デファクト暗号の評価

北陸先端科学技術大学院大学

宮地 充子

本報告書は，デファクト暗号である OAEP-RSA，DH-鍵共有法，DSA-署名の安全性について報告する．

目 次

第1章　素因数分解問題

要旨

代表的な現代暗号では, 暗号解読の難しさが数論の難問を解く難しさに対応している. その1つに挙げられるRSA暗号は大きな素因数分解問題の難しさに安全性の根拠をおいたものである. 一般に, 大きな整数の素因数分解は難しいとされており, 現在知られている最良の解法でその計算量が見積もられ, 準指数的な時間を要することがわかっている. 素因数分解問題が多項式時間で解ける問題に属するか, またそのような解法が存在するかどうかは明らかにされていない. 本章では, 素因数分解問題に対する有望なアルゴリズムのいくつかを簡単に紹介し, その計算量などについて述べる.

abstract

The security of many public-key cryptosystems relies on the apparent intractability of the computational problems. The security of RSA cyptosystem depends upon the intractability of the integer factorization problem. But, true computational complexities of the factorization problem is not known. If there is an algorithm which can solve a non-negligible function of all instances of a problem in polynomial time, then any cryptosystem whose security is based on that problem must be considered insecure. This chapter summarizes the current knowledge on algorithms for the integer IFfactorization problem.

1.1　素因数分解問題

素因数分解問題とは、次の問題をいう.

素因数分解問題: 整数 N が与えられたとき, $N = pq$ となるような整数 p, q を求めよ.

この問題の効率的な解法を見つけることができれば、大きな整数 N も最終的に素数の積に分解できる. 素因数分解問題が多項式時間で解ける問題のクラスに属するか, 時間を要する非常に難しい問題のクラスに属するかは いまだ明らかにされていない. 現在知られている最良の解法でその計算量が見積もられている.

1.2　素因数分解アルゴリズム

現在知られている大きな数 N の素因数分解に有効なアルゴリズムはいずれも, 適当な z を求め, N と z の最大公約数 $p = \gcd(N, z)$ を計算し, N の因数 $p\ (= 1)$ を求めることが基本となっている. 最大公約数はEuclidの互助法で容易に求められる. 問題は N と互いに素でない z の求め方であり, これまでにも様々な手法が提案されている.

素因数分解のアルゴリズムは,

1. 素因数 p の性質によって決まる素因数依存型,
2. 計算量が合成数 N のサイズのみによって決まる合成数依存型,

にわけられる.

1.2.1　素因数依存型のアルゴリズム

素因数依存型の主要なアルゴリズムとしては, 試行割算法, Pollard の ρ 法, Pollard の $p-1$ 法 [8], Guy の $p+1$ 法 [4], H.W.Lenstra によって提案され [5], Montgomery らによって改良された楕円曲線法 [7] などが知られている.

試行割算法: 合成数 N を小さな素数で順に割っていく手法. 計算時間は $O(p)$ であり, 最悪の場合 $O(\sqrt{N})$ の計算時間を要する. 実際, 現在の計算機能力では p が 10^6 程度以下の場合にのみ有効である.

ρ 法: 適当な多項式 f と初期値 x_1 から $x_{i+1} = f(x_i) \mod N$ を計算して, $\gcd(N, x_{2j} - x_j)$ より素因数 p を求める手法. この計算量は $O(\sqrt{p})$ である.

$p-1$ 法: 計算量は N の素因数 p に対し, $p-1$ の最大素因数のサイズに比例する. $p-1$ が B-スムーズであるとき, その計算量は $O(B \log N / \log B)$ となる. つまり, この手法は $p-1$ が小さな素数の積になっている場合に有効である.

$p+1$ 法: 計算量は N の素因数 p に対し, $p+1$ の最大素因数のサイズに比例する. つまり, $p+1$ が小さな素数の積になっている場合に有効である. $p-1$ 法と同様, $p+1$ が $B-$ スムーズであるとき, その計算量は $O(B \log N / \log B)$ である.

楕円曲線法: $p-1$ 法の拡張であり, 種数が 1 の代数曲線である楕円曲線を用いた素因数分解法である. 平均的かつ漸近的な計算量は N の最小素因数 p のサイズに依存し $L_p[1/2, \sqrt{2}]$ である. 最悪の場合は $N = pq$ かつ p と q のサイズが同じ場合であり, このときの計算量は $L_N[1/2, 1.020]$ となる.

ここで, アルゴリズムの計算量評価に用いる関数 $L_p(a, b)$ を

$$L_p[a, b] = \exp((b + o(1))(\log p)^a (\log \log p)^{1-a})$$

と定義する. ($o(1)$ は極限値が 0 の量)

1.2.2　合成数依存型のアルゴリズム

合成数依存型の主要なアルゴリズムは, 2 次合同式 $s^2 \equiv t^2 \mod N$ をみたす s, t を求め, $z = s \pm t$ とし, 最大公約数 $\gcd(N, z)$ を計算することで素因数を導くのである. この具体的なアルゴリズムとしては, Morrison-Brillhart の連分数法, Brillhart のフェルマー法 [2], Schroeppel の線形ふるい法, Pomerance によって提案され [9], Silverman によって改良された 2 次ふるい法 [10], Lenstra 兄弟, Manasse, Pollard の数体ふるい法 [6] などが知られている.

フェルマー法: N が 2 つの平方数の積として表されたとき, これら 2 つの平方数の一方が大変小さくなるという事実を用いる. これは $N = pq$ かつ p, q が同じ大きさである場合に有効である. 計算量は $O((p-q)^2 / N^{1/2})$ である.

2 次ふるい法: 合同式 $s^2 \equiv t^2 \mod N$ をみたす s, t を求めるため, 合成数 N に対して, 関数 $Q(x) = (x + \lfloor \sqrt{N} \rfloor)^2 - N$ を用意し, $x = 0, \pm 1, \pm 2, \ldots$ を代入した値を求める. これらの $Q(x)$ の素因数分解例を集めて掛け合わせ, 小さな素因数のべき指数がすべて偶数になるようにする. つまり, $s^2 \equiv t^2 \mod N$ をみたす s, t を求めるが, $(x + \lfloor \sqrt{N} \rfloor)$ を掛け合わせた部分が s であり, $Q(x)$ の素因数を掛け合わせた部分の平方根が t となる. ここで $z = s \pm t$ とし, $\gcd(N, z)$ を計算することで N の素因数を求める手法である. 素因数の大きさのオーダが $\sqrt{N}$ よりずっと小さいものを含まない場合に有効である. 計算量は $L_N[1/2, 1.020]$ となる.

数体ふるい法: この手法は, 2 次ふるい法の概念を代数的整数上で実現したものである. 計算量は $L_N[1/2, 1.117]$. 合成数が $N = c^k \pm s$ (c, s : 小さな数, k : 大きな数) の形に限定された場合の平均的な計算量は $L_N[1/3, 2(2/3)^{2/3}] = L_N[1/3, 1, 526\cdots]$ である.

