НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
май–июнь 2016 Том 16 № 3 ISSN 2226-1494 http://ntv.ifmo.ru/
SCIENTIFIC AND TECHNICAL JOURNAL OF INFORMATION TECHNOLOGIES, MECHANICS AND OPTICS
May–June 2016 Vol. 16 No 3 ISSN 2226-1494 http://ntv.ifmo.ru/en

УДК 004.056

АЛГОРИТМ АУТЕНТИФИКАЦИИ УЧАСТНИКОВ ИНФОРМАЦИОННОГО ВЗАИМОДЕЙСТВИЯ ПРИ УДАЛЕННОЙ ЗАГРУЗКЕ ОПЕРАЦИОННОЙ СИСТЕМЫ НА ТОНКИЙ КЛИЕНТ

Ю.А. Гатчин[a], О.А. Теплоухова[a]

[a] Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация
Адрес для переписки: teplouhovaoa@gmail.com

Информация о статье
Поступила в редакцию 29.12.15, принята к печати 13.04.16
doi: 10.17586/2226-1494-2016-16-3-497-505
Язык статьи – русский

Ссылка для цитирования: Гатчин Ю.А., Теплоухова О.А. Алгоритм аутентификации участников информационного взаимодействия при удаленной загрузке операционной системы на тонкий клиент // Научно-технический вестник информационных технологий, механики и оптики. 2016. Т. 16. № 3. С. 497–505. doi: 10.17586/2226-1494-2016-16-3-497-505

Аннотация

Предмет исследования. Рассмотрена проблема проверки подлинности всех компонентов информационного взаимодействия, задействованных в процессе загрузки операционной системы на тонкий клиент по сети с терминального сервера. **Описание системы.** Поскольку контроль целостности загружаемой операционной системы осуществляется аппаратно-программным модулем, в состав которого входит USB-токен для безопасного хранения криптографических ключей и загрузчика, то основной задачей является обеспечение взаимной аутентификации четырех сторон: терминального сервера, тонкого клиента, токена и пользователя. Для решения поставленной задачи было разработано два алгоритма. Первый основывается на сравнении результатов шифрования одноразового пароля с эталонным значением, хранящимся в памяти токена и обновляющимся при успешной аутентификации. Второй алгоритм предусматривает использование пар открытых и закрытых ключей токена и сервера, позволяющих путем криптографических преобразований аутентифицировать и сформировать защищенное информационное взаимодействие токена, тонкого клиента и терминального сервера. **Основные результаты.** В процессе апробации разработанные алгоритмы исследованы на выполнение предъявляемых к ним требований. Выполнено сравнение применяемых подходов по применимости в многопользовательской архитектуре системы терминального доступа, по возникающим угрозам и реализуемой степени защищенности. По итогам сравнительного анализа рекомендован к применению алгоритм на основе инфраструктуры открытых ключей в силу его хорошей масштабируемости, простоты использования. Подтвержден высокий уровень безопасности, основанный на том, что реализация методов асимметричной криптографии позволяет никогда не передавать закрытые ключи участников в процессе аутентификации. **Практическая значимость.** Предложенный алгоритм на основе инфраструктуры открытых ключей позволяет решить поставленную задачу с использованием криптографических алгоритмов в соответствии с государственным стандартом даже при отсутствии отечественного стандарта на асимметричную криптографию. Таким образом, предлагаемый алгоритм применим в государственных информационных системах с повышенными требованиями к защите информации.

Ключевые слова
тонкий клиент, операционная система, аутентификация, инфраструктура открытых ключей, модуль доверенной загрузки, электронная подпись, имитовставка

Благодарности
Работа является победителем программы «Участие молодежного научно-инновационного конкурса» («УМНИК»). Отмечена дипломом «За лучший доклад» на IV Всероссийском конгрессе молодых ученых (2015 г.).

AUTHENTICATION ALGORITHM FOR PARTICIPANTS OF INFORMATION INTEROPERABILITY IN PROCESS OF OPERATING SYSTEM REMOTE LOADING ON THIN CLIENT

Yu.A. Gatchin[a], O.A. Teploukhova[a]

[a] ITMO University, Saint Petersburg, 197101, Russian Federation
Corresponding author: teplouhovaoa@gmail.com

Article info
Received 29.12.15, accepted 13.04.16
doi: 10.17586/2226-1494-2016-16-3-497-505
Article in Russian

For citation: Gatchin Yu.A., Teploukhova O.A. Authentication algorithm for participants of information interoperability in process of operating system remote loading on thin client. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2016, vol. 16, no. 3, pp. 497–505. doi: 10.17586/2226-1494-2016-16-3-497-505

