Лекция 5

Электронные подписи

(Конспект: С. Федин)

5.1 Схемы электронных подписей

Электронная подпись гарантирует, что сообщения поступают от достоверного отправителя в неискаженном виде. Более того, получатель не только убеждается в достоверности сообщения, но и получает электронную подпись, которую в дальнейшем может использовать как доказательство достоверности сообщения третьим лицам в том случае, если отправитель попытается отказаться от своей подписи.

Определение 5.1. Схема электронной подписи — это тройка алгоритмов (G, S, V), где

- G — полиномиальный вероятностный алгоритм генерации ключей. На входе 1^k он выдает пару $(\mathtt{pri}, \mathtt{pub})$, где $\mathtt{pub}$ — открытый ключ схемы электронной подписи, а $\mathtt{pri}$ — соответствующий ему секретный ключ:
$$G(1^k) = (\mathtt{pri}, \mathtt{pub}).$$
- S — полиномиальный вероятностный подписывающий алгоритм. Ему на вход подается пара $(M, \mathtt{pri})$, где M — сообщение. Алгоритм S возвращает подпись для этого сообщения:
$$S(M, \mathtt{pri}) = signature.$$
- V — полиномиальный вероятностный проверяющий алгоритм; $V(M, s, \mathtt{pub}) = 1$ в случае, если подпись s — правильная (сгенерированная алгоритмом S с соответствующим ключом $\mathtt{pri}$) подпись для сообщения M; $V(M, s, \mathtt{pub}) = 0$ в противном случае.

Замечание 5.1. Использование случайных битов является полезным (и даже необходимым) несмотря на отсутствие возможности “ошибки”; однако и ее можно ввести, ослабя определение ($\Pr\{V(M, S(M, \mathtt{pri}), \mathtt{pub}) \neq 1\} \leq 2^{-k}$, где `pri` и `pub` — соответствующие друг другу ключи, сгенерированные $G(1^k)$).

Пример 5.1. Рассмотрим схему электронной подписи на базе RSA. Пусть N — произведение двух больших простых чисел. В качестве публичного ключа возьмем пару (N, e), где $(e, \phi(N)) = 1$, а в качестве приватного возьмем d, такое, что $ed \equiv 1 \pmod{\phi(N)}$. Подписью для сообщение M будет $M^d \bmod N$. Любой желающий может проверить подлинность подписи. Для этого достаточно проверить равенство $M = (M^d)^e \pmod N$.

Заметим, что такая схема электронной подписи является ненадежной (определения надежности мы дадим ниже), так как, например, если противник научился подделывать подпись для двух сообщений, то он сможет подделать подпись и для произведения этих сообщений: такая подпись будет просто произведением подписей для каждого из этих сообщений.

5.2 Надежность схем электронных подписей

Для определения надежности схемы электронной подписи необходимо сделать некоторые предположения о противнике. Проведем классификацию возможных противников. Отметим, что все алгоритмы G, S, V являются общедоступными, поэтому любой противник знает описание схемы подписи.

- Самый слабый из всех возможных противников может подделывать подписи, зная только публичный ключ `pub` схемы.

- Более сильный противник может пользоваться некоторой дополнительной информацей о схеме подписи: он может смотреть на некоторые подписанные сообщения — на пару (сообщение, подпись). При этом он никак не влияет на выбор этих сообщений.

- Самый же сильный противник может сам выбирать сообщения и получать для них правильные подписи, то есть может посылать выбранные им самим сообщения (конечно, отличные от того, подпись под которым он собирается подделать) на подпись подписывающему. Выбор сообщения может зависеть от предыдущих полученных подписей.

Кроме классификации возможных противников, необходимо определить угрозу безопастности схемы подписи, против которой мы хотим защищаться.

- Самая сильная угроза — это вычисление противником секретного ключа `pri`. Если противник может реализовать такую угрозу, то схема подписи является ненадежной.
- Более слабая угроза: возможность противника подделывать подпись под любым сообщением.
- Наконец, самая слабая угроза — это возможность противника подделать подпись под некоторым сообщением (выбранным им самим — или, еще один вариант, быть может, он даже не может сам сгенерировать это сообщение).

