Лекция 6

Цифровые подписи

(Конспект: В. Вальтман)

Частично использован также конспект С. Федина 2005 года.

6.1 Определения

Алиса посылает Бобу сообщения. При этом она хочет их посылать так, чтобы Боб мог удостовериться, что их посылает именно она. А злой Чарли хочет подделать письмо от Алисы. То есть послать собственное письмо так, чтобы Боб не смог понять, что это послала не Алиса.

Для этого и используется цифровая подпись. У неё есть общедоступный ключ v и секретный ключ s. Чарли может давать Алисе на подпись разные документы. При этом он хочет послать от её имени какой-то другой документ (в том смысле, что он не может дать его её на подпись). Мы хотим, чтобы у него это почти не могло получиться.

Как обычно, у нас есть параметр надежности n. В дальнейшем мы опускаем параметры 1^n и $1^{\text{длина сообщения}}$, которые считаются публичными и всякий, кто желает, может получить их на вход.

Итак, более точные определения.

Определение 6.1 (схема цифровых подписей, digital signatures scheme, DSS). Пусть G, S, V — детерминированные полиномиальные по времени машины Тьюринга, такие что:

- $G\colon (1^n, r_g) \mapsto (s, v)$,
- $S_s\colon (\alpha, r_s) \mapsto \Delta$ (здесь α — наше сообщение, а Δ — подпись),
- $V_v\colon (\alpha, \Delta, r_v) \mapsto 0$ или 1,

- $\forall \alpha \Pr_{r_g, r_s, r_v}\{V_v(\alpha, S_s(\alpha, r_s), r_v) = 0\} = 0$.

Упражнение 6.1. Покажите, что в этой схеме можно считать подписывание и проверку детерминированными, т.е. $r_v = r_s = \lambda$ (пустая строка).

В дальнейшем будем игнорировать наличие r_v и r_s.

Определение 6.2 (надежная цифровая подпись). Эта схема называется надежной, если

$$\forall A^{S_S}\ \forall p\ \Pr\{A^{S_s}(v) = (\alpha, \Delta) : V_v(\alpha, \Delta) = 1\} < \frac{1}{p(n)}.$$

Здесь p — полином, A — полиномиальная вероятностная оракульная машина Тьюринга, и ей запрещено запрашивать оракул S_s о строке α.

Определение 6.3 (одноразовая надежная цифровая подпись). Схема называется одноразовой, если к оракулу A разрешено обращаться только один раз.

Определение 6.4 ($l(n)$-ограниченная цифровая подпись, $l(n)$-DSS). Если все вышеперечисленное верно для строк длины $l(n)$ (в частности, S предназначен только для подписывания таких строк, а $A^{\cdot}$ может задавать вопросы оракулу только про такие строки), то схема называется $l(n)$-ограниченной (или $l(n)$-ограниченной надежной, если требование такой надежности соблюдено).

6.2 Дальнейшие планы

Мы попытаемся доказать следующий набор фактов (везде схемы предполагаются надежными).

1. По owf можно построить одноразовую ограниченную DSS.

2. Применив хеш-функцию к сообщению, получим из ограниченной — «неограниченную» DSS (однако, если DSS была одноразовой, одноразовой она и останется).

3. Потом получим, что если есть одноразовая DSS, которая позволяет кодировать сообщения, которые хотя бы в два раза длиннее самих ключей, то есть и многоразовая DSS.

6.3 Одноразовая ограниченная схема

Теорема 6.1. *Пусть f — owf, l — полином, $m = l(n)$. Положим*

$$G(1^n) = ((u_1^0, u_1^1, \ldots, u_m^0, u_m^1), (v_1^0, v_1^1, \ldots, v_m^0, v_m^1)),$$

где $u_k^i \in \{0,1\}^n$ выбраны случайным образом, $v_k^i = f(u_k^i)$. Возьмём

$$S(\alpha_1 \ldots \alpha_m) = (u_1^{\alpha_1}, \ldots, u_m^{\alpha_m});$$

V достаточно проверить, что $\forall i\ f(u_i^{\alpha_i}) = v_i^{\alpha_i}$. Полученная конструкция является надёжной одноразовой $l(n)$-DSS.

Доказательство. Предположим, что эта DSS взламывается с вероятностью ε. Для взлома неминуемо необходимо обратить f хотя бы для одной из строк v_k^i, поскольку подписанное сообщение должно хотя бы в одном бите отличаться от запроса к оракулу. Чтобы обратить owf в нужной нам точке y, сгенерируем ключ и заменим в нем одно из v_k^i на y (k и i выберем случайным образом); промоделируем старого противника. *Если* при обращении к оракулу противник не спросит нас $f^{-1}(v_k^i)$ (вероятность этого — $\frac{1}{2}$), то он попытается вычислить нам $f^{-1}(y)$ с вероятностью $\frac{1}{m}$ (это — вероятность, что мы угадали то место, в котором будут отличаться запрос к оракулу и подписываемое сообщение) и сделает это с вероятностью ε. Итого мы обратим f с вероятностью $\geq \frac{\varepsilon}{2m}$. □

Упражнение 6.2. Формально записать все эти вероятностные рассуждения, учитывая потенциальную возможность зависимости между событиями.

6.4 Одноразовая неограниченная схема

Определение 6.5 (collision-free hashing function family, cfhff). Семейство хеш-функций без коллизий — это полиномиальные детерминированные алгоритмы σ и H и семейство функций $\{h_\zeta\}_\zeta$, такие что

- $\sigma\colon (1^n, r_\sigma) \mapsto \zeta$,
- $h_\zeta\colon \{0,1\}^* \to \{0,1\}^{l(|\zeta|)}$,
- $H(\zeta, x) = h_\zeta(x)$,
- $\forall A\ \forall p\ \Pr\{A(\zeta) = (y, y') : h_\zeta(y) = h_\zeta(y'),\ y \neq y'\} < \frac{1}{p(n)}$,

где A и p — как обычно.

Упражнение 6.3. Показать, что существование cfhff влечёт существование owf.

Теорема 6.2. *Пусть у нас имеются надежная одноразовая ограниченная DSS и collision-free hashing function family (с одним и тем же полиномом l из их определений). Будем подписывать и проверять не само сообщение (теперь произвольной длины), а хеш-функцию от него; ее номер ζ (сгенерированный случайным образом) включим в s и в v. Полученная одноразовая DSS будет по-прежнему надёжной.*

Доказательство. Взлом новой схемы означает, что мы либо найдём коллизию, либо взломаем старую схему подписей. □

Упражнение 6.4. Дать формальное доказательство.

6.5 Многоразовая схема на основе tdpf (сохраняющих длину)

Вообще-то можно обойтись и cfhff, но для начала приведём схему, основанную на tdpf, причем Алисе придется хранить и передавать всю историю использования ключа (так что длина подписи будет расти). Далее — цитата из старого конспекта.

Заметим, что достаточно научиться подписывать последовательности сообщений 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 6.1. *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.

$\square$

6.6 Многоразовая схема на основе cfhff