電子署名入門

情報セキュリティ大学院大学　有田正剛

平成 19 年 5 月 11 日

目 次

1　はじめに

電子署名は現代の情報社会を支える基盤技術である。我たちは、電子データの真正性が要求されるたびに、それとは意識せずに日々色々な場面で電子署名を使っている。Internet では SSL、高速道路では ETC、また市役所では住民基本台帳カードが私たちの代理として署名演算を実行している。

この入門講義では、そのような電子署名とはどんな技術か、そのからくりはどうなっているか、またなぜ安全といえるか、解説したい。

電子署名はメッセージ（対象とする電子データ）の真正性を、誰もが検証できる形で、保証する電子データである。署名を作れるのは本人のみだが、検証はだれでもできるという状況を実現する。そのような電子署名を

つくる上で、数学の一分野である整数論がとても役に立つ。ここでは例として RSA 署名を取り上げ、それが整数のべき乗剰余演算を利用して作られることをみる。そして、私たちや私たちが作り出したコンピュータにとってべき乗剰余演算の逆関数を計算することが困難であるために、RSA 署名が署名として成立する（すなわち、与えられたメッセージに対してその署名を偽造できない）のをみる。

しかし、電子署名をさらに「存在的偽造不可」なものにするには、整数論だけでは足りない。ハッシュ関数と呼ばれるある種の暗号学的な圧縮関数を用いて強化する必要がある。そのような工夫により RSA 署名を強化した例として RSA-FDH 署名を紹介する。そして、RSA-FDH 署名が、ランダムオラクルモデルと呼ばれるある種のヒューリスティックスのもとで、選択メッセージ攻撃に対し存在的偽造不可であること（すなわち、敵がいくつかのメッセージの署名を入手したとしても、それら以外のどのようなメッセージに対してもその署名を偽造できないこと）の証明をスケッチする。

2　電子署名の基本機能

電子署名はメッセージ（対象とする電子データ）の真正性を、誰もが検証できる形で、保証する電子データである。署名を作れるのは本人のみだが、検証はだれでもできるという状況を実現する。

電子署名では各自は自分用の署名鍵と検証鍵の組をもつ。署名鍵は秘密に保管し、検証鍵は公開する。署名鍵はメッセージの署名を生成するのに用い、検証鍵は署名を検証するに用いる。ある署名鍵で生成された署名は、それとセットになっている検証鍵でしか、受理されないように作られる。

電子署名を用いて、実際にメッセージに署名を添付して送るには、以下のようにする。送信者 Alice は自分の検証鍵 pk_A を公開しておく。Alice は、自身の署名鍵 sk_A を用いてメッセージ m の署名 σ を生成し、メッセージ m とともに受信者 Bob に送る。メッセージと署名の対 (m, σ) を受け取った Bob は、Alice の検証鍵 pk_A を入手し、その検証鍵 pk_A を用いて (m, σ) の正当性をチェックする 正当なときのみメッセージ m を受けいれる。

ここで、Alice の署名鍵 sk_A は秘密に保管され、検証鍵 pk_A は公開されているので、

- 署名検証は Bob を含めてだれにでもできる。
- 署名生成は (署名鍵を知っている)Alice にしかできない。

ということが期待される。これが本当に正しいのなら、受信者 Bob から見

て、受け取ったメッセージ m は確かに Alice が署名したと確信でき、また、送信者 Alice から見ると、メッセージ m に署名したということを、Bob のみならず第 3 者に対しても否定できないことになる。

3 電子署名の定義

電子署名は、鍵生成アルゴリズム Gen、署名生成アルゴリズム Sign そして署名検証アルゴリズム Vrfy の 3 つの効率的なアルゴリズムの組と定義される。

鍵生成アルゴリズム Gen は、セキュリティパラメータ k を入力すると検証鍵 pk と署名鍵 sk の組を出力する：

$$(pk, sk) \leftarrow \mathsf{Gen}(k).$$

ここで、セキュリティパラメータ k とは署名の強度を指定するパラメータである。

署名生成アルゴリズム Sign は、署名鍵 sk とメッセージ m を入力すると署名 σ を出力する：

$$\sigma \leftarrow \mathsf{Sign}(sk, m).$$

署名検証アルゴリズム Vrfy は、検証鍵 pk とメッセージ m と署名 σ を入力とし、0 または 1 を出力する (0 は NG、1 は OK):

$$0/1 \leftarrow \mathsf{Vrfy}(pk, m, \sigma). \tag{1}$$

ただし、

完全性 Gen によって生成された任意の鍵ペア (pk, sk) と任意のメッセージ m に対して

$$\mathsf{Vrfy}(pk, m, \mathsf{Sign}(sk, m)) = 1$$

でなければならない。これは、正しく作られた署名は必ず受理されることを意味する。

また、安全性条件として

偽造不可能性 どのような効率的なアルゴリズム F も、(sk を知らずに)pk だけを用いて、与えられた m に対して妥当な署名 σ をつくることはできない

ことが必要である。

電子署名を作るとは、完全性と偽造不可能性を同時に満たす 3 つのアルゴリズムの組 $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ を求めることに他ならない。

