Министерство науки и высшего образования Российской Федерации
НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Ю.А. КОТОВ

ПРИЛОЖЕНИЯ ШИФРОВ КРИПТОАНАЛИЗ

Утверждено
Редакционно-издательским советом университета
в качестве учебного пособия

НОВОСИБИРСК
2019

УДК 004.056.55(075.8)
К 736

Рецензенты:
д-р техн. наук, проф. *В.И. Гужов,*
канд. техн. наук, доцент *Г.В. Трошина*

Работа подготовлена на кафедре защиты информации

Котов Ю.А.
К 736 Приложения шифров. Криптоанализ: учебное пособие / Ю.А. Котов. – Новосибирск: Изд-во НГТУ, 2019. – 76 с.

ISBN 978-5-7782-3902-9

Представлены приложения шифров для аутентификации данных и субъектов, формирования электронно-цифровой подписи, основные подходы к генерации псевдослучайных числовых последовательностей и криптоанализу шифров.

Предназначено для студентов, обучающихся по направлению 10.03.01 «Информационная безопасность» и специальности 10.05.03 «Информационная безопасность автоматизированных систем».

УДК 004.056.55(075.8)

Котов Юрий Алексеевич

ПРИЛОЖЕНИЯ ШИФРОВ
КРИПТОАНАЛИЗ

Учебное пособие

Редактор *М.О. Мокшанова*
Выпускающий редактор *И.П. Бровановa*
Корректор *И.Е. Семенова*
Дизайн обложки *А.В. Ладыжская*
Компьютерная верстка *Н.В. Гаврилова*

Налоговая льгота – Общероссийский классификатор продукции
Издание соответствует коду 95 3000 ОК 005-93 (ОКП)

Подписано в печать 28.05.2019. Формат 60 × 84 1/16. Бумага офсетная
Тираж 50 экз. Уч.-изд. л. 4,41. Печ. л. 4,75. Изд. № 317/18. Заказ № 937. Цена договорная

Отпечатано в типографии
Новосибирского государственного технического университета
630073, г. Новосибирск, пр. К. Маркса, 20

ISBN 978-5-7782-3902-9

1. ПРИЛОЖЕНИЯ ШИФРОВ

Помимо своего прямого назначения – шифрования – криптографические преобразования используются в компьютерной защите информации в следующих важных целях:

1) для аутентификации электронных данных (в том числе и программ – антивирусная защита);

2) для аутентификации электронных субъектов (пользователей информационно-телекоммуникационных систем).

Под электронными данными и документами здесь и далее будем понимать их двоичное представление в вычислительной среде, т. е. обыкновенную двоичную последовательность. Использовать шифры для аутентификации данных и субъектов можно и вне электронной среды и двоичной формы представления, никаких ограничений для этого, кроме технических, нет.

Принципиальное отличие аутентификации от идентификации заключается в том, что множество идентифицирующих признаков является внешним по отношению к реальному объекту (его дополнением), в то время как аутентифицирующие признаки – неотъемлемая (вплоть до нарушения целостности) часть объекта (рис. 1).

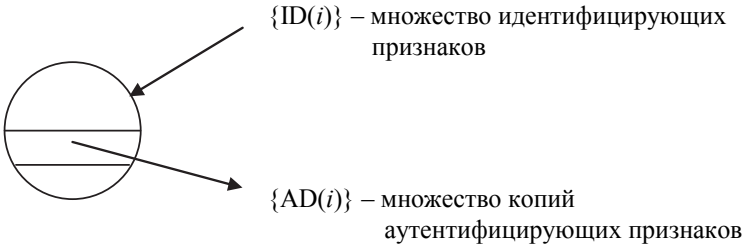

Рис. 1. Отличие аутентификации от идентификации

Шифры и процедуры аутентификации данных и пользователей связаны между собой следующим образом. Аутентификация данных основана на применении особого вида функций, называемых *хэш-функциями*, при построении которых используются криптографические преобразования. Аутентификация субъектов основана на реализации процедуры доказательства знания ключа шифра без его раскрытия, называемой также *доказательством с нулевой передачей знаний*. Именно использование шифров позволяет выделить аутентифицирующие, т. е. неотъемлемые вплоть до нарушения целостности, признаки электронных данных и субъектов, которые при соблюдении базовых требований идентичны аналогичным физическим и биометрическим признакам реальных объектов и субъектов.