また, 数体ふるい法を一般形の合成数の素因数分解に適用できるような試みも多く行なわれている.

汎用数体ふるい法: Buhler, Lenstra, Pomerance によって提案された手法による計算量は $L_N[1/3, 3^{2/3}] = L_N[1/3, 2, 080\cdots]$ である. さらに, Adleman [1] と Lenstra によって独立に提案された改良アルゴリズムの計算量は $L_N[1/3, 4 \cdot 3^{-2/3}] = L_N[1/3, 1, 922\cdots]$. その後提案された Coppersmith [3] によるアルゴリズムの計算量は $L_N[1/3, 1.901]$ となっている.

1.3　素因数分解アルゴリズムのまとめ

現在知られている有効なアルゴリズムとその計算量, アルゴリズムを有効にする条件についてまとめる.

	計算量	アルゴリズムを有効にする条件
ρ 法	$O(\sqrt{\ell})$	汎用的 (全ての合成数)
$p-1$ 法	$O(B \log N / \log B)$	$p-1$, あるいは $q-1$ が $B-$ スムーズのとき
$p+1$ 法	$O(B \log N / \log B)$	$p+1$, あるいは $q-1$ が $B-$ スムーズのとき
楕円曲線法	$L_p[1/2, \sqrt{2}]$	最大素因数 p の大きさが小さいとき
フェルマー法	$O((p-q)^2/N^{1/2})$	$\lvert p-q \rvert$ の差 $\Delta < N^{1/4}$ のとき
2 次ふるい法	$L_N[1/2, 1.020]$	p, q のサイズが同じとき
数体ふるい法	$L_N[1/2, 1.117]$	$N = c^k \pm s$ (c, s: 小さな数, k: 大きな数) のとき
汎用数体ふるい法	$L_p[1/3, 1.901]$	汎用的 (全ての合成数)

従って, RSA 暗号など素因数分解の難しさに安全性の根拠をおいた暗号を使用する際は, 上記の素因数分解アルゴリズムを困難にするパラメータ (素数 p, q) を設定する必要がある.

関連図書

[1] Adleman, L.M., "Factoring Numbers Using Singular Integers", Proc. of STOC, pp.64-71 (1991).

[2] J. Brillhart., "Fermat's factoring method and its variants", Congressus Numerantium, 32, pp. 29-48 (1981)

[3] Coppersmith, D., "Modifications to the Number Field Sieve", *J, Cryptology*, 6, 3, pp. 169-180(1993).

[4] R. K. Guy., "How to factor a number", Proc. 5th Manitoba Conf. on Numerical Math., pp. 49-89 (1975)

[5] Lenstra, H.W.Jr., "Factoring Integers with Elliptic Curves", *Annals of Mathematics*, 126, pp.649-673 (1987).

[6] Lenstra, A. K., Lenstra, H.W.Jr., Manasse, M. S. and Pollard, J. M., "The Number Field Sieve", Proc. of STOC, pp.564-572 (1990).

[7] Montgomery, P. L., "Speeding the Pollard and Elliptic Curve Methods of Factorization", *Math., Comp.*, pp. 243-264 (1987).

[8] J. M. Pollard., "Theorems on Factorization and Primality testing", Proc. Comb. Philis. SOC., 76, pp. 521-528 (1974)

[9] Pomerance, C., *The Quadratic Sieve Algorithm*, Lecture Notes in Computer Science 209, pp. 169-182 (1985).

[10] Silverman, R.D., "The Multiple Polynomial Quadratic Sieve", *Math. Comp.*, 48, pp.243-264 (1987).

第2章　離散対数問題

要旨

離散対数問題の素因数分解問題と同様, 多項式時間で解ける問題に属するか, またそのような解法が存在するかどうかが明らかにされていない問題である. 離散対数問題において重要なことは, 大きな位数の群では一般に離散対数の計算が難しいという点である. 即ち, 離散対数問題を解く効率的なアルゴリズムが発見されていないことが, 暗号への応用を可能にしている. 本章では離散対数問題に対する有望なアルゴリズムのいくつかを簡単に紹介し, その計算量などについて述べる.

abstract

The security of many cryptographic techniques depends on the intractability of the descrete logarithm problem. But, true computational complexities of the discrete logarithm problem is not known. This chapter summarized the current knowledge refarding algorithm for solving the discrete logarithm problem.

2.1　離散対数問題

離散対数問題とは, 次の問題をいう.

離散対数問題: 群 G と $G \ni g, y$ に対して, $y = g^x$ となる x が存在するならば求めよ.
(以後, $x = \log_g y$ という記法を用いる.)

この問題も効率的な解法をみつけることができれば, 大きな位数での対数を求めることができる. 離散対数問題が多項式時間で解ける問題のクラスに属するか, 難しい問題のクラスに属するかは いまだ明らかにされていない. 現在知られている最良の解法でその計算量が見積もられている.

2.2　離散対数問題を解くアルゴリズム

離散対数問題のアルゴリズムは, 次の 2 種類に大別することができる.

1. $H = \langle g \rangle \subseteq G$ としたとき H の位数 ℓ に依存して $\log_g y$ を求めるアルゴリズム. その計算時間は ℓ の最大素因数のサイズの指数時間オーダ $O(\sqrt{\ell})$ の実行時間となる.

2. $G = \mathbb{F}_q^*$ $(q = p^k)$ として, 指数計算法とよばれる手法で $\log_g y$ を求めるアルゴリズム. その計算時間は有限体 $\mathbb{F}_q$ のサイズの準指数時間オーダの実行時間となる.

前者に属するものとしては, Shanks [10], Pohlig-Hellman [7], Pollard [8] のアルゴリズムなどが知られている. 後者に属するものとしては, Adleman [1], Coppersmith [2], ElGamal [4], Pomerance [9], Coppersmith-Odlyzko-Schroeppel [3], Gordon [5] [6] のアルゴリズムなどが知られている.

2.2.1　位数に依存するアルゴリズム

Shanks のアルゴリズム: $G \ni g$ とし, $H = \langle g \rangle \subseteq G$ の位数を ℓ, $m = \lceil \ell^{1/2} \rceil, 0 \leq r, q \leq m$ とする. また, H に対して $f : H \to \{1, \ldots, n\}$ なる単射が存在し, $\log \ell$ の低次多項式時間で計算できるとする.

Step 1. 次の集合 L_1, L_2 を計算する.

$$L_1 = \{(i, f(yg^i)) \mid 0 \leq i \leq m\},$$

$$L_2 = \{(i, f(g^{mi})) \mid 0 \leq i \leq m\}.$$

Step 2. L_1, L_2 の各元を第 2 成分についてソートする.

Step 3. L_1 の元の第 2 成分と L_2 の元の第 2 成分が一致しているものを探索する. これを $(r, f(yg^r)) \in L_1$, $(q, f(g^{mq})) \in L_2$ とおく.

このとき, $yg^r = g^{mq}$ より, $y = g^{mq-r}$ が成り立つ.

故に r, q を求めることで $\log_g y = mq - r$ が得られる.