Abstract

Subject of Research. This paper presents solution of authentication problem for all components of information interoperability in process of operation system network loading on thin client from terminal server. **System Definition.** In the proposed solution operation system integrity check is made by hardware-software module, including USB-token with protected memory for secure storage of cryptographic keys and loader. The key requirement for the solution is mutual authentication of four participants: terminal server, thin client, token and user. We have created two algorithms for the problem solution. The first of the designed algorithms compares the encrypted one-time password (random number) with the reference value stored in the memory of the token and updates this number in case of successful authentication. The second algorithm uses the public and private keys of the token and the server. As a result of cryptographic transformation, participants are authenticated and the secure channel is formed between the token, thin client and terminal server. **Main Results.** Additional research was carried out to find out if the designed algorithms meet the necessary requirements. Criteria used included applicability in a multi-access terminal system architecture, potential threats evaluation and overall system security. According to analysis results, it is recommended to use the algorithm based on PKI due to its high scalability and usability. High level of data security is proved as a result of asymmetric cryptography application with the guarantee that participants' private keys are never sent in the authentication process. **Practical Relevance.** The designed PKI-based algorithm allows solving the problem with the use of cryptographic algorithms according to state standard even in its absence on asymmetric cryptography. Thus, it can be applied in the State Information Systems with increased requirements to information security.

Keywords

thin client, operating system, authentication, public key infrastructure, trusted loading module, digital signature, message authentication code

Acknowledgements

The work is a winner of the program "Participation of Youth Research and Innovation Competition" ("UMNIK"). It has been given a diploma "For the Best Report" at the IV All-Russian Congress of Young Scientists (2015).

Введение

Существующие в настоящее время сетевые вычислительные платформы по различным причинам (экономическая нецелесообразность, ограничение возможности выполнения первичных решаемых задач и др.) не учитывают в полном объеме многочисленные требования в отношении безопасности, предъявляемые со стороны возможных участников информационного взаимодействия: владельцев инфраструктуры, конечных пользователей, контент-провайдеров и т.д. Уверенность в программных решениях, обеспечивающих информационную безопасность, напрямую зависит от корректности процессов их собственной установки и выполнения [1]. Вычислительная среда, в рамках которой работает критичное программное обеспечение (ПО), должна представлять собой некую доверительную платформу, содержащую аппаратный компонент, обеспечивающий доверие между всеми участниками вычислительного процесса. Такой аппаратный компонент также должен обеспечивать защиту криптографических ключей и других данных, которые могут быть использованы для алгоритмов шифрования или управления доступом к серверам и сетевому оборудованию. Таким образом, один из ключевых механизмов безопасности, который необходимо принимать во внимание при построении доверенной вычислительной среды на базе аппаратных модулей, это аутентификация всех участников информационного обмена.

Описание поставленной задачи

В силу наблюдаемого за последние годы смещения выбора аппаратной вычислительной платформы в сторону концепции «тонкий клиент – сервер» [2], вопрос обеспечения доверенной вычислительной среды для клиентских рабочих мест также не теряет своей актуальности. Согласно исследованиям Intel [3], выделяется пять основных преимуществ безопасности в использовании тонких клиентов: предотвращение физической потери данных, лишение полномочий привилегированного пользователя, ограничения на установку недоверенного ПО, целостность клиентского ПО и возможность в кратчайший срок «откатиться» к корректному рабочему состоянию. Для реализации этих функций необходима доверенная вычислительная платформа, в рамках которой будут выполняться процессы и клиентской операционной системы (ОС), и специализированного ПО. Важными свойствами такой платформы на тонком клиенте являются аутентичность поступающих данных, целостность кода ОС и применяемого ПО (в том числе средств защиты информации), а также наличие интерфейса взаимодействия с недоверенной средой. Для обеспечения этих свойств необходимо наличие реализующего их комплекса механизмов информационной безопасности.

Одной из самых распространенных проблем систем терминального доступа (СТД) [4] является непроработанный механизм аутентификации, в том числе слабая парольная политика и недостаточная защита от подбора учетных данных (в том числе методом «грубой силы», атакой по словарю и т.д.). Часто встречающийся недостаток механизма идентификации пользователей – предсказуемый формат идентификаторов или раскрытие информации о существующих в системе идентификаторах. Например, в каче-

стве имени учетной записи пользователя домена используются различные комбинации его собственных фамилии и имени. И хотя в данном случае принцип Керкгоффса не нарушен (в секрете должен держаться только пароль), такое использование идентификаторов в сочетании с различными находящимися в широком доступе средствами автоматизации значительно повышает вероятность успеха атак на учетные записи пользователей с получением доступа в систему. Такое заключение дает повод задуматься о несложном механизме формирования идентификаторов, соответствующих требованиям безопасности.

Для защищенной обработки особо критичной информации уже давно применяются отдельные специализированные устройства, самостоятельно реализующие криптографические алгоритмы – токены и смарт-карты. Обладание таким устройством уже является одним из факторов, подтверждающих подлинность пользователя. В этом случае вместо предсказуемого имени пользователя используется идентификатор токена. При этом разграничение доступа к зашифрованным контейнерам, хранящимся в памяти таких носителей, осуществляется, как правило, с помощью PIN-кода, что является вторым фактором аутентификации. Защита от модификации реализуемых такими устройствами алгоритмов обеспечивается на аппаратном уровне производителями и подтверждается соответствующей сертификацией.

Для реализации контроля аутентичности данных, поступающих на тонкий клиент, необходимо ограничить круг источников (например, доверенным сервером). Целостность загружаемого образа ОС и разграничение доступа к процессу загрузки должны быть реализованы в механизме доверенной сетевой загрузки, включающем реализацию защищенного протокола загрузки.