4　RSA 署名

電子署名の一例として、RSA 署名を紹介する。RSA 署名は整数を法とする「べき乗剰余関数」の特性を利用してつくられる。

4.1　必要な整数論

まず、必要となる整数論的な事実と仮定を述べる。

n を異なる 2 つの素数 p,q の積である整数とする。$a \bmod n$ は整数 a を整数 n でわった余りを表す。また、a と b の差が n で割り切れるとき、a と b を n を法として等しいといい、$a \equiv b \pmod{n}$ とかく。

整数 n に対して $\mathbb{Z}_n = \{0, 1, \ldots, n-1\}$ とする。$l = \phi(n) = (p-1)(q-1)$ とする。

事実 1 $ed \equiv 1 \pmod{l}$ ならば任意の $a(\in \mathbb{Z}_n)$ に対して $(a^e)^d \equiv (a^d)^e \equiv a \pmod{n}$.

仮定 1 (RSA 仮定) $n(= pq)$ を十分大とする。y を $\mathbb{Z}_n$ からランダムに選ぶ。このとき、e, y, n を与えられて $y = x^e \bmod n$ となる x を求める効率的なアルゴリズムは存在しない。

仮定 1 は RSA 仮定と呼ばれ、その正しさは一般につよく信じられている。

事実 1 と仮定 1 は、整数 $n(= pq)$ に対して、べき乗剰余関数 $y = \mathrm{RSA}_{n,e}(x) = x^e \bmod n$ がトラップドア付き一方向関数であることを示している。

すなわち、順方向の $y = \mathrm{RSA}_{n,e}(x)$ の計算は、バイナリ法と呼ばれるアルゴリズムを用いて効率的に実行できる。しかし、仮定 1 より、逆に y から $x = \mathrm{RSA}^{-1}_{n,e}(y)$ を求める効率的なアルゴリズムは存在しない。

ところが、n の素因子 p, q が与えられれば、そのような逆方向の計算も効率的に実行できる: $x = \mathrm{RSA}^{-1}_{n,e}(p, q, y)$. (この p, q のような一方向性を破る情報をトラップドアと呼ぶ。) p, q を知っていると、$l = (p-1)(q-1)$ を計算でき、この l を用いれば、$d = e^{-1} \bmod l$ によって e から d を計算できるので、$x = y^d \bmod n$ が求める x である。実際、事実 1 より、$x^e \equiv (y^d)^e \equiv y \pmod{n}$ なので、$y = x^e \bmod n$ である。(これは、仮定 1 が成り立つためには、$n(= pq)$ の素因数分解が困難であることが必要なことを示している。)

4.2 署名方式

RSA 署名 $RSA = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ は、トラップドア付き一方向関数 $y = \mathrm{RSA}_{n,e}(x)$ を用いて、以下のように構成される。

鍵生成アルゴリズム $\mathsf{Gen}(k)$
: それぞれ k ビットの異なる素数 p, q を生成する。$n = pq, l = (p-1)(q-1)$ とし、$ed \equiv 1 \pmod{l}$ となる e, d を生成する。(n, e) を検証鍵 pk とし、(n, d) を署名鍵 sk とする。

署名生成アルゴリズム $\mathsf{Sign}(sk = (n, d), m)$
: 署名鍵 sk から n, d を取り出して、$\sigma = m^d \bmod n$ を計算し、σ を出力する。

署名検証アルゴリズム $\mathsf{Vrfy}(pk = (n, e), m, \sigma)$
: 検証鍵 pk から n と e を取り出して、$m' = \sigma^e \bmod n$ を計算する。結果 m' が m に等しければ 1 を異なれば 0 を出力する。

RSA 署名の完全性は事実 1 から直ちに従う。偽造不可能性は、与えられた m に対してその署名 σ は $m = \sigma^e \bmod n$ を満たすので、仮定 1 そのものである。

