IMES DISCUSSION PAPER SERIES

RSA署名方式の安全性を巡る研究動向について

さいとう　まゆみ
齊藤　真弓

Discussion Paper No. 2002-J-14

IMES
INSTITUTE FOR MONETARY AND ECONOMIC STUDIES
BANK OF JAPAN

日本銀行金融研究所

〒103-8660 日本橋郵便局私書箱 30 号

IMES Discussion Paper Series 2002-J-14
2002 年 3 月

RSA 署名方式の安全性を巡る研究動向について

齊藤（さいとう） 真弓（まゆみ）*

要　旨

インターネット等オープンなネットワーク上において安全なデータ交信を行う手段の 1 つとして、デジタル署名方式が活用されつつある。デジタル署名方式は、公開鍵暗号技術を利用してデータ通信者の本人確認や交信データの一貫性確認を可能とするものであり、現在、事実上の標準として幅広く利用されている署名方式は RSA 署名方式である。

RSA 署名方式は、1978 年に提案されて以来、その安全な利用方法に関する研究が続けられてきた。RSA 署名方式に対する主な攻撃としては、①公開鍵の素因数分解によって秘密鍵を導出するタイプと、②秘密鍵を導出することなく署名偽造を行うタイプの 2 つが挙げられる。特に、上記②の攻撃を想定した場合、メッセージをそのまま秘密鍵によって変換し署名を生成するという方法は安全ではなく、メッセージをハッシュ関数等で変換した上で、秘密鍵によって署名を生成する必要があることが知られている。

こうした研究成果を踏まえ、RSA 署名方式を安全に利用するためにはメッセージにどのような変換を施せば良いかに関する研究が行われている。近年では、一定の数学的仮定の下で安全性が証明可能であり、かつ、高い実用性を有する RSA 署名方式の利用方法が提案されている。それらの中で、現在最も注目を集めているのが RSA-PSS 署名方式である。RSA-PSS 署名方式は、国際標準や業界標準等への採用が検討されており、今後幅広い分野で利用されるようになることも考えられる。

本稿では、RSA 署名方式から RSA-PSS 署名方式に至る研究動向について、署名変換データの生成方法を中心に安全性の観点から説明した上で、RSA-PSS 署名方式を巡る国際標準化動向等について紹介する。

キーワード：デジタル署名方式、RSA 署名方式、RSA-PSS 署名方式、素因数分解、安全性証明

JEL Classification : L86, L96, Z00

*日本銀行金融研究所研究第 2 課（E-mail:mayumi.saitou@boj.or.jp）

本論文は、2002 年 2 月 28 日に日本銀行で開催された「第 4 回情報セキュリティ・シンポジウム」への提出論文に加筆・修正を施したものである。

目　次

頁

1．はじめに

インターネット等オープンなネットワーク上において安全にデータ交信を行うための手段の 1 つとして、公開鍵暗号技術を利用してデータ通信者の本人確認や交信データの一貫性確認を可能とするデジタル署名方式が活用されてきている。わが国においては、2001 年 4 月に「電子署名及び認証業務に関する法律」（以下、電子署名法という）が施行され、一定の条件の下でデジタル署名が手書き署名や押印と同等の法的効力を持つようになった。今後、電子金融取引をはじめ様々な分野で、デジタル署名方式が活用されていくことが予想される。

現在、デジタル署名方式の「事実上の標準」として広く利用されているのが、RSA 署名方式である。RSA 署名方式は、公開鍵暗号・デジタル署名方式の業界標準 PKCS#1、デジタル署名方式の国際標準 ISO 14888 および ISO 9796、米国連邦政府標準 FIPS 186-2、金融業務に利用される公開鍵暗号方式の米国国内標準 ANS X9.57 等にも規定されている。また、わが国の電子署名法が定める「特定認証業務の認定制度」においても、認定の対象となる認証業務で採用される 4 種類のデジタル署名方式の 1 つとして RSA 署名方式が指定されており、本制度に基づく RSA 署名方式の利用が開始されている。