В работе [5] авторами предлагается решение, позволяющее осуществлять защищенную загрузку ОС по сети с терминального сервера (ТС) на тонкий клиент, предусматривающее реализацию перечисленных выше функций. Данное решение представляет собой аппаратно-программный модуль доверенной сетевой загрузки (АПМДСЗ), позволяющий обеспечить целостность ОС с использованием инфраструктуры открытых ключей. В разрабатываемом АПМДСЗ алгоритм загрузки ОС заложен в прошивке сетевой карты. Загрузчик ОС размером 2–3 КБ хранится в защищенной памяти USB-ключа. Доступ к такой области памяти может получить только доверенный пользователь, что обеспечивает целостность загрузчика. Процесс защищенной загрузки ОС при этом происходит следующим образом. После подачи питания BIOS тонкого клиента исполняет программу микроконтроллера сетевой карты, запуская механизм загрузки. Программа микроконтроллера обращается к защищенной памяти USB-ключа и запускает загрузчик, который инициирует протокол сетевой загрузки. В рамках данного протокола образ ОС загружается с сервера в оперативную память тонкого клиента, после чего проверяется целостность образа до передачи управления на ОС. В случае успеха проверки управление передается на ОС.

Таким образом, в отличие от распространенных механизмов использования токена в качестве защищенного хранилища ключей для аутентификации, в описываемой схеме АПМДСЗ не может предоставить доступ к контейнеру, доверяя паролю пользователя, но не удостоверившись, что с другой стороны – подлинная СТД, причем еще до того, как произойдет загрузка ОС на тонкий клиент. Кроме того, в рамках реализации механизма загрузки ОС необходимо предусматривать и организацию защищенного обмена между сервером, АПМДЗ и токеном, с которого передается загрузчик.

Таким образом, одним из важнейших требований к функционалу разрабатываемого решения является доверенная аутентификация всех участвующих компонентов – сервера, тонкого клиента, токена и пользователя. Для реализации такого механизма взаимной аутентификации непосредственно перед загрузкой ОС должен быть разработан алгоритм, позволяющий реализовать следующие требования [6]:

- токен должен аутентифицировать тонкий клиент, т.е. тонкий клиент должен доказать токену, что входит в состав системы;
- тонкий клиент должен аутентифицировать токен, т.е. токен должен доказать тонкому клиенту, что является легитимным носителем и был выпущен администратором системы;
- тонкий клиент должен аутентифицировать пользователя, т.е. пользователь должен доказать системе, что предъявленный токен принадлежит ему, и он является легитимным пользователем системы.

Поставленная задача осложняется тем, что на практике, как правило, необходимо предусматривать работу многопользовательской СТД с терминальным сервером, хранящим ОС, и парком аппаратных тонких клиентов [7]. При этом на одном и том же тонком клиенте могут работать разные пользователи и иметь возможность после успешной аутентификации загружать персонифицированный образ ОС с назначенными именно им правами доступа.

Отдельно стоит заметить, что одним из основных предполагаемых целевых использований разрабатываемого модуля являются государственные информационные системы [8]. Это обусловлено наличием требований[1] по обеспечению доверенной загрузки ОС в нормативной документации в таких систе-

[1] Об утверждении Требований о защите информации, не составляющей государственную тайну, содержащейся в государственных информационных системах. Приказ ФСТЭК России от 11 февраля 2013 г. №17. Зарегистр. в Минюсте 31 мая 2013 г.

мах[1]. В силу этого дополнительным требованием к реализации механизма аутентификации является соответствие используемых криптографических алгоритмов российским стандартам (ГОСТ).

Основной сложностью, не позволяющей использовать в рассматриваемой ситуации классические алгоритмы аутентификации, является наличие третьей недоверенной стороны – токена: все популярные протоколы аутентификации и авторизации удаленных пользователей в условиях распределенной сетевой инфраструктуры (протоколы RADIUS, включающие механизмы PAP, CHAP, EAP [9], протокол Kerberos) рассчитаны на две стороны и без внесения серьезных изменений в их реализацию не могут быть применены. Кроме того, в применении уже зарекомендовавших себя алгоритмов присутствует формальный неблагоприятный момент – решения, построенного на их использовании, нельзя сертифицировать по соответствующим требованиям к средствам криптографической защиты и, как следствие, теряется возможность его применения в государственных информационных системах.

В настоящей работе рассматриваются два разработанных алгоритма аутентификации и их сравнительный анализ, позволяющий выбрать механизм, наиболее подходящий для решения поставленной задачи.

Описание предлагаемого решения

Исходя из результатов анализа угроз, рассматриваемых в отношении компонентов, участвующих в процессе загрузки ОС [10], первым этапом разрабатываемого алгоритма должна быть аутентификация СТД в лице тонкого клиента на токене. Этот шаг выполняется в первую очередь, так как необходимо обеспечить защиту хранящейся на токене информации (загрузчика ОС) от несанкционированного доступа со стороны потенциальных нарушителей, если предположить, что тонкий клиент подключен в нелегитимную систему. Только после этого пользователь, предварительно доказав владение токеном, может предоставить СТД возможность проверить легитимность загрузчика.

