Информатика, вычислительная техника и управление

DOI 10.36622/1729-6501.2025.21.4.001
УДК 004.056

АЛГОРИТМ ПРИМЕНЕНИЯ КАСКАДНОГО ШИФРОВАНИЯ И ФУНКЦИЙ ФОРМИРОВАНИЯ КЛЮЧА В УСЛОВИЯХ ПОТОКОВОЙ ОБРАБОТКИ ДАННЫХ

М.С. Войтенко, В.Ф. Барабанов

Воронежский государственный технический университет, г. Воронеж, Россия

Аннотация: рассматривается актуальная проблема обеспечения безопасности данных при передаче и хранении. Предлагаемый алгоритм рассчитан на работу в поточном режиме с использованием каскадного (двойного) шифрования, что позволяет эффективно шифровать потоки данных, без создания и хранения файла с оригинальными данными на диске, а также исключает действия, связанные с повторным чтением шифруемой информации. В основе алгоритма лежит использование двух поточных шифров (ChaCha20 и HC-256) и функций формирования ключа – Argon2 и scrypt. Ключевыми особенностями алгоритма являются: использование каскадного шифрования поточными шифрами совместно с функциями формирования ключа для обеспечения уникальности зашифрованных данных даже при идентичных исходных данных; механизм шифрования ключа позволяет изменять пароли без необходимости полного повторного шифрования данных. Объектом рассмотрения в рамках данной статьи является алгоритм, обеспечивающий высокий уровень защиты, где для доступа к данным злоумышленнику потребуется одновременно подобрать два независимых ключа, что вычислительно крайне затратно, при использовании современного оборудования. Предложенное решение предназначено для безопасного долгосрочного хранения и передачи конфиденциальной информации, например, паролей, персональных и других чувствительных данных

Ключевые слова: шифрование, поточное шифрование, каскадное шифрование, функция формирования ключа

Введение

В настоящее время использование информационных технологий требует соблюдения мер безопасности для хранимых и передаваемых данных. Без использования шифрования передаваемая информация будет доступна для перехвата злоумышленниками: от личных сообщений и фотографий до данных банковских карт и коммерческой тайны. Подавляющее большинство данных передаётся посредством сети Интернет. В данной статье предложен алгоритм применения каскадного шифрования поточными шифрами и функциями формирования ключа. Алгоритм рассчитан на работу в поточном режиме, то есть данные читаются последовательно и только один раз. Это позволяет шифровать потоки данных, такие как видео или данные с датчиков, без задержек, связанных с необходимостью загружать весь объем данных на устройство в оперативную память или на диск.

В алгоритме используются поточные шифры. Поточный шифр – это симметричный алгоритм, в котором шифрование каждого элемента открытого текста зависит от текущего состояния алгоритма и производится в реальном времени [1].

Роль пароля и функции формирования ключа

Пароль подаётся в формате base64 [2]. Использование base64 обусловлено наличием возможности использовать в пароле символ переноса строки (0xA) как часть пароля. Символ переноса строки предполагается использовать для разделения одного пароля от другого, поэтому необходимо выполнить преобразование пароля для использования в алгоритме. Поступающий пароль будет декодирован из формата base64, что обеспечивает корректную обработку пароля независимо от используемых в пароле значений байт.

Когда пароль введён, срабатывает функция формирования ключа (key derivation function или KDF). KDF – это функция, формирующая один или несколько секретных ключей на основе секретного значения [3]. KDF участвует в шифровании ключа для последующего хранения, а также значительно усложняет процесс взлома методом перебора. Ключ представляет из себя случайную последовательность (например, /dev/random), которая будет использоваться в алгоритмах шифрования данных. Применение случайной последовательности делает зашифрованные данные уникальными, даже при одинаковых исходных данных.

В качестве функций формирования ключа предполагается использовать Argon2 и/или scrypt. Ключевой особенностью данных функций является интенсивное использование вычислительных ресурсов и оперативной памяти, что делает массовый параллельный перебор паролей на специализированном оборудовании крайне затратным и медленным [4, 5].

