удостоверяющего центра — отдельное, явное и контролируемое пользователем действие, которое предпринимается в начале работы с сервисом. Пользователь точно знает, что за сертификат он установил, выполняет проверку цифрового отпечатка (fingerprint), который также необходимо получить. После того, как корневой сертификат удостоверяющего центра установлен в системе и помещен в соответствующее хранилище доверенных сертификатов, пользователь получает возможность надежной аутентификации сервера по сертификату сервера, подписанному на корневом сертификате удостоверяющего центра.

2. Для того, чтобы при правильных действиях пользователя получить возможность успешной атаки на TLS-соединение, злоумышленнику необходимо перехватить не только собственно TLS-сессию, но и канал, по которому пользователь получает сертификат удостоверяющего центра и его цифровой отпечаток (fingerprint). Даже если пользователь получает все эти данные по незащищенному http-соединению, задача атакующего многократно усложняется.

4.7.2 Предустановленные сертификаты

В состав дистрибутивов операционных систем и криптографических программ включаются корневые сертификаты некоторых удостоверяющих центров. Это делается для того, чтобы криптографические программы с момента установки считали эти удостоверяющие центры доверенными и могли работать с сертификатами, выданными этими удостоверяющими центрами. Так, в состав ОС Windows входят корневые сертификаты известного удостоверяющего центра VeriSign. Среди таких предустановленных сертификатов в настоящее время нет ни одного сертификата с алгоритмами ГОСТ.

Пользователям обязательно следует учитывать, что данные корневые сертификаты часто являются доверенными исключительно с технической точки зрения (т.к. они установлены в соответствующем хранилище). На самом деле считать их доверенными нельзя, потому что пользователь при установке этих сертификатов не имел возможность проверить их корректность путем сверки цифрового отпечатка.

Часто неопытные пользователи считают, что клиентские и серверные сертификаты необходимо получать именно в тех удостоверяющих центрах, чьи корневые сертификаты установлены в системе по умолчанию. Это на самом деле не так. Можно использовать любой удостоверяющий центр, если все участники криптопротокола (сервер и клиенты) установят его корневой сертификат как доверенный.

5 ХРАНЕНИЕ СЕРТИФИКАТОВ

5.1 Способы хранения сертификатов удостоверяющих центров

OpenSSL поддерживает два основных способа хранения сертификатов удостоверяющих центров:

1. Текстовый файл, в котором содержатся все необходимые сертификаты и списки отзывов в PEM-формате.
   В утилите openssl такой файл указывается с помощью опции -CAfile.
   Недостаток этого способа — содержимое файла полностью загружается в память. Поэтому при использовании большого количества УЦ или при использовании УЦ с объемными CRL лучше использовать второй способ.
2. Каталог, в котором содержатся сертификаты и CRL в виде отдельных файлов.
   Для удобства и быстроты работы с сертификатами в этом каталоге следует использовать утилиту `c_rehash`, которая создает ссылки на файлы сертификатов, содержащихся в каталоге-хранилище, с именами в виде шестнадцатеричных тридцатидвухбитных хэш-сумм наименований удостоверяющих центров. Благодаря наличию таких хэшированных имен библиотека OpenSSL может мгновенно найти нужный сертификат или CRL.
   В этом случае размеры и количество файлов неограничены, так как в процессе работы читается и загружается только нужная информация.
   Поскольку утилита `c_rehash` работает только с файлами с расширением .pem, следует помещать файлы сертификатов и CRL в каталог-хранилище именно с таким расширением.
   Опция утилиты openssl для работы с каталогом-хранилищем — -CApath.

В принципе, библиотека OpenSSL позволяет реализовать любые другие способы хранения сертификатов, например в базе данных или LDAP-каталоге. Но с подобными реализациями должно работать непосредственно приложение (см. соответствующий раздел руководства программиста по libcrypto).

5.2 Хранение собственного сертификата сервера и его закрытого ключа

Как правило, собственный сертификат сервера и его закрытый ключ хранятся непосредственно на сервере либо в отдельных файлах, либо в одном текстовом файле в PEM-формате.

Защита собственного сертификата сертификата сервера и его закрытого ключа должна осуществляться с помощью прав доступа файловой системы.

Закрытый ключ также может быть дополнительно защищен паролем (пассфразой). Но в таком случае при каждой перезагрузке сервера кто-то должен вводить этот пароль, что не всегда возможно (в частности, если перезагрузка выполняется дистанционно).