В качестве реализации аутентификации тонкого клиента на токене могут быть применены следующие механизмы [11]: алгоритм аутентификации с использованием одноразовых паролей и алгоритм аутентификации с использованием инфраструктуры открытых ключей. В следующих разделах приводится описание указанных алгоритмов, их достоинства и недостатки. При этом подразумевается, что описываемые механизмы аутентификации реализованы в АПМДСЗ [5], установленном в тонком клиенте.

Алгоритм аутентификации с использованием одноразовых паролей

Данный алгоритм предполагает наличие как на токене, так и в тонком клиенте списка одноразовых паролей или алгоритма генерации таких паролей. Генерация паролей может осуществляться с использованием односторонней хэш-функции или алгоритмов симметричного шифрования [12]. Запись паролей на токен должна осуществляться при ее инициализации администратором информационной безопасности. При этом можно выделить следующие требования к такому алгоритму:

– любой тонкий клиент из состава СТД, на котором пользователь хочет осуществить загрузку ОС, должен предъявлять токену один и тот же пароль (хранить на токене отдельные пароли для всех тонких клиентов технически слишком трудозатратно);
– на любом тонком клиенте из состава СТД должен быть реализован алгоритм генерации паролей с использованием закрытого ключа (ЗК), так как нет технической возможности хранить на тонком клиенте пароли для всех токенов.

Алгоритм работает со следующими данными: запрос токена – случайное число R, закрытый ключ тонкого клиента K, шифрограмма запроса токена на закрытом ключе тонкого клиента $E_K(R)$, пароль пользователя для доступа к токену. В качестве модулей, реализующих вычисления в рамках описываемого алгоритма, используются модуль токена и АПМДСЗ [5].

Описание алгоритма

Подобный алгоритм может быть реализован следующим образом.

Шаг 1. При первичной инициализации токена в его память сохраняется запрос – случайное число R, а также контрольный ответ, представляющий собой шифрограмму данного случайного числа на закрытом ключе тонкого клиента $K - E_K(R)$. Также на токене хранится эталонное значение пароля пользователя.

Шаг 2. При прохождении аутентификации тонкий клиент выводит для пользователя СТД на монитор форму авторизации с приглашением вставить токен.

Шаг 3. Пользователь вставляет токен в USB-порт тонкого клиента.

Шаг 4. Тонкий клиент запрашивает у токена записанное на него случайное число R.

Шаг 5. Токен предоставляет тонкому клиенту случайное число R.

[1] Автоматизированные системы. Защита от несанкционированного доступа к информации. Классификация автоматизированных систем и требования по защите информации. Рук. документ ФСТЭК России. Утв. решением Гостехкомиссии при Президенте РФ от 30 марта 1992 г.

Шаг 6. Тонкий клиент возвращает токену запрос, зашифрованный на своем закрытом ключе $K - E_K(R)$.

Шаг 7. Токен сравнивает полученный ответ с контрольным и, в случае их совпадения, аутентифицирует тонкого клиента.

Шаг 8. Тонкий клиент выводит на монитор запрос на введение пароля пользователя.

Шаг 9. Пользователь вводит пароль для доступа к загрузчику.

Шаг 10. Тонкий клиент передает введенное пользователем значение пароля на токен для сравнения с эталонным.

Шаг 11. Токен возвращает на тонкий клиент результат сравнения. В случае успеха алгоритм аутентификации завершен, иначе тонкий клиент повторно запрашивает пароль пользователя.

Шаг 12. После успешного завершения аутентификации тонкий клиент генерирует новую пару из запроса и ответа $\{R', E_K(R')\}$, которая записывается на токен для использования в следующем сеансе аутентификации.

В результате все тонкие клиенты в составе СТД должны использовать общий закрытый ключ K для генерации паролей, причем компрометация этого закрытого ключа приведет к возможности несанкционированного доступа ко всем выпущенным с данным ключом токенам.

Алгоритм аутентификации с использованием инфраструктуры открытых ключей

Алгоритм взаимной аутентификации пользователя, токена, тонкого клиента и терминального сервера с использованием инфраструктуры открытых ключей [13] реализуется следующим образом. Аутентификация токена в системе СТД производится за счет наличия на нем собственного закрытого ключа аутентификации и соответствующего сертификата открытого ключа (ОК). Генерация ключей осуществляется непосредственно на токене в процессе персонализации, формирование и выдача сертификатов выполняются удостоверяющим центром (УЦ). Терминальный сервер является доверенным компонентом рассматриваемой СТД. Аутентификация СТД в лице тонкого клиента на токене производится за счет наличия на терминальном сервере собственного закрытого ключа и соответствующего сертификата ОК. Генерация ключей осуществляется также непосредственно на сервере при развертывании СТД, сертификаты серверу формирует и выдает тот же УЦ. Аутентификация пользователя СТД при этом осуществляется за счет проверки пароля при запросе доступе к загрузчику, хранящемуся на токене.