RSA 署名方式は、1978 年に Rivest、Shamir、Adleman によって提案されて以来、その安全な利用方法に関する研究が盛んに行われてきた。RSA 署名方式の安全性に対する最大の脅威は、署名生成者の意図しない文書に対する署名が偽造されてしまうことである。RSA 署名方式の署名偽造に対する安全性の研究においては、①公開鍵の素因数分解によって秘密鍵を求め、RSA 関数の逆関数を導出して署名偽造を行うタイプ、②RSA 関数の逆関数を導出することなく署名偽造を行うタイプ、の 2 種類の攻撃に主として焦点が当てられてきた。上記①に関しては、より高速な素因数分解アルゴリズムの研究や、十分な安全性を確保するために必要な鍵長に関する研究が進められてきた。この結果、現時点での最高速の素因数分解アルゴリズムである数体ふるい法を前提として、商用で RSA 署名方式を用いる場合には公開鍵のサイズを 1024 bit 以上に設定することが推奨されるケースが多い。他方、上記②のタイプの攻撃を想定した場合には、メッセージをそのまま署名変換データとして利用するのでは安全とはいえず、メッセージにハッシュ関数やパディングを施したものを署名変換データに利用して署名を生成する必要があることが知られている。こうした RSA 署名方式の署名変換データの生成方法については、これまで様々な方法が提案されてきたが、中でも、1996 年に Bellare と Rogaway によって提案された RSA-PSS 署名方式が注目されている。

RSA-PSS 署名方式の特徴は、「一定の仮定の下でその安全性が証明可能である」という証明可能安全性と、通常の RSA 署名方式と同程度の実用性を兼ね備えている点である。このため、証明可能安全性を有していない他の利用方法に比べて安全性の観点から望ましく、ISO 9796-2 の改訂案に盛り込まれることが検討されているほか、PKCS#1 の最新バージョン(Ver2.1)にも追加された。さらに、CRYPTREC や NESSIE といった暗号アルゴリズム評価プロジェクトにおいても、有力なデジタル署名方式の 1 つとして評価が進められている。こうした動きをみると、今後、幅広い分野において、RSA-PSS 署名方式が利用されるようになることも考えられる。ただし、RSA-PSS 署名方式においては、処理速度向上等を企図したアルゴリズムのマイナーな見直しが 2 度行われており、今後もそうした改訂が行われる可能性もある点には留意が必要である。

本稿の構成は以下のとおりである。まず、2. において、RSA 署名方式の概要を紹介した上で、主な署名偽造攻撃のタイプとして、①公開鍵の素因数分解を行って秘密鍵を求め、RSA 関数の逆関数を導出して署名を偽造するタイプと、②RSA 関数の逆関数を導出することなく署名を偽造するタイプ、の 2 つがあることを説明する。3. では、これまで提案されてきた主な素因数分解アルゴリズムの概要を紹介するとともに、十分な安全性を確保するために必要な鍵長に関する最新の研究動向を紹介する。4. では、RSA 関数の逆関数を導出することなく署名を偽造する攻撃に焦点を当て、RSA-PSS 署名方式の提案に至る主な研究の流れを紹介する。5. では、RSA-PSS 署名方式の概要や 3 種類のバージョンの差異について説明するほか、RSA-PSS 署名方式に関する最新の国際標準化動向等について紹介する。6. は、本稿の結びである。

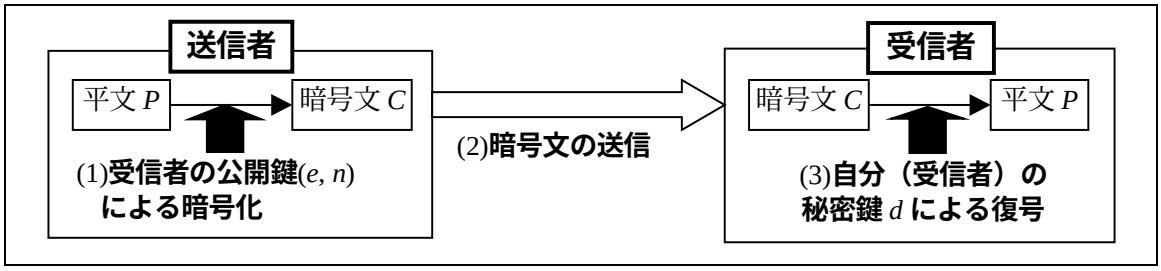
2. RSA 署名方式とは

(1)RSA 暗号／署名方式のアルゴリズム

RSA 暗号方式は、1978 年に Rivest、Shamir、Adleman によって考案された最初の本格的な公開鍵暗号方式である（Rivest, Shamir, and Adleman［1978］）。RSA 暗号方式は、暗号化用の鍵と復号用の鍵が異なり、一方の鍵から他方の鍵を求めることが計算量的に困難である[1]ため、どちらかの鍵を公開することができるという性質を持つ。通常、公開する鍵（公開鍵）を暗号化に利用し、秘密に保管される鍵（秘密鍵）を復号に利用する。

RSA 暗号方式における公開鍵と秘密鍵の生成は、①2 つの大きな素数 p と q を選び、これらの積 $n = p \cdot q$ を計算する、②$p-1$ と $q-1$ の最小公倍数 L を算出する、③L と互いに素な自然数 e を選ぶ、④$e \cdot d = 1 \pmod{L}$を満たす d を求める、という手順で行われる。この結果、得られた(e, n)を公開鍵、d を秘密鍵とする。これらの鍵ペアを生成した後、鍵ペアの所有者は、PKI（Public Key Infrastructure）[2]等を用いて公開鍵(e, n)が確かに自分の公開鍵であることを第三者に対して示すとともに、秘密鍵 d を第三者に知られないように保管しておく。