Понятно, что наиболее надежными являются схемы, надежные относительно самых слабых угроз при самом сильном противнике. Именно этот случай мы и будем рассматривать.

Важный момент: Надежность схемы рассматривается для *одного и того же* ключа для всех запросов на подпись. Дело в том, что передача ключа требует альтернативных методов аутентификации (ведь ключ не подписан, и также может быть подделан). Поэтому повторная передача ключа является крайне дорогой операцией.

5.3 Пример «надежной» схемы

Рассмотрим одну из возможных схем электронной подписи. Заметим, что достаточно научиться подписывать сообщения длины k. Рассмотрим семейство перестановок $\{f_\alpha\}_\alpha$ «с секретом», сохраняющих длину. Алгоритм, генерирующий ключи, в качестве приватного ключа берет «секрет», т.е. $f_{\alpha_1}^{-1}$, где $|\alpha_1| = k$. В качестве публичного ключа алгоритм берет f_{α_1} и некоторые случайные строки $a_j^0, a_j^1 \in \{0,1\}^k$, где $j \in 1, \dots, k$; $b_j^0, b_j^1 \in \{0,1\}^k$, где $j \in 1, \dots, p(k)$. Здесь $p(k)$ — длина битовой записи f_α при $|\alpha| = k$.

Теперь рассмотрим первый шаг (кодирование первого сообщения) подписывающего алгоритма. Чтобы закодировать i-ый бит сообщения $M = m_1 \dots m_k$, подписывающий алгоритм выбирает a_i^0, либо a_i^1, в зависимости от того, $m_i = 0$ или $m_i = 1$, соответственно, и применяет обратную перестановку, получая $f_{\alpha_1}^{-1}(a_i^{m_i})$. Таким образом, сообщение M кодируется так: $(f_{\alpha_1}^{-1}(a_1^{m_1}), \dots, f_{\alpha_1}^{-1}(a_k^{m_k}))$.

Отметим, что применять такой способ кодирования при подписывании дальнейших сообщений — не очень хорошо: в этом случае противник сможет найти $f_{\alpha_1}^{-1}$. Поэтому для каждого нового сообщения надо выбирать новую перестановку f_{α_j} (такую, что мы знаем $f_{\alpha_j}^{-1}$). Для того, чтобы закодировать i-ый бит j-ого сообщения, мы вычислим $f_{\alpha_j}^{-1}(a_i^{m_i})$, а к подписи всего сообщения добавим $\{f_{\alpha_{j-1}}^{-1}(b_i^{f_{\alpha_j,i}})\}_i$, где $f_{\alpha_j,i}$ обозначает i-ый бит представления этой перестановки; тем самым, публичный ключ f_{α_1} позволяет извлечь f_{α_j} без аутентификации посторонними средствами. Таким образом, подпись j-ого сообщения должна содержать

$$((f_{\alpha_j}^{-1}(a_1^{m_1}),\ldots,f_{\alpha_j}^{-1}(a_k^{m_k}));\ (f_{\alpha_{j-1}}^{-1}(b_1^{f_{\alpha_j,1}}),\ldots,f_{\alpha_{j-1}}^{-1}(b_{p(k)}^{f_{\alpha_j,k}}))).$$

На самом деле, в подпись надо включить не только приведенные выше строки, но и коды всех предыдущих перестановок, то есть в каждую следующую подпись включаем историю подписей предыдущих сообщений.

Такой способ не очень эффективен, так как из-за того, что мы храним историю, подписи становятся с каждым новым сообщением все длиннее.