При рассмотрении данного алгоритма предполагается, что тонкие клиенты расположены в пределах контролируемой зоны, что предусматривает наличие организационных мер, направленных на предотвращение доступа нарушителей к компонентам СТД и предотвращение модификации аппаратного обеспечения тонкого клиента. При этом непосредственно информационное взаимодействие токена, тонкого клиента и терминального сервера защищается с использованием алгоритмов шифрования и выработки имитовставки (ГОСТ 28147-89[1]) на сеансовых ключах. Формирование сеансовых ключей должно осуществляться на терминальном сервере с использованием криптопровайдера, реализующего алгоритмы ГОСТ при получении соответствующего запроса от тонкого клиента.

При этом сформированные ключи передаются на тонкий клиент по защищенном каналу связи: сеансовые ключи защищаются с использованием вспомогательных в соответствии с рекомендациями п. 5.2 RFC 4357[2]. Генерация вспомогательных ключей производится на токене и на терминальном сервере независимо после обмена сертификатами открытого ключа в соответствии со стандартом ГОСТ 34.11-94[3].

В качестве модулей, реализующих все функциональные вычисления в рамках описываемого алгоритма, используются криптографический модуль токена (с обязательной поддержкой криптографических алгоритмов ГОСТ) и криптографический модуль терминального сервера (программный криптопровайдер, также реализующий алгоритмы ГОСТ) и АПМДСЗ [5].

В состав данных, с которыми работает алгоритм, входят ключевая пара (открытый и закрытый ключи) токена, сертификат открытого ключа токена, ключевая пара (открытый и закрытый ключи) терминального сервера, сертификат ОК терминального сервера, сертификат ОК УЦ, пароль пользователя для доступа к контейнеру с ключевой парой токена.

Алгоритм взаимной аутентификации с использованием инфраструктуры открытого ключа представлен ниже, а также проиллюстрирован на рисунке.

[1] ГОСТ 28147-89. Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования. Введ. 01.07.1990. М.: Госстандарт, 1989.

[2] RFC 4357. Дополнительные криптографические алгоритмы для использования совместно с ГОСТ 28147-89, ГОСТ 34.10-94, ГОСТ 34.10-2001 и ГОСТ 34.11-94. 2006.

[3] ГОСТ 34.311-95. Информационная технология. Криптографическая защита информации. Функция хэширования. Введ. 01.01.1995. М.: Издательство стандартов, 1994.

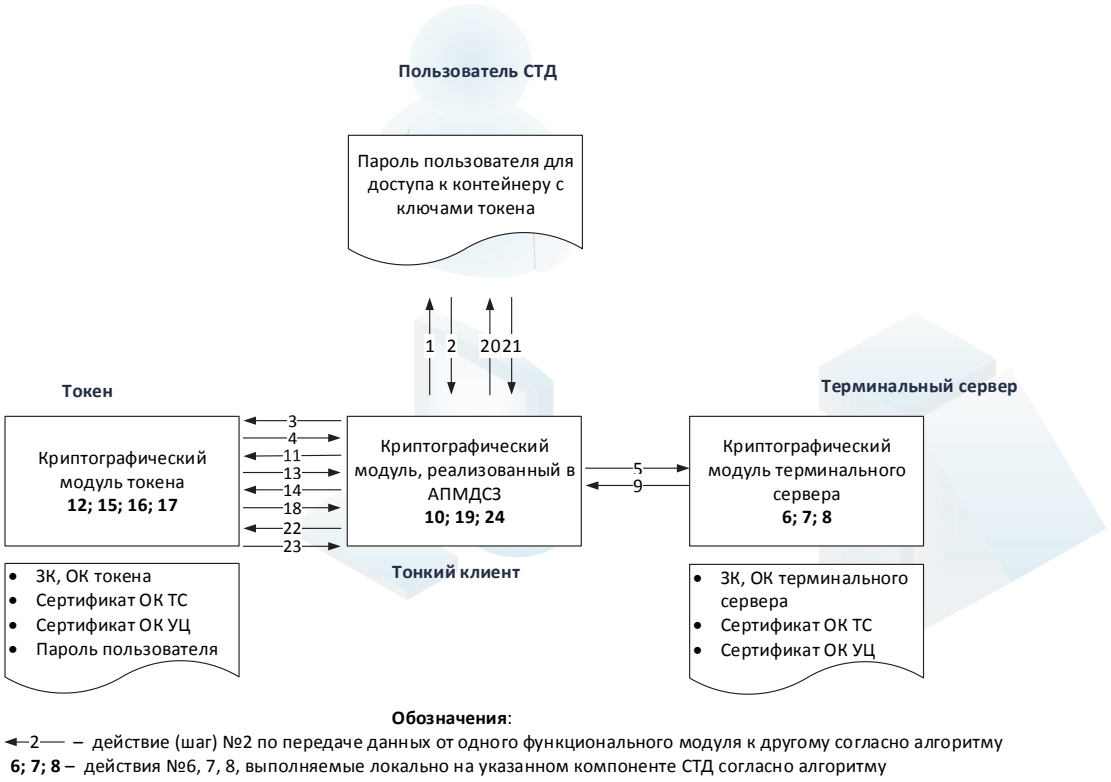

Рисунок. Иллюстрация алгоритма на основе инфраструктуры открытого ключа

Описание алгоритма

Шаг 1. Тонкий клиент выводит для пользователя СТД на монитор форму авторизации с приглашением вставить токен.

