Глава 3

Криптографические примитивы.

Лекция 8.

3.1 Симметричные и асимметричные методы шифрования.

Методы шифрования можно разделить на два типа: на *симметричные* и *асимметричные*.

При симметричном шифровании и для зашифровки, и для расшифровки сообщения применяется один и тот же ключ. Поэтому для безопасности передачи зашифрованного сообщения по открытым каналам необходимо предварительно обеим сторонам в тайне выбрать единый ключ, и лишь после этого начинать обмен информацией. При таком способе шифрования возникают очевидные трудности, связанные с необходимостью либо предварительной встречи, либо поиска надежного курьера, которому можно доверить секретный ключ. В частности, организовать надежный конфиденциальный канал связи для большого числа абонентов при таком способе шифрования довольно затруднительно.

Общая идея асимметричных методов шифрования заключается в том, что для зашифровки сообщения применяется один ключ, а для расшифровки — другой. Ключ для зашифровки открыто передается по любому каналу связи, в то время как ключ для расшифровки держится в секрете. Эти ключи, как правило, являются элементами некоторых алгебраических структур и связаны друг с другом некоторым соотношением. Но параметры этих структур и соотношение подбираются такими, чтобы время, требуемое для построения секретного ключа по открытому, было несравнимо больше того времени, которое нужно для построения открытого ключа по секретному.

Асимметричные методы шифрования обычно медленнее симметричных. Поэтому, если требуется передать большой объем информации, очень часто тело сообщения шифруют при помощи какого-нибудь симметричного алгоритма, а использующийся при этом ключ — при помощи асимметричного. Подобного рода схемы называются гибридными.

3.1.1 Один пример асимметричного алгоритма.

Первой работой, в которой была реализована идея асимметричного криптографического преобразования, является работа [**?**]. Авторы этой работы использовали тот факт, что задача дискретного логарифмирования в мультипликативной группе F_p^* поля F_p, где p — простое число, существенно сложнее задачи возведения в степень.

Отметим, что это обстоятельство можно очень легко использовать для защиты произвольного ресурса, будь то компьютер, или канал связи, от несанкционированного доступа. Для этого нужно выбрать достаточно большое простое число p и какую-нибудь образующую g группы F_p^*. Далее, каждый пользователь "придумывает" себе секретный пароль $t \in \mathbb{Z}$ и вычисляет элемент g^t поля F_p, который система и будет воспринимать как идентификатор данного пользователя. И каждый раз, когда пользователь решает войти в систему, он вводит свой секретный пароль t, система вычисляет g^t и сравнивает результат с идентификатором пользователя. Возможность взлома такой системы зависит от способности злоумышленника вычислять дискретный логарифм по основанию g от идентификатора пользователя, то есть выбор достаточно больших p может обеспечить необходимую безопасность. Отметим однако, что образующую g нужно выбирать не слишком маленькой, чтобы исключить возможность найти g перебором.

3.1.2 Алгоритм Диффи–Хеллмана обмена ключами.

Алгоритм, о котором сейчас пойдет речь, позволяет двум лицам, общающимся по незащищенному каналу связи, без предварительного обмена секретной информацией выработать общий ключ, который будет известен только им двоим. В дальнейшем он может быть использован в каком-либо симметричном алгоритме зашифрования и расшифрования информации.

Предположим, абонент A и абонент B выбрали простое число p и образующую g группы F_p^*. Случайным образом абонент A выбирает секретный ключ $a \in \mathbb{Z}$, $2 \leq a \leq p-2$, а абонент B — секретный ключ $b \in \mathbb{Z}$, $2 \leq b \leq p-2$.

Алгоритм 1 (Диффи, Хеллман)**.** *Данные: Простое число p, первообразный корень g по модулю p, секретный ключ a абонента A и секретный ключ b абонента B.*
Найти: Общий ключ $k \in F_p^*$, известный только абонентам A и B.
1. Абоненту A вычислить $x = g^a$ и переслать результат абоненту B.
2. Абоненту B вычислить $y = g^b$ и переслать результат абоненту A.
3. Абоненту A положить $k = y^a$,
4. Абоненту B положить $k = x^b$.

Корректность данного алгоритма следует из равенств $k_B = (g^a)^b = (g^b)^a = k_A$. Открытыми ключами здесь являются элементы g^a и g^b. Они передаются по незащищенному каналу связи, поэтому могут стать известными третьим лицам. Но если число p было выбрано достаточно большим, то для того, чтобы по известным g^a и g^b при помощи существующих на данный момент алгоритмов найти секретные числа a и b (и, соответственно, g^{ab}), потребуется очень много времени, что и обеспечивает требуемую безопасность. Аналогично, абоненты A и B не смогут за приемлемое время вычислить секретные ключи друг друга, то есть каждый из них при смене собеседника может не менять свой секретный ключ.