このアルゴリズムにおける全体の計算時間は $O(\sqrt{\ell})$ となる. さらに $O(\sqrt{\ell})$ 個の元を格納するための記憶領域も必要となる.

Pohlig-Hellman のアルゴリズム: $H = \langle g \rangle \subseteq G$ の位数を ℓ とする.

Step 1: ℓ が

$$\ell = \prod_{i=1}^{k} {p_i}^{e_i}, \quad p_1 < \cdots < p_k \ (p_i : \text{異なる素数})$$

と分解されているとする.

Step 2: 各 $\mathbb{Z}_{{p_i}^{e_i}}$ における $\log_g y$ を求める.

Step 3: 中国人剰余定理

$$\mathbb{Z}_\ell \simeq \mathbb{Z}_{{p_1}^{e_1}} \times \cdots \times \mathbb{Z}_{{p_k}^{e_k}}$$

より Step 2 で求めた値を合成することで $\log_g y \in \mathbb{Z}_\ell$ を求める.

このアルゴリズムの実行時間は, Step 2 に依存し, $O(\sum_{i=1}^{k} e_i(\log \ell + \sqrt{p_i}))$ である. よって ℓ の最大素因数 p_k のサイズの指数関数オーダとなる. しかし, ℓ が $O(\log \ell)$- スムーズであるとき, $O(\sqrt{\log \ell})$ を得る. 即ち, $p - 1$ が小さな素数の積になっている場合に有効である.

2.2.2　指数計算法

指数計算法は効率よい因数分解アルゴリズムと多くの類似点をもつ. 一般に, 指数計算法のアルゴリズムは 2 部構成となっており, Step 1 では, $H = \langle g \rangle \subseteq G$ から適切に選んだ部分集合 (因子基底) の離散対数を求めてデータベースとして蓄積し, Step 2 では, このデータベースを利用して実際に $\log_g y$ を求める. この一般的アルゴリズムを以下に記述する.

指数計算法の一般的アルゴリズム

Step 1: $H = \langle g \rangle \subseteq G$, H の位数を ℓ とする. 因子基底として $\mathcal{B} = \{p_1, \ldots, p_m\} \subseteq H$ を定め,

$$g^{b_i} = \prod_{j=1}^{m} {p_j}^{a_{ij}}$$

と分解される b_i を探す. 両辺の対数をとれば,

$$b_i \equiv \sum_{j=1}^{m} a_{ij} \log_g p_j \pmod{\ell}$$

となり, これは $\log_g p_j$ を未知数とする方程式と見ることができる. このような b_i に関して, $\sharp\{b_i\} \geq \sharp\mathcal{B}$ となったとする. このとき, $\mathbb{Z}_\ell$ 上の行列 $A = (a_{ij})$ の階級が $m(= \sharp\mathcal{B})$ となれば, $\log_g p_j$ $(1 \leq j \leq m)$ に関する線形方程式は一意の解をもち, $\mathcal{B}$ の各元の離散対数がわかる.

Step 2: ランダムに r を選び, yg^r が以下のように分解されるまで続ける.

$$yg^r = \prod_{j=1}^{m} {p_j}^{e_j}$$

このとき 両辺の対数ととることによって

$$\log_g y = \sum_{j=1}^{m} e_j \log_g p_j - r$$

となり, $\log_g y$ を求めることができる.

Adleman のアルゴリズム: 連分数法による素因数分解のアイデアを離散対数問題に応用したアルゴリズム. $G = H = \mathbb{Z}_p^*$ の離散対数問題に対して, 因子基底 $\mathcal{B}$ として $u = L_p[1/2, 0]$ 以下の素数の集合を設定するものである. 即ち, 上のアルゴリズムにおける Step 1 では b_i をランダムに選び, g^{b_i} がスムーズになるような b_i のみをふるいにかけることになる. このように $\mathcal{B}$ を選ぶことで, g^{b_i} がスムーズになる確率の評価に若干の仮定が入るがアルゴリズムの実行時間は $L_p[1/2, c](c \approx 1)$ となる.

Pomerance のアルゴリズム: Adleman のアルゴリズムを改良して, 仮定なしの厳密な実行時間を評価したアルゴリズムである. その速度は Adleman のアルゴリズムとほぼ同等で, $L_p[1/2, \sqrt{2}]$ である.

Gordon のアルゴリズム: 数体ふるい法による素因数分解の概念を応用したアルゴリズム. 一般的数体ふるい法に基づくものと, 特殊数体ふるい法に基づくものの 2 種類がある. 前者の場合は一般の p に対して適用可能であり, 実行時間は $L_p[1/3, 3^{2/3}] = L_p[1/3, 2.08008]$ である. 後者の場合はある性質をもつ p に対して適用可能であり, 実行時間は $L_p[2/5, 1.00475]$ である. ここでいう p の特別な性質とは, 以下のような整係数既約モニック多項式 f の選択を可能にするということである.

1. $f \in \mathbb{Z}_p[X]$ の係数は適当に小さい.
2. ある整数 x, y が存在し, その大きさはどちらも $p^{1/k}$ 程度で, $y^k f(x/y) \equiv 0 \pmod{p}$ をみたす.
3. $\mathcal{O}_K = \mathbb{Z}[\alpha]$ は一意分解整域.

2.3　離散対数問題アルゴリズムのまとめ

現在知られている有効なアルゴリズムとその計算量, アルゴリズムを有効にする条件についてまとめる.

アルゴリズム	計算量	アルゴリズムを有効にする素数の条件
Shanks のアルゴリズム	$O(\sqrt{\ell})$+ 記憶領域 $O(\sqrt{\ell})$	汎用的 (全ての素数)
Pohlig-Hellman のアルゴリズム	$O(\sum_{i=1}^{k} e_i(\log \ell + \sqrt{p_i}))$	$p-1$ が小さな素数の積のとき
Adleman のアルゴリズム	$L_p[1/2, c]$	汎用的 (全ての素数)
Pomerance のアルゴリズム	$L_p[1/2, \sqrt{2}]$	汎用的 (全ての素数)
Gordon のアルゴリズム	$L_p[1/3, 3^{2/3}]$	汎用的 (全ての素数)

従って, 離散対数問題の難しさに安全性の根拠をおいた暗号を使用する際は, 上記のアルゴリズムを困難にするパラメータ (素数 p) を設定する必要がある.

関連図書

[1] Adleman, L.M., "A subexponential Algorithm for the Discrete Logarithm Problem with Applications to Cryptography", Proc. of FOCS, pp.50-60 (1979).

[2] Coppersmith, D., "Fast Evaluation of Logarithms in Fields of Characteristic Two", IEEE Trans. Inform. Theory, IT-30, pp.587-594 (1984).

[3] Coppersmith, D., Odlyzko, A. M. and Schroeppel, R., "Discrete Logarithms in $GF(p)$", *Algorithmica*, Vol.1, pp.472-492 (1986).