Обработанный функцией пароль используется для шифрования случайного секретного ключа с помощью логической операции XOR. Основное преимущество данного подхода в том, что пароль и производная от пароля не используются для непосредственного шифрования данных. Шифруется случайный ключ, который применяется в поточных шифрах. Это также позволяет менять пароли без необходимости повторно шифровать данные. Повторному шифрованию подвергнутся только ключи.

Результатом операции будет служить зашифрованный секретный ключ. Схема шифрования пароля обозначена на рис. 1.

Алгоритм шифрования ключа выглядит следующим образом:

$$EK = KDF(password) \oplus key,$$

где EK – это зашифрованный ключ (encrypted key). Аналогично выглядит алгоритм дешифрования:

$$key = KDF(password) \oplus EK.$$

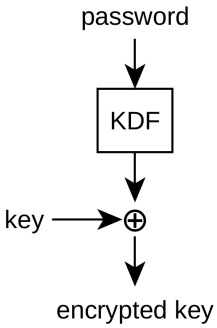

Рис. 1. Шифрование ключа с применением KDF

Каскадное шифрование данных

Предполагается шифровать дважды: первый раз с использованием потокового шифра ChaCha20, второй – HC-256, так же потокового.

На рис. 2 показана схема применения двойного шифрования данных.

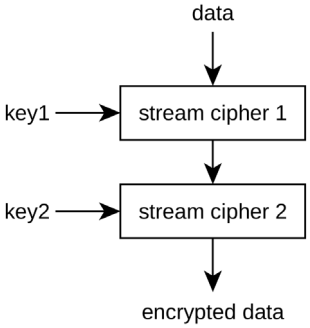

Рис. 2. Схема применения двойного шифрования данных

Оба перечисленных шифра являются поточными, то есть данные шифруются операцией XOR с псевдослучайной последовательностью, генерируемой на основе ключа. Двойное шифрование обеспечивает дополнительный уровень защиты, поэтому если один из ключей или алгоритмов шифрования будет скомпрометирован, данные останутся защищёнными благодаря второму уровню шифрования. Для взлома злоумышленнику потребуется подобрать оба ключа одновременно, что вычислительно крайне затратно при использовании современного оборудования.

Алгоритм двойного шифрования выглядит следующим образом:

$$ED = cipher2(key2, cipher1(key1, data)),$$

где ED – это зашифрованные данные (encrypted data). Аналогично выглядит алгоритм для дешифрования:

$$data = cipher1(key1, cipher2(key2, ED))$$

Ввод и вывод данных

Порядок ввода и вывода данных представлен на рис. 3. В алгоритм подаётся два пароля (в формате base64 каждый) по очереди, производится операция шифрования или дешифрования (если используется соответствующий аргумент или переменная окружения). Выходные данные при шифровании будут следующими: первые 32 байта занимает зашифрованный ключ, вторые 32 байта – второй ключ, после чего некоторое место занимают данные, а последние 32 байта занимает зашифрованная хеш-сумма (например, «Стрибог» в режиме 256 бит [6]) оригинальных данных для проверки целостности.

В случае, если данные подлежат дешифровке, входной поток будет идентичным, но выходной поток будет содержать расшифрованную информацию. Последний байт в потоке, после окончания дешифровки, служит меткой о целостности расшифрованных данных на основе проверки хэш-суммы.

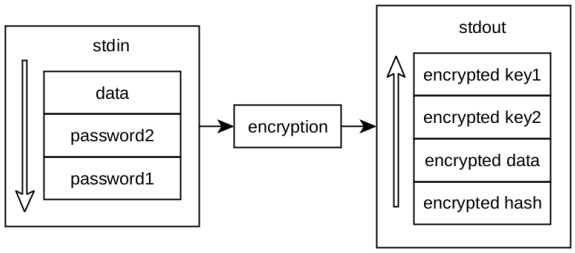

Рис. 3. Порядок ввода и вывода данных (режим шифрования)

Параметры функций формирования ключа

Параметры KDF следует задавать таким образом, чтобы использовать наибольший доступный объем оперативного запоминающего устройства (ОЗУ) и приемлемое время при формировании ключа. Это создаст дополнительные затраты атакующему, так как в случае подбора ключа грубой силой, будет необходимо применять идентичные параметры KDF.