В описанном виде схема Диффи-Хеллмана уязвима относительно так называемой "атаки посередине" , в которой абонент - нарушитель C выступает в качестве посредника между A и B.

$$A \xleftarrow{z=g^c} \xrightarrow{x=g^a} C \xleftarrow{y=g^b} \xrightarrow{z=g^c} B$$

Если злоумышленнику удалось убедить абонента A воспринять его как абонента B, а абонента B — соответственно, как A, он, наладив с каждым из них связь, при помощи алгоритма Диффи–Хеллмана, создаёт секретный ключ g^{ac} для обмена информацией с A и секретный ключ g^{bc} для обмена информацией с B. В результате он получает полный контроль за перепиской A и B, включая возможность искажения и передачи ложной информации. Чтобы предотвратить подобную возможность, используют так называемые протоколы аутентификации.

Отметим также, что алгоритм 1 по своей структуре несколько отличается от изложенных ранее. Обычно предполагается, что у алгоритма ровно один исполнитель, и по этой причине команды имеют безадресный характер. Что же касается алгоритмов типа алгоритма 1, то они предполагают как минимум двоих исполнителей. Поэтому такие алгоритмы носят название *протоколов.*

3.2 Хеш-функции.

3.2.1 Хеш-функции и псевдослучайные последовательности.

Хеш-функции[1] используются, как составной элемент в различных криптографических приложениях. Эти функции отображают сколь угодно длинные сообщения в целые числа ограниченной величины. Значения хеш-функций называют хеш-значениями. Наиболее важные их применения связаны с подтверждением достоверности информации и со схемами цифровой подписи. При этом хеш-значения сообщений служат аналогами "отпечатков пальцев" человека. Люди имеют индивидуальные отпечатки пальцев. На этом строятся криминологические расследования. Иногда мы будем называть хеш-значения "отпечатками сообщений". Заметим, что по отпечаткам пальцев человека невозможно понять, какой у него цвет глаз, какой рост, каковы его вкусы. Нечто подобное имеет место и для отпечатков сообщений. По хеш-значению сообщения невозможно понять, что в нём содержится.

Перекодировав буквы числами, любое сообщение можно представить числом, вообще говоря большим целым числом. Каждое же целое число можно записать в двоичном виде, другими словами, записать в виде последовательности 0 и 1. Длины таких последовательностей могут быть сколь угодно велики. Множество всевозможных конечных последовательностей из 0 и 1 будем обозначать символом $\{0,1\}^*$. А для совокупности таких последовательностей, имеющих фиксированную длину $n, n \geq 1$, будем использовать обозначение $\{0,1\}^n$.

Рассмотрим отображение

$$H : \{0,1\}^* \mapsto \{0,1\}^n, n \in \mathbb{N}, \tag{3.1}$$

при каком-либо фиксированном натуральном n. Пример такого отображения - сумма цифр входного сообщения (аргумента H), она же - количество 1 в сообщении, взятая по модулю 2^n.

Отображение H называется *хеш-функцией, функцией хеширования* или *перемешивающей функцией*, если она имеет описываемые далее три свойства.

[1] Термин hash function имеет нетривиальный перевод на русский язык "хеш-функция". Английское слово hash — в словарях, не относящихся к компьютерам и информационным технологиям, переводится как перемешивать, резать, рубить, крошить, делать фарш, а если это слово - существительное, им обозначают рубленую смесь готового мяса и овощей, обычно запеченного или обжаренного. Мы будем иногда для словосочетания hash function использовать, как русский перевод, слова - "хеширующая функция"или "перемешивающая функция".

1. *Однонаправленность.* Значения H должны быть вычислимы с помощью некоторого алгоритма, причём этот алгоритм должен работать достаточно быстро. Конечно, время работы зависит от длины входа, будем считать эту зависимость полиномиальной. При этом вычисление значений обратной функции H^{-1} должно быть очень трудоёмким. Другими словами, по заданному числу y любое решение x уравнения $H(x) = y$ должно быть очень трудно вычислимым. Можно также сказать, что для любого выходящего сообщения функции H должно быть "очень трудно"вычислить какой-нибудь его прообраз (соответствующее входное сообщение).