[4] ElGamal, T., "A subexponential-Time Algorithm for Computing Discrete Logarithms over $GF(p^2)$", *IEEE Trans. Inform. Theory*, IT-31, pp.473-481 (1985).

[5] Gordon, D. M., "Discrete Logarithm in $GF(p)$ Using the Number Field Sieve", to appear in *SIAM Journal on Discrete Math.*

[6] Gordon, D. M., "Designing and Detecting Trapdoors for Discrete Log Cryptosystems", Proc. of CRYPTO'92, LNCS 740, pp.66-75 (1992).

[7] Pohlig, S. and Hellman, M., "An Improved Algorithm for Computing Logarithms over $GF(p)$ and Its Cryptographic Significance", *IEEE Trans. Information Theory*, 24, pp. 106-110 (1978).

[8] Pollard, J. M., "Monte Carlo Methods for Index Computation mod p", *Math. Comp,*. 32, pp.918-924 (1978).

[9] Pomerance, C., "Fast, Rigorous Factorization and Discrete Logarithm Algorithm, Discrete Algorithms and Complexity", Proc. of the Japan-U.S. Joint Seminar, Academic Press, pp. 119-143 (1987).

[10] Shanks, D., "Class Number, a Theory of Factorization", and Genera, Proc. Symposium Pure Mathematics, AMS (1972).

第3章　RSA-OAEP の評価

要旨

RSA-OAEP は，RSA 暗号関数を安全性の根拠とした素因数分解問題に基づく暗号方式である．本章においては，RSA-OAEP のプリミティブ（素因数分解問題）に対する安全性及びスキーム（OAEP 変換）に対する安全性について記述する．

RSA-OAEP は，RSA 暗号関数が一方向性関数であるという仮定の下で，適応的選択平文攻撃において強秘匿 (IND-CCA2) であることを報告する．

abstract

RSA-OAEP is an security-enhanced encryption scheme based on RSA-encryption function, which is categorized into Integer-Factorization(IF)-based public key cryptosystems. In this report, we discuss the security on RSA-OAEP in view of both the primitive basis like IF and the scheme like OAEP-transformation.

We report that RSA-OAEP is semantically secure against the adaptive cipher chosen attack under the assumption that RSA-encryption is a one-way permutation.

3.1　RSA-OAEP について

OAEP(Optimal Asymmetric Encryption Padding) は, a trapdoor permutation generator f (落し戸付き一方向性置換) を変換し，安全性を強化する方法である ([1]). ここで，正整数 k_0, k_1, n に対し，$k = n+k_0+k_1$ とし，置換関数 f は,

$$f : \{0,1\}^{n+k_1} \times \{0,1\}^{k_0} \longrightarrow \{0,1\}^{n+k_1} \times \{0,1\}^{k_0}$$

により与えられる．OAEP 変換に用いる oracles G 及び H を以下の入出力関数とする．

$$\begin{aligned} G: \quad & \{0,1\}^{k_0} \longrightarrow \{0,1\}^{n} \\ G: \quad & \{0,1\}^{n} \longrightarrow \{0,1\}^{k_0} \end{aligned}$$

上記 G 及び F を用いて，f の OAEP 変換は以下のように定義される．

暗号化

1. 入力を $x \in \{0,1\}^n$ とする．
2. 乱数 $r \in \{0,1\}^{k_0}$ を生成する．
3. $s = (x||0^{k_1}) \oplus G(r)$ とする．
4. $t = r \oplus H(s)$ とする．
5. $w = s||t$ とする．
6. $y = f(w)$ を出力する．

復号化

1. 入力を $y \in \{0,1\}^k$ とする.
2. $w = f^{-1}(y)$ とする.
3. s を w の最初の $n + k_1$ ビットとする.
4. t を w の最後の k_0 ビットとする.
5. $r = t \oplus H(s)$ とし，$z = G(r) \oplus s$ とする.
6. Z の最後の k_1 ビットが 0^{k_1} ならば，x を出力．そうでなければ，$*$ を出力する.

RSA - OAEP とは，上記 OAEP 変換において，一方向性置換 f に RSA 暗号関数

$$f : \mathbb{Z}_N \longrightarrow \mathbb{Z}_N (x \to x^e \pmod N) \tag{3.1}$$

を用いるものである．ここで, n は 2 つの大きな素数の積 $n = pq$ であり，e は RSA 暗号の公開鍵，すなわち $l = \mathrm{lcm}(p-1, q-1)$ に対して，$ed \equiv 1 \pmod l$, $e, d \in \mathbb{Z}_N$ により定められる.

3.2　プリミティブ（素因数分解）の攻撃法について

RSA-OAEP の基本となる一方向性関数 $f(x) = x^e \pmod N$ は，素因数分解に基づく．(ただし現時点で，RSA 暗号関数は，素因数分解と等価であると示されていない.) この結果，一方向性関数を構成する素因数 p, q を素因数分解が容易になるように選択すると，RSA-OAEP は解読可能となる．素因数分解の攻撃法に関する議論については，他の素因数分解に基づく暗号方式と全く同等の議論が成立する．詳しくは，?? 章を参照されたい.

3.2.1　プリミティブの攻撃に対する安全性評価

前章で述べたように素因数分解が可能なパラメータの選択を行うと RSA-OAEP は容易に解読される．このような直接攻撃を回避するために，?? 章を満たすように，p, q を生成する必要がある.

3.3　スキーム (RSA-OAEP) への攻撃法について

まず初めに，一方向性置換として用いられる RSA 暗号関数について定義する.

Definition 1 (RSA 暗号関数) n は 2 つの大きな素数の積 $n = pq$ であり，e は *RSA* 暗号の公開鍵，すなわち $l = lcm(p-1, q-1)$ に対して，$ed \equiv 1 \pmod l$, $e, d \in \mathbb{Z}_N$ により定められる元とする．このとき，*RSA* 暗号関数 f とは以下のように定義される.

$$f : \mathbb{Z}_N \longrightarrow \mathbb{Z}_N (x \to x^e \pmod N) \tag{3.2}$$

を用いるものである.

次章では，RSA 暗号関数に特化した攻撃方法及び RSA-OAEP の安全性について述べる.

3.3.1　RSA 暗号関数への攻撃

RSA-OAEP は RSA 暗号関数の逆関数を求めるのが困難であるということを仮定する．しかしながら，パラメータ p, q, e, d の設定によっては，RSA 暗号関数の逆関数が容易に求められる場合がある．このうち，RSA-OAEP のプリミティブに相当する素因数分解が容易になる場合については，3.2 章に述べた．本章では，e, d の設定により RSA 暗号関数が容易に解読される条件について述べる.