1.1. Криптографическая аутентификация данных

Криптографическая аутентификация данных, как уже отмечалось, основана на использовании особого вида функций, называемых хэш-функциями. Общее характеристическое свойство класса хэш-функций (от англ. *hash* – «мелко рубить», «кромсать», «портить», т. е. «измельчать», «перемешивать») – переменная длина аргумента и фиксированная длина значения функции. Так как аутентифицируемые данные имеют различную и в общем случае неизвестную заранее длину, необходимость применения хэш-функции для их обработки является очевидной. При этом такая функция должна в обязательном порядке выделять аутентифицирующие признаки данных, для чего ей необходимо удовлетворять следующим основным требованиям:

1) для двух различных (хотя бы на 1 бит) данных должны формироваться различные хэш-значения;

2) восстановление по хэш-значению исходных данных (т. е. обратное преобразование) должно быть невозможным.

В литературе можно встретить более широкий (иногда и значительно) список дополнительных свойств аутентифицирующих хэш-функций, но при внимательном рассмотрении они либо являются следствием свойств 1 или 2, либо просто избыточны.

Возникает вопрос: если хэш-функция выделяет аутентифицирующие признаки данных, то какой реальный (и неотделимый) признак реального объекта она представляет? Помочь может аналогия с рисунком папиллярных линий – именно он является аутентифицирующим

признаком, отпечаток пальцев – его копией. В случае с электронными данными (т. е. последовательностью битов «1» и «0») аутентифицирующий признак – *двоичный рисунок* данных, а хэш-значение – «отпечаток» такого рисунка. Двоичный рисунок легко увидеть, представив двоичную последовательность в виде некоторой двоичной страницы. Так как изменение хотя бы одного бита этого рисунка по определению должно приводить к изменению значения хэш-функции, то он действительно является неотъемлемым – вплоть до нарушения целостности – признаком электронных данных или документа.

Максимальное количество аутентифицируемых данных очевидным образом связано с величиной хэш-значения и может быть вычислено по формуле $D = 2^m$, где m – размер хэш-значения в битах. Погрешность аутентификации зависит от возможности совпадения хэш-значений для различных входных данных и может быть нулевой только в случаях конечного множества входных данных. В остальных случаях она определяется в терминах стойкости хэш-функций к коллизиям, мерой которой является вычислительная сложность нахождения соответствующих совпадений. Выделяют два вида коллизий хэш-функций. Коллизией первого рода называется случай, когда для заданного сообщения можно подобрать другое с таким же хэш-значением (так называемый второй прообраз); коллизией второго рода называют случай, когда можно найти два сообщения (вообще говоря, произвольных) с одинаковыми хэш-значениями. Коллизия первого рода характеризует особенности построения хэш-функции. Коллизия второго рода связана с ограниченностью хэш-значения и неограниченностью входных данных. В этом случае на основании известного как «парадокс дней рождений» результата минимальным подмножеством множества значений хэш-функции, в котором с вероятностью больше одной второй найдутся совпадающие значения для различных входных векторов X_1 и X_2, является подмножество мощностью $2^{m/2}$. То есть для поиска такого совпадения потребуется в худшем случае просмотреть только половину, а не все пространство значений хэш-функции. Поэтому m-битная хэш-функция считается стойкой, если сложность нахождения коллизий для нее близка к $2^{m/2}$.

Размер хэш-значения m для наиболее известных алгоритмов составляет: MD5 –128 бит, SHA – 160 бит, ГОСТ Р 34.11–94 – 256 бит.

В основе методов построения хэш-функций, отвечающих требованиям 1 и 2, лежит схема гаммирования с обратной связью: $TS_i = TS_{i-1} \oplus TO_i$, где $TS_1 = G_1$; TO_i – блок открытого текста; TS_i – блок зашифрованного текста; G_1 – начальный ключ или вектор инициализации; G_i – гамма (ключ) шифра; $\oplus$ – операция сложения по модулю два (рис. 2).