2. *Отсутствие коллизий. Коллизией* для хеш-функции H называется любая пара конечных последовательностей x и y, состоящих из 0 и 1, для которых выполняется равенство $H(x) = H(y)$. Для любой хеш-функции множество коллизий бесконечно. Это утверждение легко доказать с помощью принципа ящиков Дирихле. Действительно, множество возможных последовательностей, которые могут быть поданы на вход хеш-функции бесконечно, а количество возможных выходов не превосходит 2^n. Количество входов больше количества выходов, поэтому должно быть несколько входов, отображающихся на один какой-нибудь выход. Более того, существует бесконечное количество входов, имеющих один и тот же конкретный выход. Таким образом, любая хеш-функция имеет бесконечное количество коллизий. Упомянутое выше свойство "отсутствие коллизий"у хеш-функции, означает лишь то, что задача нахождения хотя бы одной коллизии очень сложна в вычислительном отношении. Это - отсутствие коллизий до определённой поры.

3. *Сложность нахождения второго прообраза.* Это свойство можно выразить иными словами, сказав, что знание одного решения уравнения (системы уравнений) $H(x) = y$ не упрощает поиск ещё одного решения.

Для некоторых хеш-функций очень просто построить коллизии или какие-нибудь другие нарушения указанных трёх свойств. Например, если $H(x)$ есть сумма цифр в двоичной записи числа x, то $H(9) = H(17) = 2$ - коллизия. Но, вообще говоря, не существует метода, который для заданной хеш-функции доказывал бы, что поиск коллизий у неё требует очень большого времени или же, наоборот, легко находил существующие коллизии. Безопасность хеш-функции это всегда вопрос веры, основанной на многочисленных проверках и большом объёме компьютерных вычислений в попытках скомпрометировать её. Используемые на практике хеш-функции прошли сложные, иногда длящиеся годами, испытания, являются победителями или призёрами специальных конкурсов, и имеют собственные имена.

Как только для какой-нибудь хеш-функции найдены коллизия или какой-нибудь алгоритм, существенно снижающий трудоёмкость решения задач в пунктах 1) - 3), эта функция считается скомпрометированной и заменяется другой. Так было в сравнительно недавнее время с хеш-функциями, называвшимися MD5 и SHA-1.

В РФ используется функция хеширования, носящая имя "Стрибог". Она была введена в 2012 году соответствующим стандартом и заменила другую функцию, действовавшую в период с 1994 по 2011 годы. Длина выходных сообщений выбывшей функции равнялась 256 битов, но для неё в 2008 году был найден алгоритм обращения, см. п. 1), требующий 2^{225} операций. Это количество всё ещё велико для современных суперкомпьютеров, тем не менее эта функция считалась скомпрометированной и была заменена функцией "Стрибог".

В связи с биткойнами используется хеш-функция, называемая SHA-256, [2] все её значения могут быть записаны строками, состоящими из 256 нулей и единиц, т.е. являются целыми числами из промежутка $0 \leq x < 2^{256}$. Эта функция была разработана Агенством национальной безопасности США в 2002 году. В настоящее время для неё, как и любой другой работающей в серьёзных приложениях хеширующей функции, коллизии не известны. И не то, чтобы кто-то знал коллизию и скрывал её. Нет, вообще никто не знает коллизию для SHA-256, и за прошедшие годы никто не смог найти и предъявить её.

О том, как устроены и работают хеширующие функции, мы поговорим позже. Некоторое представление об этом дают слова, указанные в комментарии к переводу названия hash function. В дальнейшем мы узнаем больше и о том, какова роль таких функций в приложениях. Сейчас же мы покажем,

[2] SHA — сокращение от Secure Hashing Algorithm.

как используются хеширующие функции при построении псевдослучайных чисел.

Пусть $H(x)$ какая-нибудь принятая хеширующая функция, длину её выходов, как и ранее, будем обозначать буквой n. Следующий алгоритм строит псевдослучайную последовательность натуральных чисел x длины не более $\ell - 1$ битов, т.е. чисел, удовлетворяющих неравенствам $0 \le x < 2^{\ell-1}$.

Алгоритм 9. Данные: ℓ, d - *целые числа,* $\ell > 16$, $d > 0$.

Выход: *Псевдослучайная последовательность длины* d, *состоящая из чисел* x, $0 \le x < 2^{\ell-1}$.

1. Положить $I = \lceil \frac{\ell}{n} \rceil - 1 \ge 0$. [3], *а также*

$$a = \ell - 1 - I \cdot n$$

2. Выбрать произвольное натуральное число s.