Шаг 2. Пользователь вставляет токен в usb-порт тонкий клиент.

Шаг 3. Тонкий клиент передает токену запрос на считывание его идентификатора и сертификата ОК аутентификации токена.

Шаг 4. Токен передает на тонкий клиент идентификатор и сертификат ОК аутентификации токена (данные передаются в открытом виде).

Шаг 5. Тонкий клиент передает терминальному серверу запрос на генерацию сеансовых ключей шифрования. В запрос включается сертификат ОК аутентификации токена, полученный на шаге 4.

Шаг 6. Терминальный сервер осуществляет проверку электронной подписи (ЭП) полученного сертификата ОК токена с использованием сертификата ОК УЦ, который хранится непосредственно на сервере. Также терминальный сервер производит проверку актуального статуса сертификата ОК токена. Если ЭП сертификата не верна, либо сертификат был отозван, переход на шаг 9.

Шаг 7. Терминальный сервер осуществляет генерацию случайных чисел R_1, R_2 и вспомогательных симметричных ключей шифрования K_1 и K_2 на основе собственного ЗК и ОК токена в соответствии с рекомендациями п. 5.2 RFC 4357.

Шаг 8. Терминальный сервер генерирует сеансовые ключи шифрования K_c и выработки имитовставки K_{im} в соответствии со стандартом ГОСТ 28147-89.

Шаг 9. Терминальный сервер передает ответное сообщение на тонкий клиент. При успешном выполнении шагов 6–8 в сообщение включаются сертификат ОК терминального сервера, сеансовые ключи K_c и K_{im} для использования на тонком клиенте, блок данных для передачи на токен (подробнее см. шаг 14 данного алгоритма). При возникновении ошибок на шагах 6–8 в сообщение включается только соответствующий код ошибки.

Шаг 10. При получении от терминального сервера сообщения об успешной проверке сертификата ОК токена, тонкий клиент сохраняет полученные сеансовые ключи. Если от терминального сервера получен код ошибки, тонкий клиент выводит на монитор пользователю СТД соответствующее сообщение, посылает токену запрос на временную блокировку и выполнение алгоритма аутентификации завершается.

Шаг 11. Если на шаге 10 получено сообщение об успешной проверке, тонкий клиент передает на токен запрос на инициализацию протокола взаимной аутентификации, содержащий сертификат ОК терминального сервера, полученный от него на шаге 9.

Шаг 12. Токен осуществляет проверку ЭП полученного сертификата с использованием ОК УЦ, который хранится на токене. ОК терминального сервера извлекается из сертификата и сохраняется в памяти токена. Токен формирует и передает на тонкий клиент ответное сообщение, которое служит подтверждением того, что токен произвел проверку ЭП. Если при выполнении шагов 11–12 возникла ошибка, токен включает в ответное сообщение код ошибки и выполнение алгоритма аутентификации завершается.

Шаг 13. Если на шаге 13 токен передает положительное ответное сообщение, тонкий клиент передает на токен блок данных, полученных от терминального сервера на шаге 9. Блок данных представляет собой конкатенацию

$R_1 \| E(K_c)_{K_1} \| I(K_c)_{K_1,R_1} \| R_2 \| E(K_{im})_{K_2} \| I(K_{im})_{K_2,R_2}$,

где R_1, R_2 – случайные числа, сгенерированные на шаге 7; $E(K_c)_{K_1}$, $E(K_{im})_{K_2}$ – шифрограммы сеансовых ключей K_c и K_{im} на вспомогательных ключах K_1 и K_2 (стандарт ГОСТ 28147-89, режим простой замены); $I(K_c)_{K_1,R_1}$, $I(K_{im})_{K_2,R_2}$ – значения имитовставки, вычисленные по стандарту ГОСТ 28147-89 (режим выработки имитовставки) на вспомогательных ключах K_1 и K_2 с использованием чисел R_1, R_2 в качестве векторов инициализации и сеансовых ключей K_c и K_{im} в качестве входных данных.

Шаг 14. Токен извлекает из полученного сообщения значения R_1, R_2 и выполняет формирование вспомогательных ключей шифрования K_1 и K_2 на основе ЗК токена и ОК терминального сервера, полученного на шаге 11, в соответствии с рекомендациями п. 5.2 RFC 4357.

Шаг 15. Токен выполняет расшифровку сеансовых ключей K_c и K_{im} с использованием вспомогательных.

Шаг 16. Токен выполняет подсчет контрольных значений имитовставки в соответствии с ГОСТ 28147-89 и сравнивает полученные значения с контрольными. Если проверка имитовставки прошла успешно, токен сохраняет сеансовые ключи в своей памяти, инициализирует протокол защищенного взаимодействия с использованием сеансовых ключей, предоставляет тонкому клиенту доступ к загрузчику.