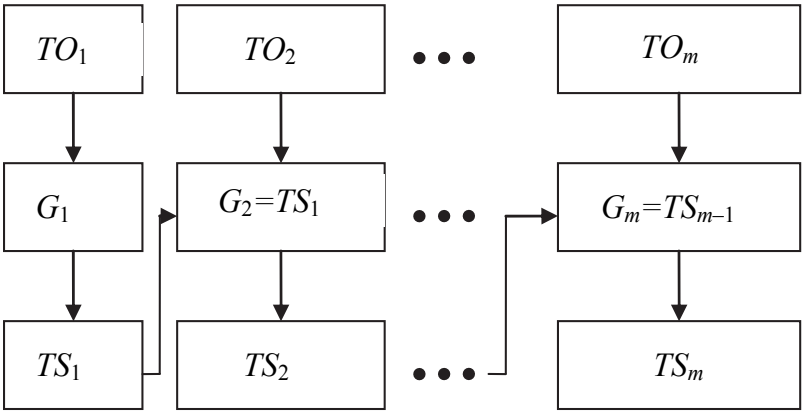

Рис. 2. Схема гаммирования с обратной связью

Если проанализировать рис. 2, то можно увидеть, что каждый бит второго блока функционально зависит от первого блока гаммы и *каждого бита* первых двух блоков открытого текста и т. д., т. е. *каждый бит последнего блока зависит от всех битов открытого текста.* Это позволяет проверить целостность исходного текста вплоть до изменения хотя бы одного бита путем сравнения значения текущего последнего блока с некоторым эталонным значением.

Как правило, данная схема модифицируется следующим образом: входная последовательность разбивается на блоки $(k - n)$ бит, которые объединяются со значением в n бит, полученным на предыдущем шаге в результате применения так называемой сжимающей функции (базового хэширования, например, гаммирования с обратной связью: для рис. 2 $k = mn$, где n – размер отдельного блока). Сжимающая функция преобразует получившийся в результате объединения блок размером в k бит в блок размером n бит. Начальное значение n_0 (соответствую-

щее TS_1 на рис. 2, называемое также и стартовым вектором, и вектором инициализации и т. п.) заполняется произвольным, известным всем способом, например нулем; последнее значение n_j после выполнения *j* итераций принимается в качестве значения хэш-функции.

Если исходные данные не кратны $(k - n)$ или в хэш-сумму необходимо добавить длину документа, обычно применяют два способа решения данных проблем. В первом случае в начало документа перед хэшированием добавляется поле фиксированной длины, в котором записывается длина текста. Затем объединенный блок данных дополняется нулями до ближайшего кратного $(k - n)$ размера. Во втором варианте документ дополняется справа одним единичным битом, а затем до кратного $(k - n)$ размера нулями. Необходимость в поле длины в данном случае отпадает – никакие два разных документа после выравнивания не станут одинаковыми. Возможны и многопроходные алгоритмы, когда входной блок неоднократно повторяется, а затем дополняется до ближайшей границы порции. Сжимающие функции могут называться хэш-функциями, функциями свертки и т. п. Значения этих функций могут называться хэш-значениями, хэш-суммами, свертками или просто – хэш.

Выделяют два типа криптографических хэш-функций использующие и не использующие ключи шифрования, т. е. шифры (в данном случае с симметричным ключом). Первые хэш-функции называют также *кодами аутентификации сообщений* (КАС) или *message authentication code* (MAC) [7]. Они позволяют одновременно аутентифицировать как источник данных, так и сами данные в системах с доверяющими друг другу пользователями. В этом случае стартовым вектором для таких функций является ключ шифра. Вторые хэш-функции называются также *кодами обнаружения ошибок* или *modification detection code* (MDC), или *manipulation detection code, message integrity code* (MIC) [7]. Такие хэш-функции могут применяться в системах как с доверяющими, так и не доверяющими друг другу пользователями, но для аутентификации данных с их помощью требуется привлечение дополнительных средств (например, шифрования, использования защищенного канала, цифровой подписи и т. п.). Стартовый вектор для таких функций является общим для всех пользователей.

Стойкость ключевых функций хэширования к коллизиям первого и второго рода характеризуется их *вычислительной устойчивостью.* Под

ней понимают высокую сложность подбора для заданного множества сообщений $\{x_1, \ldots, x_t\}$ (возможно пустого) с известными значениями сверток еще одного сообщения x ($x \neq x_i$, $i = 1, \ldots, t$) с правильным значением свертки (возможен случай $h(x) = h(x_i)$, $i \in \{1,\ldots, t\}$). При этом вычислительная сложность рассматривается здесь как вычислительная неразрешимость задачи, означающая, что ее решение с использованием вычислительной техники за реальное время невозможно.