小さい e への攻撃　ランダムに e を生成すると，通常 e の大きさは，合成数 N の大きさ程度になる．しかし，暗号化や署名検証の処理を高速にするため，頻繁に e を小さく数ビット程度にとる場合が多い．しかし小さい e を利用したユーザへの同報通信，すなわち同じメッセージ M を暗号化して送る場合には，M が露呈する ([4])．この攻撃を回避するために，e は数 10 ビットの大きさを用いることが望ましい．

小さい d への攻撃　通常は，e は高速化の理由から小さくとるが，d を小さくとることはあまりない．しかし，何らかの生成ミスで d の値が小さくなった場合，攻撃されることが示されている ([2])．彼らの結果によると，$d < N^{0.292}$ となる d を用いると，RSA 暗号関数が解読可能になる．このため，d は可能な限り N の大きさと等しく生成することが望ましい．

3.3.2　RSA-OAEP 変換への攻撃

本章では，OAEP 変換により確保される安全性について記載する．一般に暗号の強度について議論する場合，攻撃アルゴリズムの手法及び秘匿のレベルの 2 つの観点から議論する．攻撃アルゴリズムの手法については，章と同じであるので省略し，ここでは最も強力な秘匿レベルである semantically secure(強秘匿性) について議論する．

まず，必要な定義を行う．有限ビット列から無限ビット列への写像全体の集合を，$\Omega = \{F|F : \{0,1\}^* \longrightarrow \{0,1\}^\infty\}$ とする．$\mathcal{G}$ は鍵生成関数，暗号化関数，復号化関数などを特定する暗号 generator とし，入力 1^k に対して，2 つの関数 G, H にアクセスする確率的暗号関数及び復号関数 $\mathcal{E}$ 及び $\mathcal{D}$ を出力する．攻撃アルゴリズム $\mathcal{A}$ は，find-stage 及び guess-stage をもち，find-stage $\mathcal{A}^{G,H}(\varepsilon, \mathtt{find})$ においては，暗号関数 $\mathcal{E}$ を利用し，平文 x_0, x_1 を出力し，そのどちらか一方に対する暗号文 c を得る．

Definition 2 $t, q_{gen}, q_{hash}, \varepsilon > 0$ とする．このとき，高々 t ステップで操作し，q_{gen} 回，q_{hash} 回の質問を G 及び H にする find-*stage* をもつ攻撃者 $\mathcal{A}$ に対して，

$$Pr\ \ [(\mathcal{E},\mathcal{D}) \leftarrow \mathcal{G}(1^k); G, H \leftarrow \Omega; (x_0, x_1, c) \leftarrow \mathcal{A}^{G,H}(\varepsilon, \mathtt{find}); b \leftarrow \{0,1\}; y \leftarrow \mathcal{E}^{G,H} : \mathcal{A}^{G,H}(y_0, x_0, x_1, c) = b] - \frac{1}{2} \geq \varepsilon$$

となるとき，$\mathcal{A}$ を $(t, q_{gen}, q_{hash}, \varepsilon)$-*break* $\mathcal{G}(1^k)$ であるという．ここで，Pr は，$(\mathcal{E},\mathcal{D}) \leftarrow \mathcal{G}(1^k)$, $G, H \leftarrow \Omega$, $(x_0, x_1, c) \leftarrow \mathcal{A}^{G,H}(\varepsilon, \mathtt{find})$, $b \leftarrow \{0,1\}$, $y \leftarrow \mathcal{E}^{G,H}$ となるとき，$\mathcal{A}^{G,H}(y_0, x_0, x_1, c) = b$ となる確率を表す．

OAEP が対象とする一方向性置換とは，以下のように定義される．正整数 k_0, k_1, n に対し，$k = n + k_0 + k1$ とし，置換関数 f

$$f : \{0,1\}^{n+k_1} \times \{0,1\}^{k_0} \longrightarrow \{0,1\}^{n+k_1} \times \{0,1\}^{k_0}$$

を考える．このとき，一方向性置換とは，以下のような関数である．

Definition 3 (one-way permutation(一方向性置換)) $\tau, \varepsilon > 0$ とする．このとき，高々 τ ステップで操作する任意の攻撃者 $\mathcal{A}$ に対して，

$$Pr[(s,t) \leftarrow \{0,1\}^{n+k_1} \times \{0,1\}^{k_0}; y \leftarrow f(s,t) : \mathcal{A}(y) = (s,t)] < \varepsilon$$

となるとき，f を (τ, ε)-*one-way permutation(一方向性置換)* であるという．ここで，Pr は，$(s,t) \leftarrow \{0,1\}^{n+k_1} \times \{0,1\}^{k_0}$ かつ $y \leftarrow f(s,t)$ のとき，$\mathcal{A}(y) = (s,t)$ となる確率を表す．

一方，OAEP で利用される型に特化した paritial-one-way permutation(部分一方向性置換) は以下のように定義される．

Definition 4 (partial-one-way permutation(部分一方向性置換)) $\tau, \varepsilon > 0$ とする．このとき，高々 τ ステップで操作する任意の攻撃者 $\mathcal{A}$ に対して，$\exists \varepsilon > 0$ *s.t.*

$$Pr[(s,t) \leftarrow \{0,1\}^{n+k_1} \times \{0,1\}^{k_0}; y \leftarrow f(s,t) : \mathcal{A}(y) = s] < \varepsilon$$

となるとき，f を (τ,ε)-*partial-one-way permutation(*部分一方向性置換*)* という．

one-way permutation f に OAEP 変換を施した f-OAEP は，以下の性質を満たす ([1]).

Theorem 1 ([1]) f が一方向性置換であれば，*OAEP* 変換を施した f-*OAEP* は，

- *Semantically secure(*適応的選択平文攻撃に対して強秘匿 *(IND-CPA))*
- *Plaintext awareness(*平文既知性 *(IND-CCA1))*

注意したいのは，定理 1 から f が一方向性置換というだけでは，最強の安全性である適応的選択暗号文攻撃において強秘匿 (IND-CCA2) は示せない．

しかしながら，f が部分一方向性置換であれば，OAEP 変換を施した f-OAEP は，以下の性質を満たすことが示されている ([3]).

Theorem 2 ([3]) f が部分一方向性置換であれば，*OAEP* 変換を施した f-*OAEP* は，*Semantically secure(*適応的選択暗号文攻撃に対して強秘匿 *(IND-CCA2))* である．

さらに，f が RSA 暗号関数の場合には，部分一方向性置換であることと一方向性置換であることは等価になる ([3]). よって，以下の結果がえられている．

Theorem 3 ([3]) *RSA* 暗号関数，$f(x) = x^e \pmod N$ が一方向性置換であるという仮定の元で，*RSA-OAEP* は適応的選択暗号文攻撃において強秘匿 *(IND-CCA2)* を満たす．

3.4　スキームとプリミティブに対する安全性評価

以上の結果から，RSA 暗号関数が一方向性関数であるようにパラメータ p, q, e, d を生成すると，適応的選択暗号文攻撃に対して強秘匿であり，非常に安全な方式といえる．なお，パラメータ設定の条件は下記のようになる．

- 素数 p 及び q の条件
  1. 2 つの素数 p, q において，$p \pm 1$ $q \pm 1$ が $B-$ スムースでない．
  2. n の素因数の大きさが $\sqrt{n}$ であること．
  3. 2 つの素数 p, q 差 $\triangle$ が大きいこと．
  4. $n = c^k \pm s$ (c, s: 小さな数) と表されないこと．
- e, d の条件
  1. e が数十ビット以上であること．
  2. $d > N^{0.292}$ であること．

3.5　まとめ

本章では，RSA-OAEP は，RSA 暗号関数が一方向性関数であるという仮定の下で，適応的選択平文攻撃において強秘匿 (IND-CCA2) であり，非常に安全であることを報告した．さらに，RSA 暗号関数が一方向性関数であるために必要な条件である適切なパラメータ設定についても報告した．

関連図書

[1] M. Bellare and P. Rogaway, "Optimal asymmetric encryption", *Advances in Cryptology-Proceedings of EUROCRYPT'94*, Lecture Notes in Computer Science, **950**(1995), Springer-Verlag, 92-111.