Шаг 17. Токен передает на тонкий клиент ответное сообщение. В случае успешного выполнения шагов 15–17 сообщение служит подтверждением успешной аутентификации тонкого клиента на токене. В противном случае сообщение содержит код ошибки, возникшей при выполнении шагов 15–17 данного алгоритма, и выполнение алгоритма аутентификации завершается.

Шаг 18. В случае положительного ответного сообщения от токена его аутентификация на тонком клиенте осуществляется за счет проверки имитовставки данного сообщения следующим образом: вычисляется контрольная имитовставка как результат шифрования полученной шифрограммы на ключе K_{im} (ГОСТ 28147-89, режиме выработки имитовставки). После этого производится сравнение полученной имитовставки с контрольной: если результат сравнения отрицательный, тонкий клиент прерывает выполнение протокола и прекращает сеанс связи с токеном; положительный результат подтверждает факт наличия на токене сеансовых ключей, а следовательно, и ЗК аутентификации токена.

Шаг 19. Тонкий клиент выводит на монитор запрос на введение пароля пользователя.

Шаг 20. Пользователь вводит пароль для доступа к загрузчику.

Шаг 21. Тонкий клиент передает на токен пароль пользователя для сравнения.

Шаг 22. Токен сравнивает полученный пароль с эталонным: при положительном ответе алгоритм аутентификации успешно завершен и осуществляется передача управления на АПМДСЗ; если значения паролей отличаются, токен сообщает об этом на тонкий клиент.

Шаг 23. Если на шаге 23 токен передал отрицательный результат проверки, тонкий клиент выводит об этом сообщение на экран и запрашивает пароль пользователя повторно (переход на шаг 20). При введении пользователем неправильного пароля заданное количество раз подряд токен блокируется. Разблокирование токена осуществляет администратор информационной безопасности.

Поскольку одной из важных функциональных особенностей инфраструктуры открытого ключа является проверка сертификатов открытых ключей на отозванность, все участвующие компоненты должны иметь возможность получать такую информацию. Данный функционал может быть реализован либо указанием в самом сертификате адреса сервера, на котором расположен актуальный (как правило, выпускающийся ежедневно) список отозванных сертификатов из УЦ, либо организацией службы Online Certificate Status Protocol для получения актуальной информации о статусе сертификата в реальном режиме времени из базы УЦ [13]. Исходя из описания алгоритма, в проверке статуса сертификата нуждаются криптомодули токена и терминального сервера. Но если терминальный сервер может получить такую информацию любым из указанных способов, находясь в доверенной сетевой инфраструктуре, то токену, чтобы получить либо список отозванных сертификатов либо обратиться к OCSP-серверу, необходимо аутентифицироваться в системе, что в итоге приводит к замкнутому кругу. Таким образом, при реализации алгоритма аутентификации следует учитывать, что токен не может распознать ситуацию, когда предъявленный ему сертификат уже недействителен, и необходимым требованием к безопасности в данном случае является гарантированное уничтожение выведенных из обращения закрытых ключей терминального сервера и УЦ СТД при их смене.

Сравнение алгоритмов

Если сравнивать два предлагаемых алгоритма с точки зрения выполнения требований, предъявляемых при постановке задачи, то аутентификация всех участвующих компонентов полноценно реализуется в обоих случаях. То же относится и к требованию применимости рассматриваемых алгоритмов в государственных информационных системах в отношении реализации отечественной криптографии.

Однако если смотреть на предлагаемые алгоритмы с точки зрения многопользовательской архитектуры СТД, то алгоритм с использованием одноразовых паролей в недостаточной мере удовлетворяет требованиям безопасности: в результате использования общего ЗК, который должен храниться на всех тонких клиентах для возможности аутентификации токенов, в системе появляется серьезная уязвимость. При этом альтернативный алгоритм изначально строится на асимметричной криптографии, т.е. подразумевается, что закрытые ключи всех участников никогда не передаются в процессе аутентификации. Но необходимо отметить важное требование к применению обоих алгоритмов – это обеспечение контрольных мер против несанкционированного доступа к самим тонким клиентам в целях предотвращения модификации их программно-аппаратного обеспечения.

Таким образом, к преимуществам алгоритма с одноразовыми паролями можно отнести простоту реализации с технической точки зрения, но существенными недостатками являются низкая масштабируемость системы и недостаточный выдвигаемым требованиям уровень безопасности. В свою очередь, преимуществами алгоритма с использованием инфраструктуры открытых ключей относятся простота централизованного управления (в том числе выдача ключей и сертификатов), масштабируемость решения, простота использования, высокий уровень обеспечиваемой безопасности. Но существенным недостатком по сравнению с предыдущим алгоритмом является необходимость развертывания инфраструктуры открытых ключей, что, безусловно, несет дополнительные эксплуатационные расходы.

Кроме того, в отличие от существующих алгоритмов выработки общего секретного ключа в процессе клиент-серверной аутентификации, важным преимуществом алгоритма на основе инфраструктуры открытых ключей является возможность реализовать аутентификацию на основе пар открытых и закрытых ключей и организовать защищенный обмен всех участвующих сторон с использованием криптографических алгоритмов ГОСТ, несмотря на то, что отечественного стандарта на асимметричную криптографию в настоящий момент нет.