Утверждение 5.1. *Построенная выше схема электронной подписи надежна.*

Доказательство. Предположим противное: пусть построенная схема ненадежна. Так как мы рассматриваем самого сильного противника (обозначим его A), то считаем, что он может попросить подписывающий алгоритм подписать некоторое полиномиальное число $F(k)$ сообщений по своему выбору. После этого с вероятностью $\frac{1}{Q(k)}$ он подписывает сообщение, которое еще не спрашивал. Покажем, что в этом случае существует алгоритм B, получающий на вход перестановку h «с секретом», для которой ему неизвестна обратная, и $y \in \{0,1\}^k$, и возвращающий $h^{-1}(y)$ с вероятностью $\frac{1}{\mathrm{poly}(k)}$.

Построим этот алгоритм. Он будет пытаться использовать h как очередную перестановку для подписывания сообщения, которое запрашивает A, а y в качестве a_i или b_i. Так как B не знает h^{-1}, то мы будем поступать следующим образом. Сгенерируем случайно $x_j^0, x_j^1, z_j^0, z_j^1$, где $j \in \{1,\ldots,k\}$. Положим $a_j^i = h(x_j^i)$; $b_j^i = h(z_j^i)$, для $i \in \{0,1\}, j \in \{0,1\}^k$. (Благодаря выбору a и b мы сможем подписывать сообщение перестановкой h, не зная h^{-1}.) Выберем j_0 случайно из $\{1,\ldots F(k)\}$. Алгоритм B будет работать почти также как и подписывающий алгоритм, за тем исключением, что когда ему надо будет выбрать в качестве очередной перестановки $f_{\alpha_{j_0}}$, он выберет h.

Далее, случайно выберем одну из строчек a_j^i или b_j^i и заменим ее на y (теперь если нам придется для подписывания применять h^{-1} к тому биту, который мы закодировали точкой y, нас ждет неудача — но вероятность этого мала). Запустим алгоритм A на полученном открытом ключе. Заметим, что если A запрашивает подпись для сообщения M_{j_0}, то B правильно подпишет это сообщение с вероятностью $\frac{1}{2}$ — с этой вероятностью он может отгадать, где находится y.

Для некоторого сообщения M алгоритм A с вероятностью $\frac{1}{Q(k)}$ подделает правильную подпись s. Эта подпись должна отличаться от тех подписей, которые давал ему алгоритм B, при запросах A. Вероятность того, что она отличается на α_{j_0}-ом запросе — хотя бы $\frac{1}{F(k)}$. Кроме того, с вероятностью хотя бы $\frac{1}{F(k)}$ алгоритм F спросит хотя бы α_{j_0} запросов у B. Заметим также, что с вероятностью $\frac{1}{2(k+p(k))}$ алгоритм сможет правильно инвертировать y.

Итак, вероятность успеха алгоритма B составляет не менее $\frac{1}{4(k+p(k))F^2(k)} = \frac{1}{\mathrm{poly}(k)}$, что и требовалось для получения противоречия.

$\square$

5.4 Практическая схема

Приведем другую схему подписи, ***PSS0***: улучшим, используя хэш-функции, схему подписи на базе RSA, приведенную в начале лекции. Грубо говоря, семейство криптографических хэш-функций $H_i : \{0,1\}^* \to D_i \subseteq \{0,1\}^{|i|}$ (для всех $i \in \{0,1\}^*$) состоит из легко вычислимых функций, для которых “трудно” найти пару строчек $x, y, x \neq y$, таких, что $H(x) = H(y)$. Здесь D_i — это область определения RSA с соответствующим параметром надежности (в дальнейшем i опускается).

Итак, опишем схему подписи.

Алгоритм, генерирующий ключи, генерирует (e, N) в качестве `pub` и (d, N) в качестве `pri`. N, d, e — это параметры RSA, уже описанные в начале лекции.

Для того, чтобы подписать сообщение M, подписывающий алгоритм вычисляет $y = H(M \circ r)$, где $r \in \{0,1\}^s$ — случайная строчка, $\circ$ — операция конкатенации строк, $s = |r|$ — еще один параметр надежности системы. Потом, вычислив $x = y^d \bmod N$, использует в качестве подписи пару (r, x).