[2] D. Boneh and G. Durfee, "Cryptanalysis of RSA with private key d less than $N^{0.292}$" *Advances in Cryptology-Proceedings of EUROCRYPT'99*, Lecture Notes in Computer Science, **1592**(1999), Springer-Verlag, 1-11.

[3] E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern, "RSA-OAEP is secure under the RSA assumption", *The 2001 Symposium on Cryptography and Information Security*, **SCIS2001-8B-4**, Jan. 2000, 441-446.

[4] J. Hastad, "On using RSA with low exponent in a public key network", *Advances in Cryptology-Proceedings of Crypto'85*, Lecture Notes in Computer Science, **218**(1986), Springer-Verlag, 403-408.

第4章　Diffie-Hellman鍵共有方式の評価

要旨

本章ではDiffie-Hellman鍵共有方式[1]の紹介を行なう．この方式は，大別して予備通信を行なわない方式と行なう方式に分類できる．最初にそれぞれの方式について説明し，これらの方式の数学上の安全性について考察する．また，スキームとプリミティブの安全性について考察するため，攻撃の種類を受動的攻撃と能動的攻撃に分類する．次に，これらの攻撃の本質的な違いについて説明した後，能動的攻撃による攻撃の例を示す．最後に，この攻撃を防ぐためにユーザの認証機能が付加された方式について考察する．

abstract

In this chapter, we give a survey of Diffie-Hellman key exchange scheme[1]. This scheme can be classified into two types, such as non-interactive key exchange type and interactive one. At first, we explain those types and consider the mathematical underlying problem which those types have. Next, we classify the attack against key exchange scheme into two attacks(types), such as passive attack and active attack. Later, we explain about the substantial difference between them, and give an example for active attack. Finally, we consider the authenticated key exchange schemes which prevent the attack of the active attack.

4.1　プロトコルの紹介

Diffie-Hellman鍵共有方式(以下，DH鍵共有方式と略す)[1]は，大きく分けて予備通信を行なわない方式と予備通信行なう方式に分類できる．以下にそれぞれの方式を記述する．ここでは想定としてユーザAとユーザBが鍵を共有する場合を考える．

4.1.1　予備通信を行なわない方式

- **鍵生成:** 素数 p を生成し、乗法群 $\mathbb{Z}_p^*$ での元 g を求める．g の位数を q とする．(p,q,g) をシステム共通のパラメータとして公開する．
- **ユーザ加入:** A は乱数 $a \in \mathbb{Z}_q$ を生成し，$y_A = g^a \pmod p$ を計算する．同様に B は乱数 $b \in \mathbb{Z}_q$ を生成し，$y_B = g^b \pmod p$ を計算する．ここで，A の公開鍵は y_A，秘密鍵は a．また B の公開鍵は y_B，秘密鍵は b である．
- **鍵共有:** 以下のステップを実行する．

  Step1 A は $K_{AB} = (y_B)^a \pmod p$ を計算する．

  Step2 B は $K_{BA} = (y_A)^b \pmod p$ を計算する．

鍵共有段階では，

$$K_{AB} = (y_B)^a = (g^b)^a = (g^a)^b = (y_A)^b = K_{BA} \pmod p$$

より，$K_{AB} = K_{BA}$ となり，AB 間の鍵共有が可能となる．

4.1.2　予備通信を行なう方式

- **鍵生成:** 予備通信を行なわない方式の鍵生成と同様である.
- **鍵共有:** 以下のステップを実行する.

  Step1 A は乱数 $r_A \in \mathbb{Z}_q$ を生成し，$x_A = g^{r_A} \pmod p$ を計算する．A は x_A を B に送信する.

  Step2 B は乱数 $r_B \in \mathbb{Z}_q$ を生成し，$x_B = g^{r_B} \pmod p$ を計算する．B は x_B を A に送信する.

  Step3 A は $K_{AB} = (x_B)^{r_A} \pmod p$ を計算する.

  Step4 B は $K_{BA} = (y_A)^{r_B} \pmod p$ を計算する.

鍵共有段階では,

$$K_{AB} = (x_B)^{r_A} = (g^{r_B})^{r_A} = (g^{r_A})^{r_B} = (x_A)^{r_B} = K_{BA} \pmod p$$

より，$K_{AB} = K_{BA}$ となり，AB 間の鍵共有が可能となる.

4.2　プリミティブ (離散対数問題) の攻撃法について

DH 鍵共有方式の安全性は，離散対数問題に基づく．ただし現時点で散対数問題との等価性は現在のところ示されていない．すなわち，離散対数問題を多項式時間で解くアルゴリズムが発見された場合，DH 鍵共有方式を破ることが可能であるが，その逆が成り立つかどうかはわかっていない．離散対数問題の攻撃に関する議論については，2 章に記述されている.

4.3　各攻撃法に対するプリミティブの安全性結果

前章で述べられた考察より，DH 鍵共有方式は離散対数問題が解けるようなパラメータを設定してしまうと，容易に破ることができる．このような攻撃を防止するためのパラメータ設定に関しては，2 章に記述されている.

4.4　スキームとプリミティブの組合せに対し評価された攻撃法

4.4.1　帰着関係

DH 鍵共有方式の安全性について考察するため，以下に示すような Diffie-Hellman 問題を定義する.

- **Diffie-Hellman 問題:** Diffie-Hellman 問題とは，素数 p, $g \in \mathbb{Z}_p^*$, $y_A \in \mathbb{Z}_p^*$, $y_B \in \mathbb{Z}_p^*$ が与えられたとき，$K = g^{ab} \pmod p$, $y_A = g^a \pmod p$, $y_B = g^b \pmod p$, $a \in \mathbb{Z}_p^*$, $b \in \mathbb{Z}_p^*$ なる $K \in \mathbb{Z}_p^*$ が存在するならば、K を計算する問題である.

Diffie-Hellman 問題は離散対数問題に帰着する．ただし Diffie-Hellman 問題と離散対数問題の等価性は現在のところ示されていない．すなわち，離散対数問題を多項式時間で解くアルゴリズムが発見された場合，Diffie-Hellman 問題を多項式時間で解くことが可能であるが，その逆が成り立つかどうかはわかっていない.

4.4.2　プロトコルの安全性

攻撃者にどのような攻撃を許すかという観点から，受動的攻撃と能動的攻撃に分類することができる.