Заключение

Предлагаемый в настоящей работе метод аутентификации позволяет решить проблему проверки подлинности всех компонентов, задействованных в процессе загрузки операционной системы на тонкий клиент по сети с терминального сервера. В результате проведенного сравнения двух разработанных алгоритмов аутентификации, один из которых основан на применении одноразовых паролей, а второй – на инфраструктуре открытых ключей, были выявлены их основные достоинства и недостатки. Несмотря на то, что оба алгоритма реализуют предъявляемые к ним базовые требования по трехстороннему подтверждению подлинности с использованием криптоалгоритмов ГОСТ, алгоритм на основе одноразовых паролей не обеспечивает должный уровень информационной безопасности в силу необходимости хранения единого закрытого ключа на всех тонких клиентах. В свою очередь, алгоритм, основанный на применении инфраструктуры открытых ключей, реализует аутентификацию с использованием асимметричных методов криптографии. Это, во-первых, позволяет избежать хранения в системе терминального доступа закрытых ключей каждого токена, что потребовало бы существенных затрат системных ресурсов. Во-вторых, такой подход позволяет обеспечить возможность проведения взаимной аутентификации токена на каждом тонком клиенте из состава системы терминального доступа без хранения единого закрытого ключа и, соответственно, минимизировать ущерб от его возможной компрометации. Таким образом, на основании вышеизложенного сравнительного анализа для решения задачи по реализации доверенной аутентификации компонентов, участвующих в процессе сетевой загрузки операционной системы на тонкий клиент, рекомендуется использовать алгоритм, основанный на инфраструктуре открытых ключей.

References

1. Mikhalevich I.F. Problems of creation of trusted environment of functioning of the automated control systems, protected construction. *XII Vserossiiskoe Soveshchanie po Problemam Upravleniya VSPU-2014* [XII All-Russia meeting on Control Problems VSPU 2014]. Moscow, 2014, pp. 9201–9207.
2. Shpunt Ya. Using thin clients. Benefits, costs, and pitfalls. *Intelligent Enterprise/RE*, 2011, no. 5(227), pp. 54–55. (In Russian)
3. Kohlenberg T., Ben-Shalom O., Dunlop J., Rub J. Evaluating thin-client security in a changing threat landscape. *IT@Intel White Paper*, 2010, 8 p.
4. Novikov S.V., Zima V.M., Andrushkevich D.V. Approach to building securer distributed networks of data processing based on trusted infrastructure. *SPIIRAS Proceedings*, 2015, vol. 38, no. 1, pp. 34–51. (In Russian)

5. Gatchin Yu.A., Teploukhova O.A. Integrity monitoring implementation for the operating system image loaded through a network to the thin clients. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2015, vol. 15, no. 6, pp. 1115–1121. doi: 10.17586/2226-1494-2015-15-6-1115-1121
6. Balmer S. *Trusted Computing Base Extension Control System For Client Workstations*. Masters Thesis. Monterey, California, Naval Postgraduate School, 1999, 118 p.
7. Balmer S.R., Irvine C.E. Analysis of terminal server architectures for thin clients in a high assurance network. *Proc. 23rd National Information Systems Security Conference*. Baltimore, MD, 2000, pp. 192–202.
8. Gatchin Yu.A., Teploukhova O.A. Realization of the protected connection to the state information systems on the basis of the slim client. *The International Technical-Economic Journal*, 2015, no. 5, pp. 55–62.
9. van der Walt D. *FreeRADIUS. Manage your network resources with FreeRADIUS. Beginner's Guide*. Packt Publishing, 2011, 344 p.
10. Teploukhova O.A. Building a model of security threats operating system image loaded over the network to the thin client terminal access systems. *Sbornik Tezisov Dokladov III Vserossiiskogo Kongressa Molodykh Uchenykh* [Proc. III All-Russian Congress of Young Scientists]. St. Petersburg, 2014, no. 1, pp. 235–237. (In Russian)
11. Smith R.E. *Authentication: From Passwords to Public Keys*. Addison-Wesley Professional, 2001, 576 p.
12. Alferov A.P., Zubov A.Yu., Kuz'min A.S., Cheremushkin A.V. *Osnovy Kriptografii* [Basics of Cryptography]. Moscow, Gelios ARV, 2005, 480 p.
13. Gorbatov V.S., Polyanskaya O.Yu. *Osnovy Tekhnologii PKI* [Basics of PKI Technology]. Moscow, Goryachaya Liniya – Telekom, 2004, 248 p.

Гатчин Юрий Арменакович – доктор технических наук, профессор, заведующий кафедрой, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, gatchin@mail.ifmo.ru

Теплоухова Ольга Александровна – аспирант, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, teplouhovaoa@gmail.com

Yuri A. Gatchin – D.Sc., Professor, Head of Chair, ITMO University, Saint Petersburg, 197101, Russian Federation, gatchin@mail.ifmo.ru

Olga A. Teploukhova – postgraduate, ITMO University, Saint Petersburg, 197101, Russian Federation, teplouhovaoa@gmail.com