5 選択メッセージ攻撃における存在的偽造不可能性

5.1 RSA 署名の問題点

先に見たように、RSA 署名は与えられた (検証鍵 $pk = (n, e)$ と) メッセージ m に対して偽造不可能性をもつ。すなわち、どのような (署名鍵をもたない、効率的な) 敵も与えられた m に対して $\sigma^e = m \bmod n$ となる σ を生成することはできない。しかし、与えられた m に限らず、とにかく検証式 $\sigma^e = m \bmod n$ を満足するような (m, σ) ならすべて偽造と認めるならば (このような偽造を存在的偽造と呼ぶ)、RSA 署名はいくらでも偽造可能である。実際、σ をランダムに生成し、$m = \sigma^e \bmod n$ によって m を求め、(m, σ) を出力すればよい。このとき m もランダムになるので通常意味のあるメッセージとはならないが、アプリケーションによってはこのようなランダムメッセージに対する署名であっても意味をもつ場合がある。

また、アプリケーションによっては、敵が自ら選んだいくつかのメッセージに対する正当な署名を何らかの手段で入手できるような場合があり得る。このような状況における敵の攻撃を選択メッセージ攻撃と呼ぶが、この攻撃のもとでは RSA 署名は与えられた m に対してさえ偽造可能であ

る。実際、敵 F は次にようにして、(自分で選んだ) たった 2 つのメッセージ m_1, m_2 に対する署名 σ_1, σ_2 を入手すれば、与えられた m の署名 σ を計算することができる。

敵 F は検証鍵 $pk = (n, e)$ とメッセージ m を入力として受け取り、次のように動作する。まず、乱数 $r(\in \mathbb{Z}_n)$ を生成し、$m_1 = mr \bmod n$ を計算し、m_1 の正当な署名 σ_1 を入手する。次に、$m_2 = r^{-1} \bmod n$ を計算し、m_2 の正当な署名 σ_2 を入手する。最後に、σ_1 と σ_2 の積 $\sigma = \sigma_1\sigma_2 \bmod n$ を計算し、(m, σ) を出力する。

この σ に対して、$\sigma^e \equiv (\sigma_1\sigma_2)^e \equiv \sigma_1^e\sigma_2^e \equiv m_1m_2 \equiv m \bmod n$ となるので、(m, σ) は妥当である。

5.2 選択メッセージ攻撃における存在的偽造不可能性の定義

前節で RSA 署名では存在的偽造を防げないこと、また選択メッセージ攻撃においては (存在的と限らない) 偽造そのものを防げないことを見た。RSA 署名を、次節で見る RSA-FDH 署名のように、選択メッセージ攻撃において存在的偽造すらできないように強化することができる。

しかし、その前に、「選択メッセージ攻撃において存在的偽造すらできない」ということを、つぎにように敵 F と挑戦者 C との間のゲームを用いて定式化する。

選択メッセージ攻撃における存在的偽造不可能性

電子署名 $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ に対して、敵 F と挑戦者 C との間でゲームを行う。ルールは以下の通り。まず、挑戦者 C が鍵生成アルゴリズム Gen を実行し、鍵ペア (pk, sk) をつくる。挑戦者 C は検証鍵 pk を敵 F に渡す。pk を受け取ったら、敵 F は何らかの計算をしてメッセージ m_i を選択し、挑戦者 C に渡す。これに対して挑戦者 C は署名鍵 sk を用いて m_i の署名 σ_i を計算する: $\sigma_i \leftarrow \mathsf{Sign}(sk, m_i)$. 挑戦者 C は署名 σ_i を敵 F に渡す。このような m_i と σ_i のやりとりを敵 F は挑戦者 C との間で好きなだけ繰り返すことができるものとする。敵 F は受け取った (複数の)σ_i の情報を用いて、何らかのメッセージと偽造署名の対 (m, σ) を出力する。もしも m がどの m_i とも異なり、σ が m の妥当な署名 (すなわち $\mathsf{Vrfy}(pk, m, \sigma) = 1$) ならば、敵 F の勝ち (そうでないなら挑戦者 C の勝ち) とする。

このとき、どのような敵 F もほとんど 0 に等しい確率でしか勝てないならば、電子署名 $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ は**選択メッセージ攻撃において存在的偽造不可能**であると呼ぶ。

6 RSA-FDH 署名

RSA 署名において、メッセージにハッシュ関数を施すことで、より偽造が困難な RSA-FDH 署名が得られる。ここで、ハッシュ関数 H とは (暗号学的な) 圧縮関数であって、全てのビット列の集合 $\{0,1\}^*$ から $\mathbb{Z}_n$ への関数である：$H : \{0,1\}^* \to \mathbb{Z}_n$. ただし、どのような効率的なアルゴリズムも $H(x) = H(y)$ となる異なる x, y を見つけることはできないとする (衝突困難性)。