Набор параметров, использующий 4гб ОЗУ и затрачивающий 32,76 секунд в Argon2d будет следующий: *p=4096000, t=15, p=1*.

Для Scrypt параметры будут следующие: *log_n=22, r=8, p=4*. Использование ОЗУ будет идентичным, а затрачиваемое время будет около 30,73 секунд.

Были использованы библиотеки на языке RUST: argon2 версии 0.5.3 и scrypt версии 0.11.0. Затрачиваемое время на формирование ключа было определено в ходе проведения тестов используя процессор AMD Ryzen 5 3600x в однопоточном режиме.

Следует отметить, что процессы формирования ключей не требуют последовательного исполнения и могут быть разделены на несколько потоков, но в таком случае объем использования ОЗУ будет суммироваться.

Заключение

Представленный в данной статье алгоритм применения каскадного шифрования с применением поточных шифров и функциями формирования ключа можно использовать для долгосрочного хранения чувствительных данных: паролей, данных паспорта, договоров и других частных данных, нуждающихся в защите от несанкционированного доступа при хранении и передаче посредством сети Интернет.

Конечный размер данных будет увеличен на 96 байт (64 байта в начале сообщения и 32 байта в конце сообщения), что позволяет прогнозировать итоговый размер зашифрованных данных.

В случае применения использования иных алгоритмов шифрования или функции хэширования, которые имеют отличные от представленных объемы ключей или хеш-сумм, итоговый объем данных можно вычислить следующим образом:

$$S = \sum Sk_i + Sd + Ssh,$$

где S – итоговый объем, Sk – объем ключа, Ssh – объем хэш-суммы.

Алгоритм может иметь ограничения по количеству данных, подлежащих шифрованию. Поточные шифры имеют внутренние n-битные счётчики, что задаёт данное ограничение. ChaCha20 имеет ограничение в 256 Гб, так как имеет 32-битный счётчик и использует 8 байт на блок [7]. Шифр HC-256 же способен генерировать ключевую последовательность длиной до 2^{128} бит [8].

Решением данной проблемы является замена алгоритмов потокового шифрования на другие с большим счётчиком состояний или создание производных ключа для генерации новых псевдослучайных последовательностей, однако, второй вариант может повлечь неустановленные риски криптостойкости, так как создаёт связь между генерируемыми последовательностями.

Другим явным недостатком является необходимость использования двух разных паролей, что может оказаться трудным для запоминания или внести дополнительную сложность для реализации автоматической системы шифрования.

Литература

1. Криптографическая защита информации // Н.А. Гатченко, А.С. Исаев, А.Д. Яковлев. СПб: НИУ ИТМО, 2012. 142 с.

2. RFC 4648: The Base16, Base32, and Base64 Data Encodings. URL: https://datatracker.ietf.org/doc/html/rfc4648# section-4 (дата обращения 18.09.2025)

3. Data privacy / M. Bezzi [et al.] // Privacy and Identity Management for Life. Springer, 2011. pp. 185-186.

4. Alwen J., Blocki J. Efficiently Computing Data-Independent Memory-Hard Functions // Advances in Cryptology. CRYPTO 2016. pp. 241-271.

5. RFC 7914: The scrypt Password-Based Key Derivation Function. URL: https://datatracker.ietf.org/doc/html/rfc7914.html (дата обращения: 18.09.2025).

6. ГОСТ 34.11-2018. Информационная технология. Криптографическая защита информации. Функция хеширования. М.: Стандартинформ, 2018. 25 с.

7. Королев И.Ф. Эффективная реализация поточного шифра CHACHA20 // Вестник Сыктывкарского университета. Сер. 1: Математика. Механика. Информатика. 2017. Вып. 4 (25). С. 33-43.