Проверяющий алгоритм легко может проверить подлинность подписи. Для этого ему достаточно, вычислив $y = H(M \circ r)$, проверить равенство $x^e \equiv y (\mathrm{mod}\ N)$.

Докажем, что такую схему подписи трудно подделать, в предположении, что RSA трудно взломать. А именно, докажем следующее утверждение:

Утверждение 5.2. *Пусть, P_{RSA} — вероятность взломать RSA с параметром надежности k, а P_{PSS0} — вероятность взломать схему PSS0 с параметрами надежности k, s. Тогда*

$$P_{PSS0} \leq P_{RSA} + \frac{(q_{hash} - 1)q_{sign}}{2^s},$$

где q_{hash} — количество обращений, которое делает соперник к хэш-функции, а q_{sign} — количество запросов, которое соперник посылает нам за подписью.

Доказательство. Путь A — некоторый противник. Построим алгоритм B, взламывающий RSA с помощью A. Нам понадобится четыре вспомогательных массива, каждый длиной q_{hash} и две процедуры $HASH(M \circ r)$ и $SIGN(M)$.

- $V[j]$ — j-ый запрос к хэш-функции: (M_j, r_j).
- $R[j]$ — строчка r_j.
- $Y[j]$ и $X[j]$ — массивы, назначание которых будет видно из приведенных ниже процедур.

Опишем процедуру $HASH(M \circ r)$. Сначала она сохраняет запрос и строчку в массивы:

$$V[j] := M \circ r,$$
$$R[j] := r.$$

Далее пробегает по V и проверяет, был ли уже такой запрос. Если такого запроса не было, то полагаем

$$X[j] := x_0,$$

где x_0 — случайно выбирается из D.

$$Y[j] := y(X[j])^e \pmod N$$

В противном случае, если такой запрос был (пусть его номер j_0), присваиваем

$$X[j] := X[j_0],$$
$$Y[j] := Y[j_0].$$

В итоге функция $HASH(M \circ r)$ возвращает $Y[j]$.

Опишем процедуру $SIGN(M)$. Сначала выберем случайно строку $r \in \{0,1\}^s$. Пробежим по R и посмотрим было ли уже выбрано такое r. Если такое r уже было выбрано, выходим из процедуры — нам не повезло. В противном случае присваиваем

$$V[j] := M \circ r,$$

$$R[j] := r,$$

$$X[j] := x_0,$$

где x_0 — случайно выбирается из D.

$$Y[j] := (X[j])^e \bmod N.$$

Возвращаем $X[j]$.

Тепреь опишем собственно сам алгоритм B. Его задача — на входе (y, e, N) вернуть $y^d \bmod N$. Запустим алгоритм A на входе (N, e). Алгоритм вернет сообщение M и его подпись (r, x). При этом, если он будет спрашивать подпись какого-либо сообщения или значение хэш-функции, то будем возвращать ему соответственно результаты процедур $SIGN$ и $HASH$. Положим

$$y := H(M \circ r)$$

Пробежим по V и найдем номер j_0 такой, что $V[j_0] = (M, r)$. Вернем $xX[j_0]^{-1} \pmod N$.

Посчитсаем вероятность успеха алгоритма B. Оценим вероятность того, что нам не повезло в процедуре $SIGN$. Нам не повезло — это значит, что мы выбрали на j-ом шаге в качестве очередного r такое r_0, что $\exists j_0 \in \{1, \dots j\} : \ R[j_0] = r_0$. Так как мощность $\{1, \dots j\}$ не превосходит $q_{hash} - 1$, то вероятность неудачи оценивается как $\frac{q_{hash}-1}{2^s}$. Процедура $SIGN$ потерпит неудачу не более q_{sign} раз, значит итоговая вероятность неудачи алгоритма B равна $\frac{(q_{hash}-1)q_{sign}}{2^s}$. Итак,

$$P_{\text{PSS0}} \le P_{\text{RSA}} + \frac{(q_{hash}-1)q_{sign}}{2^s},$$

что и требовалось.

$\square$