RSA-FDH 署名=(Gen, Sign, Vrfy) は以下の通り。

鍵生成アルゴリズム $\mathsf{Gen}(k)$ /* RSA 署名と同じ */

それぞれ k ビットの異なる素数 p, q を生成する。$n = pq, l = (p-1)(q-1)$ とし、$ed \equiv 1 \pmod{l}$ となる e, d を生成する。(n, e) を検証鍵 pk とし、(n, d) を署名鍵 sk とする。

署名生成アルゴリズム $\mathsf{Sign}(sk = (n, d), m)$

署名鍵 sk から n, d を取り出して、$\sigma = H(m)^d \bmod n$ を計算し、σ を出力する。

署名検証アルゴリズム $\mathsf{Vrfy}(pk = (n, e), m, \sigma)$

検証鍵 pk から n と e を取り出して、$h' = \sigma^e \bmod n$ を計算する。結果 h' が $h = H(m)$ に等しければ 1 を異なれば 0 を出力する。

RSA-FDH 署名に対しては、RSA 署名に対して成立した先の攻撃はうまくいかない。

まず、存在的偽造について見る。先と同様に、σ をランダムに生成し、$h = \sigma^e$ とする。偽造を完成させるにはこの h に対して、さらに $h = H(m)$ となる m を求めなければならない。これは衝突困難性より不可能である。(実際、h から $h = H(m)$ となる m が求まるようだと、ランダムな x に対して $h = H(x)$ としておけば、m, x が衝突ペアとなってしまう。)

また、先の選択メッセージ攻撃も、やはり衝突困難性より $H(m) = H(m_1)H(m_2)$ とはならないので、うまくいかない。(実際、$m = m_1 m_2 \bmod n$ である任意の m_1, m_2 に対していつも $H(m) = H(m_1)H(m_2)$ となるようだと、$H(m)$ の値は m そのものよりも $m \bmod n$ で決まっている。)

しかし、だからといってどのような選択メッセージ攻撃を行っても存在的な偽造ができないといえるだろうか? ランダムオラクルモデルと呼ばれるヒューリスティックに頼ればそのようにいえる。

定理 1 *RSA 仮定 (仮定 1)* が真ならば、*RSA-FDH* 署名はランダムオラクルモデルのもとで選択メッセージ攻撃において存在的偽造不可能である。

ここで定理 1 の証明をフォーマルに述べることはできないが、そのスケッチは紹介したい。暗号理論において安全性がどのように証明されるのか、その雰囲気は伝わると思う。

まず、ランダムオラクルモデルについて述べる。ランダムオラクルモデルとは、アルゴリズムの実行モデルであって、各アルゴリズムにおけるハッシュ関数 $H(\cdot)$ の実行を、ランダムオラクル O_H への問い合わせに置き換えてしまうモデルである。各アルゴリズムは m に対して $H(m)$ を自力では計算せず、なぜか必ずランダムオラクル O_H へ m を問い合わせる。m を受け取ったランダムオラクル O_H は m に対して乱数 $h(\in \mathbb{Z}_n)$ を割り当てて $H(m)$ の値として返す。h を受け取ったアルゴリズムは $h = H(m)$ として以降の計算を通常どおり続行する。

ハッシュ関数 $H(\cdot)$ には衝突困難性が求められるので、通常、その値 $h = H(m)$ は乱数にしか見えないようなものになる。それなら $H(\cdot)$ の計算をランダムオラクル O_H への問い合わせに置き換えても安全性の検証上、一定の意味があるだろうという発想である。

つぎに、定理 1 の証明のスケッチを紹介する。選択メッセージ攻撃のもとで RSA-FDH 署名の存在的偽造に成功する敵 F が存在したと仮定する。すなわち、ある敵 F があって、F は先にみたゲームを挑戦者 C との間で実行し、挑戦者 C に勝つとする。このゲームにおいて、F はランダムオラクルに何回か問い合わせを行い (ランダムオラクルモデル)、また挑戦者 C に対していくつかのメッセージの署名を要求し、最終的にこれらの結果を用いて RSA-FDH 署名の偽造署名 (m, σ) を出力する。我々は、このような F があれば、それを用いて RSA 関数の逆関数の値を計算できる敵 I を作り出せることを示す。つまり、I は、F が署名を偽造する能力を利用して RSA 関数の逆関数の値を計算するわけである。しかし、このような I の存在は RSA 仮定に反するので定理の仮定のもとで許されない。そのようなことになってしまったのは上のような敵 F が存在すると仮定したためである。よって、F は存在せず、RSA-FDH 署名はランダムオラクルモデルのもとで選択メッセージ攻撃において存在的偽造不可能である。以上が、証明の大枠である。