8. Wu H. A New Stream Cipher HC-256. URL: https://www.iacr.org/archive/fse2004/30170226/30170226. pdf (дата обращения: 18.09.2025).

Поступила 20.10.2025; принята к публикации 10.11.2025

Информация об авторах

Войтенко Максим Сергеевич – аспирант, Воронежский государственный технический университет (394006, Россия, г. Воронеж, ул. 20-летия Октября, 84), e-mail: voytenkom@yandex.ru
Барабанов Владимир Федорович – д-р техн. наук, профессор, Воронежский государственный технический университет (394006, Россия, г. Воронеж, ул. 20-летия Октября, 84), e-mail: bvf@list.ru

ALGORITHM FOR APPLYING CASCADE ENCRYPTION AND KEY DERIVATION FUNCTIONS IN STREAM DATA PROCESSING ENVIRONMENT

M.S. Voytenko, V.F. Barabanov

Voronezh State Technical University, Voronezh, Russia

Abstract: this article addresses the current problem of ensuring data security during transmission and storage. The proposed algorithm is designed to operate in a stream mode using cascade (double) encryption, which enables efficient encryption of data streams without creating and storing a file with the original data on the disk; it also eliminates the need for re-reading the information being encrypted. The algorithm is based on the use of two stream ciphers (ChaCha20 and HC-256) and key derivation functions - Argon2 and scrypt. The key features of the algorithm are: the use of cascade encryption with stream ciphers combined with key derivation functions to ensure the uniqueness of encrypted data even when the original data is identical; the key encryption mechanism allows for changing passwords without the need for complete data re-encryption. The object of consideration in this article is an algorithm that provides a high level of protection, where an attacker would need to simultaneously compromise two independent keys to access the data, which is computationally prohibitive even with modern equipment. The proposed solution is intended for the secure long-term storage and transmission of confidential information, such as passwords, personal, and other sensitive data

Key words: encryption, stream cipher, cascade encryption, key derivation function

References

1. Ganchenko N.A., Isaev A.S., Yakovlev A.D. “Cryptographic information security” (“Kriptograficheskaya zashchita informatsii”), ITMO University, 2012, 142 p.
2. RFC 4648: The Base16, Base32, and Base64 Data Encodings, available at: https://datatracker.ietf.org/doc/html/rfc4648#section-4 (date of access: 18.09.2025).
3. Bezzi M. et al. “Data privacy”, *Privacy and Identity Management for Life*, Springer, 2011, pp. 185-186.
4. Alwen J., Blocki J. “Efficiently computing data-independent memory-hard functions”, *Advances in Cryptology. CRYPTO 2016,* pp. 241-271.
5. RFC 7914: The scrypt Password-Based Key Derivation Function, available at: https://datatracker.ietf.org/doc/html/rfc7914.html (date of access: 18.09.2025).
6. GOST 34.11-2018 “Information technology. Cryptographic data security. Hash-function”, 2018, 25 p.
7. Korolev I.F. “Efficient implementation of ChaCha20 stream cipher”, *Bulletin of Syktyvkar University, Series 1: Mathematics. Mechanics. Infotmatics (Vestnik Syktyvkarskogo universiteta. Ser. 1: Matematika. Mekhanika. Informatika)*, 2017, no. 4 (25), pp. 33-43.
8. Hongjun Wu. “A New Stream Cipher HC-256”, available at: https://www.iacr.org/archive/fse2004/30170226/30170226.pdf (date of access: 18.09.2025).

Submitted 20.10.2025; revised 10.11.2025

Information about the authors

Maksim S. Voytenko, graduate student, Voronezh State Technical University (84 20-letiya Oktyabrya, Voronezh 394006, Russia), e-mail: voytenkom@yandex.ru
Vladimir F. Barabanov, Dr. Sc. (Technical), Professor, Voronezh State Technical University (84 20-letiya Oktyabrya, Voronezh 394006, Russia), e-mail: bvf@list.ru