3. Определить последовательность целых чисел z_k *равенствами*

$$z_0 = H(s), \qquad z_k = z_{k-1} + H(s+k) \cdot 2^{kn}, \quad 1 \le k \le I - 1.$$

4. Положить

$$x = z_{I-1} + (H(s+I) \pmod{2^a}) \cdot 2^{I \cdot n}.$$

При $I = 0$ *в предыдущей формуле присутствует только последнее слагаемое.*

5. Если количество построенных чисел x *меньше* d, *положить* $s = s + I + 2$ *и перейти в пункт 3. В противном случае завершить работу.*

В согласии с определением чисел I, a и свойствами функции $\lceil y \rceil$ имеем

$$\frac{\ell}{n} - 1 \le I < \frac{\ell}{n} \quad \text{и} \quad -1 < a \le n - 1.$$

Ясно что построенное число x неотрицательно. Следующее из пунктов 3 и 4 алгоритма равенство

$$x = \sum_{k=0}^{I-1} H(s+k) \cdot 2^{nk} + (H(s+I) \pmod{2^a}) \cdot 2^{I \cdot n}.$$

есть представление числа x в 2^n-ичной системе счисления, причём старшая "цифра"числа x не превосходит $2^a - 1$. Поэтому

$$x \le (2^n - 1) \sum_{k=0}^{I-1} 2^{nk} + (2^a - 1) \cdot 2^{nI} = 2^{a+nI} = 2^{\ell-1}.$$

[3] Символ $\lceil y \rceil$ обозначает наименьшее целое, превосходящее или равное действительному числу y. При любом $y > 0$ выполняется неравенство $\lceil y \rceil \ge 1$

Можно также сказать, что последовательность цифр 0 и 1 числа x в двоичной системе счисления, расположенная в обратном порядке есть конкатенация[4] векторов $H(s)\|H(s+1)\|\ldots\|H(s+I)$ с обрезанными старшими цифрами у вектора $H(s+I)$.

Для построения псевдослучайной последовательности чисел y на промежутке $2^{\ell-1} \le y < 2^{\ell}$ достаточно сдвинуть полученную выше последовательность чисел x на величину $2^{\ell-1}$, т.е. положить $y = 2^{\ell-1} + x$.

Ещё одна вариация на ту же тему. Предположим нужно построить псевдослучайную последовательность целых чисел t, принадлежащих промежутку $A \le t < B$. Положим для этого $I = \lceil \frac{\log(B-A)}{n} \rceil - 1$, где логарифм вычисляется по основанию 2 и определим x формулой

$$x = A + \left(\sum_{k=0}^{I} H(s+k) 2^{nk} \pmod{B-A} \right). \tag{3.2}$$

Тогда $A \le x < A + (B-A) < B$ и максимальное возможное значение суммы в (3.2) равно

$$S = \sum_{k=0}^{I} (2^n - 1) \cdot 2^{nk} = 2^{n(I+1)} - 1.$$

Так как

$$n(I+1) = n\lceil \frac{\log(B-A)}{n} \rceil \ge n \cdot \frac{\log(B-A)}{n} = \log(B-A),$$

то $2^{n(I+1)} - 1 \ge B - A - 1$.

Теперь мы обсудим алгоритм, также основанный на парадоксе дней рождения и позволяющий найти коллизию у любой хеш-функции $H(x)$. Он носит вероятностный характер. Обозначим буквой r количество всех различных значений функции $H(\overline{x})$, положим также $m = [\sqrt{2r}] + 1$. Не трудно проверить, что при $r \ge 6$ выполняется неравенство $m < r$. Выберем случайно различные аргументы $x_1, \ldots, x_m$ функции $H(\overline{x})$ и вычислим хеш-значения

$$H(x_1), \quad \ldots, \quad H(x_m).$$

Если среди этих значений есть одинаковые, то коллизия построена. При выбранном m справедливы неравенства $(m-1)^2 > \lceil \sqrt{2r} \rceil^2 > 2r$ и $\ln \mu < -1$, так что в противном случае доля последовательностей без одинаковых элементов

[4]Конкатенация $a_1\|a_2\|\ldots\|a_m$ векторов $a_1, a_2, \ldots, a_m$ есть вектор, полученный состыковкой этих векторов в единый длинный вектор $a_1 a_2 \ldots a_{m-1} a_m$.

не превосходит e^{-1}. Поэтому $\mu < e^{-1} = 0,367...$, и количество последовательностей с различными хеш-значениями существенно меньше половины всех последовательностей. Это доказывает, что если все значения хеш-функции равновероятны, то при случайном выборе набора аргументов $x_1, x_2, \ldots, x_M$ вероятность получить последовательность значений, имеющую равные элементы, превосходит $1 - e^{-1} > \frac{1}{2}$. Если все элементы выбранной последовательности хеш-значений оказались различными, можно выбрать вторую последовательность. Если опять не повезло, можно выбрать третью последовательность. Вероятность выбрать последовательность с равными элементами после k шагов не меньше $1 - e^{-k}$. С ростом k она очень быстро приближается к 1. К сожалению, этот гарантированно работающий алгоритм требует слишком много времени. Например, в случае $r = 2^{256}$ имеем $m = \lceil\sqrt{2r}\rceil + 1 > 2^{128}$. Последовательность из 2^{128} хеш-значений имеет огромную длину.

Конец восьмой лекции