	DH 鍵共有方式の種類	
	予備通信なし	予備通信あり
能動的攻撃の可能性	無	有
能動的攻撃の具体例	–	中間侵入攻撃
回避方法	–	認証機能の付加

表 4.1: 能動的攻撃の可能性とその回避方法

- **受動的攻撃:** 攻撃者はユーザの公開鍵など，公開された情報のみを用いて共有鍵を計算する．特に予備通信を行なう方式の場合，公開情報としてユーザ同士がやりとりする通信系列の情報が含まれる．
- **能動的攻撃:** 攻撃者は自ら生成した値をユーザに送信したり，通信間に流れる情報を奪う (抜きとる) 攻撃が許される．この攻撃の後，攻撃者は公開された情報を用いて共有鍵を計算する．

上記の攻撃方法に関する安全性の考察を，予備通信を行なわない方式と行なう方式に分けて説明する．

予備通信を行なわない場合

ユーザ間の予備通信を行なわないため，通信経路上に情報は流れない．よって能動的攻撃は存在せず，受動的攻撃のみが存在する．このとき，各ユーザの公開鍵を基にして，共有鍵を計算しなければならない．これは Diffie-Hellman 問題を解くことと同等である．このときの安全性は離散対数問題に基づく．

予備通信を行なう場合

受動的攻撃の安全性は，Diffie-Hellman 問題に帰着する．能動的攻撃の代表的なものとしては，中間侵入 (intruder in the middle) 攻撃があげられる．これは，攻撃者が AB 間の通信系列でやりとりされる情報を途中で奪い，攻撃者自身が生成した情報と取り替えることによって共有鍵を計算する方式である．以下に攻撃者 (C とする) は B のなりすましをして AC 間の共有鍵を計算する手順を示す．

- **中間侵入攻撃:** 以下のステップを実行する．

  Step1 A は乱数 $r_A \in \mathbb{Z}_q$ を生成し，$x_A = g^{r_A} \pmod p$ を計算する．A は x_A を B に向けて送信する．

  Step2 C は B に送信されるはずの x_A を抜きとる．C は乱数 $r_C \in \mathbb{Z}_q$ を生成し，$x_C = g^{r_C} \pmod p$ を計算する．C は x_C を C に送信する．

  Step3 A は $K_{AC} = x_C^{r_A} \pmod p$ を計算する．

  Step4 B は $K_{CA} = x_A^{r_C} \pmod p$ を計算する．

ここで，

$$K_{AC} = (x_C)^{r_A} = (g^{r_C})^{r_A} = (g^{r_A})^{r_C} = (x_A)^{r_C} = K_{CA} \pmod p$$

より，$K_{AC} = K_{CA}$ となり，AC 間の鍵共有が可能となり，C は B のなりすましに成功したことになる．

4.4.3　認証機能が付加された方式

中間侵入攻撃に代表されるような能動的攻撃を防止するためは，送られてきた値が正規のユーザによって生成されたかどうか認証を行えばよい (表 4.1 参照)．以下ではこのような認証機能が付加された方式を

述べる．具体的には，ユーザが生成した値に署名をを付加し，各ユーザはその署名を確認することにより，能動的攻撃を回避する方法を考える．ここで，sig_y，ver_y は，それぞれ公開鍵 y による署名あるいは検証関数とする．

- **鍵生成:** 予備通信を行なわない方式の鍵生成と同様である．
- **鍵共有:** 以下のステップを実行する．

  Step1 A は乱数 $r_A \in \mathbb{Z}_q$ を生成し，sig_{y_A} を用いて，r_A に対する署名 $Sign_{AB} = sig_{y_A}(r_A)$ を作成する．A は $(r_A, Sign_{AB})$ を B に送信する．

  Step2 B は乱数 $r_B \in \mathbb{Z}_q$ を生成し，sig_{y_B} を用いて，r_B に対する署名 $Sign_{BA} = sig_{y_B}(r_B)$ を作成する．B は $(r_B, Sign_{BA})$ を A に送信する．

  Step3 A は ver_{y_B} を用いて，r_B が B によって生成された値かどうか検証する．もし，正しいと判断されたならば，A は $K_{AB} = (r_B)^{r_A} \pmod p$ を計算する．

  Step4 B は ver_{y_A} を用いて，r_A が A によって生成された値かどうか検証する．もし，正しいと判断されたならば，A は $K_{BA} = (r_A)^{r_B} \pmod p$ を計算する．

鍵共有段階では，

$$K_{AB} = (x_B)^{r_A} = (g^{r_B})^{r_A} = (g^{r_A})^{r_B} = (x_A)^{r_B} = K_{BA} \pmod p$$

より，$K_{AB} = K_{BA}$ となり，AB 間の鍵共有が可能となる．

4.5　まとめ

DH 鍵共有方式に関する評価を行なった．4.4.3 章で記述されているような，公開鍵証明証の作成方法は，X.509 によって規格化されている．認証機能が付加された方式は，前章で述べられた方式以外に，認証局 (CA, Certificated Authority) によって公開鍵の認証が付加された鍵共有方式がある．鍵共有の仕組みが DH 鍵共有に基づいた方式の中で，代表的なものとしては認証機能が離散対数問題に基づく MTI 方式 [3] や素因数分解問題に基づく Giraut 方式 [2] があげられる．

関連図書

[1] W. Diffie and M. E. Hellman, "New directions in cryptography", *IEEE Trans. Information Theory*, vol. IT-22, pp.644-654, 1976.

[2] M. Giraut, "Self-certified public keys", *Eurocrypt '91*, LNCS No. 547, pp.490-497, 1992.

[3] T. Matsumoto, Y. Takashima and H. Imai, "On seeking smart public-key distribution systems", *Transactions of the IECE(Japan)*, 69, pp.99-106, 1986.

第5章　DSA署名の評価

要旨

本章では，DSA 署名方式の紹介を行う．DSA 署名は，アメリカ連邦標準技術局 (NIST) により提案されたデジタル署名標準 (DSS)[1] であり，ElGamal 署名方式を改良したものである．まず，DSA 署名のプロトコルの詳細を紹介する．次に，DSA 署名のプリミティブ（離散対数問題）に対する安全性及びスキーム（DSA 署名）に対する安全性について記述する．DSA 署名の安全性に関しては，離散対数問題が難しいなら，ハッシュ関数がランダム関数である仮定の下で適応的選択平文攻撃に対して安全であることを報告する．

abstract

In this chapter, we survey the digital signature algorithm(DSA), which is proposed by National Institute of Standards and Technology(NIST)[1]. DSA is a modification of the ElGamal Signature Scheme. First we give the protocol of DSA. Next we discuss the security on DSA in view of both the primitive basis like DLP and the scheme like DSA. We report that DSA is secure against the adaptive chosen message attack under the assumption that a hash function is a random function, and that the discrete logarithm problem is infeasible.