Поэтому для серверных приложений обычно используют закрытые ключи, защищенные только правами доступа файловой системы. Для интерактивных приложений лучше использовать ключи, защищенные на пароле.

Некоторые приложения, например OpenVPN, позволяют хранить собственные сертификаты и ключ в формате PKCS#12.

5.3 Хранение сертификатов промежуточных удостоверяющих центров

У стороны, проверяющей сертификат, как правило, есть только доверенный сертификат корневого удостоверяющего центра. Если сертификат другой стороны подписан не им, а каким-то промежуточным удостоверяющим центром, то эта сторона должна предоставить все сертификаты, позволяющие выстроить цепочку от ее собственного сертификата до доверенного сертификата корневого удостоверяющего центра. Как правило, для хранения такой цепочки используется файл в формате PEM, содержащий все необходимые сертификаты в таком порядке, в каком они будут проверяться.

Приложение. ФОРМАТЫ ХРАНЕНИЯ КРИПТОГРАФИЧЕСКОЙ ИНФОРМАЦИИ

.1 Форматы кодирования криптографических документов DER и PEM

OpenSSL поддерживает два основных формата, определяющих кодирование криптографических документов: DER и PEM.

DER — бинарные файлы.

PEM — текстовые файлы.

Криптографический документ в формате PEM имеет вид

```
-----BEGIN <ТИП ДОКУМЕНТА>-----
<Документ в формате DER, закодированный в base64>
-----END <ТИП ДОКУМЕНТА>-----
```

Т.е. содержание документа окружается заголовками специального вида, состоящими из пяти знаков «-», слов begin и end и указания типа документа (CERTIFICATE, CRL и т.д.)

Благодаря наличию таких заголовков в одном файле может содержаться несколько документов в формате PEM: например, цепочка сертификатов или сертификат и его закрытый ключ.

По умолчанию в OpenSSL используется именно формат PEM.

.2 Форматы, определяющие структуру файлов, содержащих криптографическую информацию

Файлы, содержащие цепочки сертификатов, сертификаты и ассоциированные с ними ключи, могут представляться в нескольких форматах, определяющих состав и строение файлов, содержащих ключевую информацию; сама эта информация кодируется в вышеописанных форматах PEM и DER.

Наиболее распространенные из форматов, определяющих структуру файлов, содержащих криптографическую информацию — PKCS#7 и PKCS#12.

.2.1 Формат PKCS#7

PKCS#7 — формат защищенных сообщений, который, кроме самого сообщения, может содержать необходимые для работы с ним сертификаты и CRL.

Многие удостоверяющие центры распространяют свои сертификаты и CRL в виде PKCS#7-документов, в которых нет сообщения, есть только служебная информация.

PKCS#7-документ может быть закодирован и формате DER, и в формате PEM.

Расширения файлов сертификатов могут быть .cer или .crt . Расширение служебных PKCS#7-сообщений обычно бывает .p7b или .p7f. Эти расширения никак не коррелируют с форматами DER и PEM, т.е. файл с любым из этих расширений может быть закодирован как в формате DER, так и в формате PEM.

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

.2.2 Формат PKCS#12

PKCS#12 — формат для переноса сертификата и связанного с ним закрытого ключа с машины на машину или для резервного копирования. В этом формате могут также содержаться сертификаты удостоверяющего центра.

Файлы формата PKCS#12 всегда кодируются в формате DER. Эти файлы требуют особенно пристального внимания, поскольку содержат закрытый ключ; при неосторожном обращении с таким файлом закрытый ключ легко скомпрометировать. Как правило, эти файлы защищены на пароле.

Расширение файлов формата PKCS#12 либо .p12, либо .pfx .

.3 Преобразование форматов

OpenSSL предоставляет возможность преобразования файлов в формате DER в формат PEM. Для этого используются команды:

Для сертификатов:

```
openssl x509 -in filename.der -inform der -out filename.pem
```

Для списков отзывов:

```
openssl crl -in filename.der -inform der -out filename.pem
```

Для сообщений:

```
openssl pkcs7 -in filename.der -inform der -out filename.pem
```

Для извлечения сертификатов из сообщений формата PKCS#7:

```
openssl pkcs7 -in filename.pem -print_certs
```

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

Лист регистрации изменений									
Номера листов (страниц)					Всего листов (страниц) в докум.	№ документа	Входящий сопроводительного докум. и дата	Подпись	Дата
Изм.	измененных	замененных	новых	аннулированных					

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения