テーマ２：公開鍵暗号と署名の安全性

本日の講義の目的

- 公開鍵署名の安全性をランダムオラクルの枠組みで示す

4.1. 目標とする定理

公開鍵暗号系を一つ考える．つまり，その暗号系で用いられる落とし戸関数群とその生成器 G を一つ仮定する．我々はこの公開鍵暗号系を使って公開鍵署名を実現する方法を提案し，その安全性を議論したい．つまり，この公開鍵暗号系を使って，署名するためのアルゴリズム Sign と，署名を検証するアルゴリズム Verify を設計し，それらが安全性の条件を満たしていることを示したい．ただし，公開鍵暗号系自身の安全性が保証されなければ署名は当然無理なので，ここでは公開鍵暗号系の安全性は成り立つものと仮定して議論する．

以上の目標を，以前に準備した記号を使い明確に記述することから始めよう．

以下では k をセキュリティパラメータの 1 つの値，(f, f^{-1}) を $\mathtt{G}(k)$ の出した公開鍵と暗号鍵とする．つまり，任意の $x \in \{0,1\}^k$ に対して

$$f^{-1}(f(x)) \;=\; x$$

が成り立つ．また f, f^{-1} は多項式時間アルゴリズムである．それに対し，仮定とする公開鍵暗号系自身の安全性は，f だけでは逆関数の計算が困難である，という性質だった．それをキチンと書くと次のようになる．

$$\begin{aligned}
&\forall \text{多項式時間アルゴリズム } \mathtt{A},\ \forall c > 0 \\
&\exists k_c,\ \forall k \geq k_c \\
&\qquad \Pr\left[\mathtt{A}(f,y) = x \,\middle|\, (f,f^{-1}) \leftarrow \mathtt{G}(k), x \leftarrow \{0,1\}^n\right] \;<\; k^{-c}.
\end{aligned} \tag{1}$$

このような公開鍵暗号系を用いた署名の方法だが，我々は次のような手法を考える．

$$\begin{aligned}
\mathtt{Sign}(f^{-1}, x) &= f^{-1}(H(x)) \\
\mathtt{Verify}(f, x, \sigma) &= \begin{cases} \mathrm{OK}, & \text{if } f(\sigma) = H(x), \\ \mathrm{NG}, & \text{otherwise.} \end{cases}
\end{aligned}$$

ただし，ここで H は，誰もが簡単に計算できるランダムハッシュ関数とする．つまり，$\{0,1\}^k$ から $\{0,1\}^k$ へ対応を（少なくとも見た目は）ランダムに決める関数である．

我々の目標は，この署名の対偽造安全性を示すことである．以前の記号を使うと次の性質である．

$\forall$ 多項式時間アルゴリズム $\mathtt{F}$, $\forall c > 0$

$$\exists k_c,\ \forall k \geq k_c \tag{2}$$
$$\Pr\left[\mathtt{Verify}(f,x,\sigma)=1 \,\middle|\, (f,f^{-1}) \leftarrow \mathtt{G}(k), (x,\sigma) \leftarrow \mathtt{F}^{\mathtt{Sign}(f^{-1},\cdot)}(k)\right] \ < \ k^{-c}.$$

我々はハッシュ関数 H が，たとえば次のように，ランダムオラクル R を使って定義されている場合を考える．そのもとでは，(1) $\Rightarrow$ (2) が証明できるのである．

$$\text{任意の } x \in \{0,1\}^k \text{ に対して } H(x) \ = \ R(x0)R(x00)\cdots R(x0^k).$$

4.2. 証明のアイデア

対偶を示す．つまり，$\neg$ (2) $\Rightarrow$ $\neg$ (1) を示す．そのために，ある多項式時間偽造生成アルゴリズム $\mathtt{F}_0$ が存在して，またある定数 $c_0 > 0$ が存在して，セキュリティパラメータ k をどんなに大きくとっても

$$\Pr\left[\mathtt{Verify}(f,x,\sigma)=1 \,\middle|\, (f,f^{-1}) \leftarrow \mathtt{G}(k), (x,\sigma) \leftarrow \mathtt{F}_0^{\mathtt{Sign}(f^{-1},\cdot)}(k)\right] \ \geq \ k^{-c_0} \tag{3}$$

となると仮定する．

その仮定のもと $\neg$ (1) を示すには，要するに暗号を破る多項式時間アルゴリズム $\mathtt{A}_0$ を作って見せればよい．その計算で仮定している $\mathtt{F}_0$ をうまく使うのである．

まず，アルゴリズム $\mathtt{F}_0$ だが，議論を簡単にするために，その実行 $\mathtt{F}_0(k)$ に関して次のように仮定しよう．

(a) 実行中にランダムハッシュ関数 H の計算を，ちょうど $n(k)$ 回だけ行う．ここでは，$u_1, ..., u_{n(k)}$ に対して，H の計算が行われるものとする．

(b) 実行中に $\mathtt{Sign}(f^{-1}, u)$ の計算を頼む場合には，$\mathtt{F}_0$ はランダムハッシュ関数 H を使って，$H(u)$ の値を必ず計算している．つまり，$u \in \{u_1, ..., u_{n(k)}\}$ である．

アルゴリズム $\mathtt{A}_0(y)$ の実行：(ただし y $(= f(x))$ を想定)

1. $\mathtt{F}_0(k)$ が使うランダムビットをランダムに用意する．また，$t \in \{1, ..., n(k)\}$ をランダムに 1 つ決めておく．
2. このランダムビットと t を使って，$\mathtt{F}_0(k)$ の計算を以下のようにシミュレートする．
   - $\mathtt{F}_0(k)$ が $H(u_i)$ を使う度に $r_i \in \{0,1\}^k$ をランダムに選び，$y_i = f(r_i)$ を $H(u_i)$ として返す．ただし，$i = t$ の場合のみ，y を返す．このインチキなランダムハッシュ関数の値を $H'(u_i)$ と書くことにする．
   - $\mathtt{F}_0(k)$ が $\mathtt{Sign}(f^{-1}, u)$ の計算を依頼したとする．このとき，仮定 (b) から，何らかの i, $1 \leq i \leq n(k)$, に対し，$u = u_i$ のはずである．もし $i = t$ ならば「失敗」として諦める．そうでない場合には，r_i を署名 $(= f^{-1}(H'(u_i)))$ として返す．
3. シミュレーションが無事終了して，$\mathtt{F}_0(k)$ が (m, σ) を答えとして出力したとする．もし $m \neq m_t$ ならば「失敗」として諦める．一方，もし $m = m_t$ ならば，σ を $f^{-1}(y)$ として出力する．

このアルゴリズムにより y の逆元 $x = f^{-1}(y)$ が求まる確率が

$$\Pr\left[\, \mathtt{A}_0(y) = x \,\middle|\, (f, f^{-1}) \leftarrow \mathtt{G}(k), x \leftarrow \{0,1\}^k, y = f(x) \,\right] \;\geq\; (k^{-c_0} - 2^{-k})/n(k) \qquad (4)$$

であることを示す.

鍵はランダムハッシュ関数の性質から導かれる次の 2 点である. (i) アルゴリズム $\mathtt{F}_0$ の性能は, ランダムハッシュ関数が H であろうが, H' であろうが, まったく変わらない. 具体的には $\mathtt{F}_0$ の成功確率が同じである.（ここでは $x \in \{0,1\}^k$ が一様ランダムで選らばれていることも重要.）(ii) $m \notin \{H'(u_1), ..., H'(u_{n(k)})\}$ でない限り, $\mathtt{F}_0$ が成功する確率は 2^{-k} 以下である.

その上で, もし $m = m_t$ で, $\mathtt{F}_0$ が成功している場合には,

$$\sigma = f^{-1}(H'(m_t)) = f^{-1}(y)$$

となることが重要である.

以上の点を確認していけば (4) を証明することができる.