目標である (RSA 仮定に対する) 敵 I は (RSA-FDH 署名に対する) 敵 F を用いて以下のように構成される。(敵 F がランダムオラクルに問い合わせを行う回数を q とする。) I には入力として n, e, y が与えられる。I の目的は $y = \mathrm{RSA}_{n,e}(x)$ となる x を求めることである。I は以下のようにして、F に対して挑戦者 C の役割を演じ、F の能力を利用する。I はまず 1 以上 q 以下の範囲の乱数 i^* を生成しておく。I は自身のサブルーチンとし

て F を起動する。つぎに I は挑戦者 C として $pk = (n, e)$ を RSA-FDH 署名の検証鍵ということにして F に与える。F は RSA-FDH 署名に対する攻撃を開始する。攻撃の途中で、F はランダムオラクルにハッシュ値を問い合わせたり、挑戦者 C に署名を問い合わせたりする。これらに対して I は以下のように返答する。

- ランダムオラクルへの (i 回目の) 問い合わせ m_i に対して
  I は $i = i^*$ かどうかみる。もしそうならば I は (ランダムオラクルからの返答として)y を返す。そうでないならば、x_i を $\mathbb{Z}_n$ からランダムに選択し、$y_i = \mathrm{RSA}_{n,e}(x_i)$ を計算し、y_i を返す。

- 挑戦者 C への署名問い合わせ m に対して
  I は $m = m_i$ となる i を求める。(F は m をすでにランダムオラクルに尋ねているとしてよい。その方が F にとって情報が増えて有利だから。) $i = i^*$ なら I は (RSA 仮定に対する) 攻撃をあきらめる。そうでないなら、I は x_i を (m の署名として)F に返答する。

これらの問い合わせ結果に基づき、F は何らかの計算を行って (m^*, σ) を出力してくる。これに対し、$m^* = m_{i^*}$ なら I は σ を出力して終了する。(ここでも F は m^* をすでにランダムオラクルに尋ねているとしてよい。その方が F にとって情報が増えて有利だから。) そうでないなら I は攻撃をあきらめる。

上で、F は I の中で起動されているので、F にとって I がその世界の全てである。そのため、I は F に対してランダムオラクルや挑戦者の役割を演じている。もしもこの演技 (シミュレーション) がまずければ、F はまともに動かず、その能力を利用できない。まず、I が F に与える入力 $pk = (n, e)$ は RSA-FDH 署名における公開鍵として妥当である。次に、I によるランダムオラクルのシミュレーションを検証する。I は $i = i^*$ のとき y を答えている。この y はもともと I への課題として $\mathbb{Z}_n$ からランダムに選ばれている。よって F からみてもランダムな $\mathbb{Z}_n$ の要素なのでシミュレーションは OK である。(ランダムオラクルは $\mathbb{Z}_n$ のランダムな要素を返すのだった。) $i \neq i^*$ のときは、I は x_i を $\mathbb{Z}_n$ からランダムに選択しているので、$y_i = \mathrm{RSA}_{n,e}(x_i)$ も $\mathbb{Z}_n$ のランダムな要素である。よって、やはり OK。さらに、I による署名問い合わせに対する応答のシミュレーションを検証する。上で I は問い合わせ $m = m_i$ に対して $i \neq i^*$ のとき x_i を答えている。この答えは m に対する署名として正しい。実際、$(x_i)^e = y_i = H(m_i)$ である。ここで辻褄があうように I は $H(m_i)$ の値を $y_i (= x_i^e)$ に決めていたのである。(I がこのような “ズル” をしていようとは F からはまったく見えない。)

以上から、I は F に対してランダムオラクルや挑戦者の役割をほぼうまく演じていることがわかった。(うまくいかないケースは I が署名問い合わせに応答する際に $i = i^*$ となってしまう場合である。この場合は、I はあきらめてしまうのだった。しかし、i^* はランダムに選ばれているので、こうはならないケースは無視できない程度 (確率 $1/q$ 以上) にはある。) よって、F は攻撃に成功し、妥当なメッセージと署名の組 (m^*, σ) を出力する。ここで、もしも $m^* = m_{i^*}$ となっていると、I によるランダムオラクルのシミュレーションの仕方から、$\sigma^e = H(m^*) = y$ なので、σ が目的の ($y = \mathrm{RSA}_{n,e}(x)$ となる)x である。もちろん、$m^* = m_{i^*}$ となるとは限らないが、i^* はランダムに選ばれているので、$m^* = m_{i^*}$ となる確率は $1/q$ 以上で無視できない程度にはあり、RSA 仮定を破るには十分である。