5.1　プロトコルの紹介

DSA 署名方式は ElGamal 署名方式を改良したものである．この改良で g の位数を小さくする手法により，署名サイズおよび処理量が小さくなっている．つまり，原始元 (位数 $p-1$) を生成元 $g(q|p-1$ を満たす位数 q) に置き換えることにより，乗法群 $\mathbb{Z}_p^*$ の中の g が生成する部分群上での演算に限定することができる．その結果，署名サイズおよび処理量を小さくできる．以下，プロトコルを記述する．

DSA 署名のアルゴリズム　平文 m に対する DSA 署名を構成する

Step 1. **(鍵生成):** ユーザは，$\mathbb{Z}_p^*$ において離散対数問題が難しいような素数 p と素数位数 $q(q|p-1)$ となるような $g \in \mathbb{Z}_p^*$ を生成する．ユーザは秘密鍵として $x \in \mathbb{Z}_q$ を選び，$y \equiv g^x (\mathrm{mod}\ p)$ を計算する．p, q, g をシステムパラメータとする．

Step 2. **(署名生成):** 乱数 $k \in \mathbb{Z}_q^*$ を生成し，$r_1 \equiv g^k (\mathrm{mod}\ p)$ を計算する．続いて $r_2 \equiv r_1 (\mathrm{mod}\ q)$ を計算する．また，$S \equiv k^{-1}(h(m, r_2) + xr_2) (\mathrm{mod}\ q)$ を計算する．以上により，署名 (r_2, S) を得る．

Step 3. **(署名検証):** まず，$r' \equiv g^{s^{-1}h(m,r_2)} y^{s^{-1}r_2} (\mathrm{mod}\ p)$ を計算する．次に，$r_2 \equiv r' (\mathrm{mod}\ q)$ をチェックし，成り立てば検証成功である．

5.2　プリミティブに対して評価されている攻撃法

DSA 署名で使われる秘密鍵の安全性は離散対数問題に基づく。つまり，離散対数問題を容易にする素数 p や生成元 g を選択すると，DSA 署名は秘密鍵が洩れてしまう。離散対数問題に対する攻撃法に関する議論については，他の ElGamal タイプの署名方式と全く同等の議論が成立する．詳しくは 2 章を参照されたい．

5.2.1　プリミティブの攻撃法に対する安全性結果

離散対数問題を容易にする素数 p や位数 q となるような生成元 g を選択すると，DSA 署名を偽造できる。このような直接攻撃を回避するために，2 章を満たすような p, q, g を生成する必要がある．

5.3　スキーム (DSA 署名方式) への攻撃法

署名方式に対する攻撃法は，2 つの観点から考えることができる．1 つはどのように偽造できるか (偽造の度合) という観点であり，もう 1 つは攻撃者にどのような攻撃 (攻撃の強度) を許すかという観点である．

5.3.1　偽造の種類

署名の偽造は，偽造できる度合で次の 4 つが存在する．

- 全面的解読：秘密鍵が計算できる．任意の平文に対する偽造が可能になる．
- 一般的偽造：署名アルゴリズムと機能的に等価なアルゴリズムが構成できる．任意の平文に対する偽造が可能になる．
- 選択的偽造：攻撃者があらかじめ選んだ平文に対する偽造が可能になる．
- 存在的偽造：ある平文に対する偽造が可能になる．ただし，平文を攻撃者が操作できない．

5.3.2　攻撃の種類

署名方式に対する攻撃は大きく受動的攻撃と能動的攻撃に分けられる．これらの攻撃は強度によって次の 4 つが存在する．

1. 受動的攻撃：公開鍵などの公開情報のみを利用する攻撃
   (a) 直接攻撃：公開鍵のみを利用する攻撃
   (b) 既知平文攻撃：平文の集合 $\{m_1, \cdots, m_n\}$ に対する署名 $\{S_1, \cdots, S_n\}$ を利用する攻撃．ただし，この平文の組を選ぶことはできない．
2. 能動的攻撃：攻撃のための情報を署名者から入手でき，それを利用する攻撃
   (a) 一般選択平文攻撃：攻撃者は入手したい署名の平文の組 $\{m_1, \cdots, m_n\}$ を選択し，その平文の署名 $\{S_1, \cdots, S_n\}$ を入手することができる攻撃．ただし，平文の組は署名文をみる前に固定される．
   (b) 適応的選択平文攻撃：攻撃者は入手したい平文の署名を署名者の署名結果をみながら適応的に選択し，入手することができる攻撃．

5.4　攻撃法に対する安全性判定

デジタル署名の安全性は、第三者が利用する偽造の度合および攻撃の強度により判断する．最も弱い度合の偽造は存在的偽造であり，署名方式が存在的偽造すら不可能ならばその他全ての偽造が不可能になる．また，攻撃者にとって最も有利に展開できる攻撃は適応的選択平文攻撃であり，署名方式が適応的選択平文攻撃によって偽造されなければ，その他全ての攻撃に対しても偽造されない．署名方式が適応的選択平文攻撃で存在的偽造不可であるとき，その署名方式は安全であるという．

Theorem 4 適応的選択平文攻撃において，もし無視できない確率で *DSA* 署名の存在的偽造に成功するなら，離散対数問題が多項式時間で解ける．

上記定理の対偶は,「適応的選択平文攻撃において，もし離散対数問題を多項式時間で解くことが難しいなら，DSA 署名の存在的偽造に成功しない.」である．実際，離散対数問題を多項式時間で解くことが難しいので，この定理が証明されると DSA 署名方式は適応的選択平文攻撃に対していかなる偽造もできないことことがいえる．つまり，離散対数問題が難しいなら DSA 署名方式は安全である．ただし，プロトコルで用いるハッシュ関数が真の乱数を生成することを想定した場合に限る．安全性の証明では，ハッシュ関数をランダムオラクルモデルと想定し，プロトコルに用いられる $h(m, r_2)$ は m, r_2 と何ら相関を持たない乱数であることを仮定している．

5.5　まとめ

DSA 署名のプロトコルの紹介とそれに対する攻撃の種類及び偽造の種類をまとめた．そして，離散対数問題を多項式時間で解くことが難しいなら，DSA 署名方式が適応的選択平文攻撃に対して存在的偽造すらできないことを言及した．

また，DSA 署名の原型ともいえる ElGamal 署名に対する攻撃に Bleichenbacher の攻撃 [3] がある．この攻撃は g の位数を小さい素因数に分解できる場合にそこを狙って攻撃する方法である．しかしながら，DSA 署名ではパラメータの設定で g の位数を素数位数 q にしているので，この攻撃は受けない。

関連図書

[1] National Institute of Standards and Technology (NIST). *FIPS Publication 186:Digital Signature Standard*, May 19, 1996.

[2] D. Pointcheval and J. Stern. Security Proofs for Signature Schemes. In *Advances in Cryptology – EUROCRYPT'96*, pages 387–398, 1996.

[3] D. Bleichenbacher. Generating ElGamal Signature Without Knowing the Secret Key. In *Advances in Cryptology – EUROCRYPT'96*, pages 10–18, 1996.