Ключевые хэш-функции применяются в ситуациях, когда стороны доверяют друг другу и могут иметь общий секретный ключ. Обычно в этих условиях не требуется, чтобы система обеспечивала защиту в случае отказа получателя от факта получения сообщения или его подмены. Поэтому атаки на ключевые хэш-функции заключаются в имитации, т. е. передаче сфабрикованных сообщений в пустом канале, а также в подмене передаваемых сообщений с целью навязывания приемной стороне ложных сообщений [7].

При этом говорить о вычислительной устойчивости ключевых хэш-функций можно только в условиях неопределенности используемого ими ключа, так как знание ключа дает возможность вычислять значение свертки для любого набора данных. В то же время обратное утверждение неверно, так как подбор значения хэш-функции возможен в некоторых случаях и без предварительного определения ключа.

Для создания устойчивых хэш-функций могут использоваться симметричные криптографические алгоритмы. Олин из таких вариантов хэш-функции был предложен Уинтерницем (рис. 3) [7].

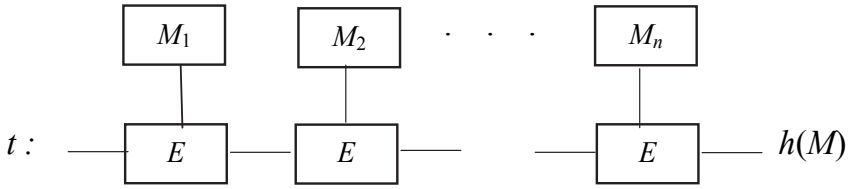

Рис. 3. Схема вычисления хэш-функции Уинтерница

Сообщение M разбивается на блоки M_i, по длине равные длине ключа, и на этих блоках, как на ключах, шифруется последовательно

некий фиксированный текст *t*. Пусть $E_{M_i}(t)$ – операция шифрования текста *t* на ключе M_i. Тогда $h(M) = E_{M_i}\left(E_{M_{i-1}}\ldots\left(E_{M_1}(t)\right)\ldots\right)$.

Поскольку шифратор обычно всегда проектируется стойким к анализу на основе открытого текста, которым здесь выступает *t*, то нахождение *M* для заданного *h*(*M*) оказывается вычислительно невозможным. Хэш-функция является стойкой в сильном смысле, если задача поиска частично эквивалентных ключей для нее сложна. Два ключа k_1, k_2 называются эквивалентными, если для всех текстов x $E_{k_1}(x) = E_{k_2}(x)$. Два ключа называются частично эквивалентными, если существует вычислимое непустое множество $\{x_i\}$, такое, что $E_{k_1}(x) = E_{k_2}(x)$ для всех $x \in \{x_i\}$. Например, частично эквивалентные ключи существуют для шифра DES, что ставит под сомнение стойкость хэш-функций, построенных на его основе. В этом случае криптографическая атака может быть такой. Подготавливаются два текста, различающиеся одним блоком. Первый текст содержит блок X_1, второй – блок X_2, причем эти блоки являются частично эквивалентными для текста, определенного предшествующими блоками. Затем первый текст отдается на подпись, а потом заменяется на второй, сохраняя при этом подпись [7].

Схема построения хэш-функций Уинтерница (рис. 3) может быть обобщена для всех симметричных блочных алгоритмов и шифрования с обратной связью (рис. 2) в виде схемы

$$h_i = E_A(B) \oplus C,$$

где три различные переменные *A, B, C* могут принимать одно из следующих четырех возможных значений: M_i, h_{i-1}, $M_i \oplus h_{i-1}$, константа. Всего существует $4^3 = 64$ варианта реализации общей схемы, но 52 из них являются слабыми или небезопасными. Остальные 12 вариантов безопасного хэширования приведены ниже:

1) $h_i = E_{h_{i-1}}(M_i) \oplus M_i$;

2) $h_i = E_{h_{i-1}}(M_i \oplus h_{i-1}) \oplus M_i \oplus h_{i-1}$;

3) $h_i = E_{h_{i-1}}(M_i \oplus h_{i-1}) \oplus M_i \oplus h_{i-1}$;

ОГЛАВЛЕНИЕ