図 1：RSA 暗号方式による暗号通信の例



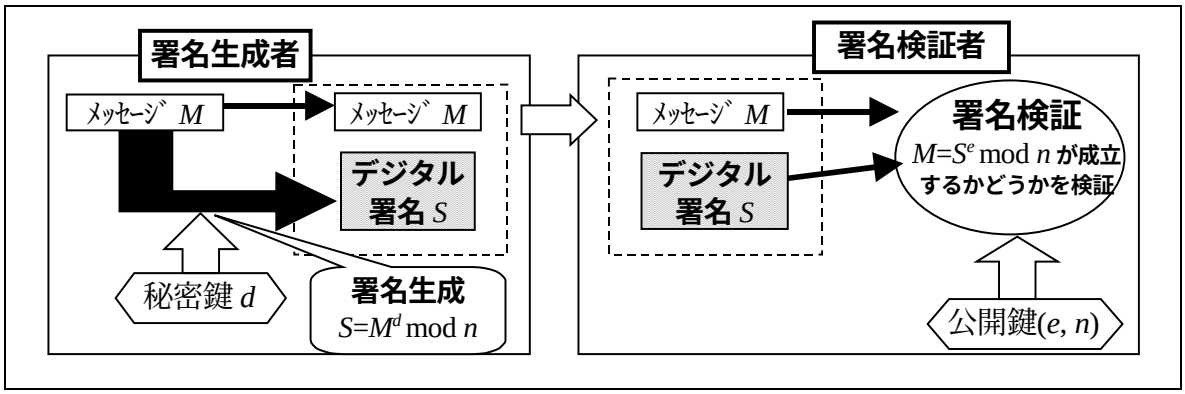
RSA 暗号方式を利用した暗号通信（図 1 参照）では、まず、送信者が平文 P と受信者の公開鍵(e, n)を用いて $C = P^e \bmod n$ を計算し、暗号文 C を作成する。暗号文 C を受信した受信者は、自分の秘密鍵 d を用いて $P = C^d \bmod n$ を計算し、平文 P を入手する。

---

[1] 「計算量的に困難である」とは、その計算を行うことは理論的には可能であるものの、実際にその計算を実行するには計算量が非常に大量となり、膨大な費用と時間を必要とすることから、事実上不可能であることを意味する。どの程度の計算量が事実上不可能であるかは、その時々の技術条件等によって左右される。

[2] PKI は、公開鍵暗号における公開鍵・秘密鍵ペアとその持ち主を第三者に対して検証可能な形で結び付けるための仕組みである。PKI では、認証機関と呼ばれる第三者機関が各公開鍵に対して公開鍵証明書を発行し、公開鍵を利用する際に、その公開鍵証明書を検証して、公開鍵の所有者および有効性の確認を行う。PKI に関する技術や最近の動向については、宇根[2002]を参照。

図 2：RSA 署名方式の署名生成と署名検証



他方、RSA 暗号方式は、デジタル署名方式（RSA 署名方式）としても利用できる。あるメッセージ M に対して生成されるデジタル署名 S は、S の生成者（署名生成者）の確認、および、S が生成された後に M が改竄されていないかどうかの確認を可能とする。RSA 暗号方式をデジタル署名方式として利用する場合（図 2 参照）、まず、署名生成者はメッセージ M と自分の秘密鍵 d を用いて $S = M^d \bmod n$ を計算し、M に対するデジタル署名 S を生成する（署名生成）。デジタル署名 S とメッセージ M を受け取った署名検証者は、署名生成者の公開鍵(e, n)を用いて $M = S^e \bmod n$ が成立するかどうかを検証することによって、①デジタル署名 S の署名生成者が公開鍵(e, n)に対応する秘密鍵の持主であることと、②M が改竄されていないこと、を確認する（署名検証）。このように、RSA 署名方式では、秘密鍵 d が署名生成鍵として用いられ、公開鍵(e, n)が署名検証鍵として用いられる。

ここで、RSA 関数 f を $f(M) = M^e \bmod n$（ただし、(e, n)は RSA 暗号方式における公開鍵）と定義すると、RSA 暗号方式の暗号化・復号、および RSA 署名方式の署名生成・検証は、表 1 のように整理することができる。なお、RSA 関数 f の逆関数 f^{-1} は、$f^{-1}(M) = M^d \bmod n$ と表わされる。

表 1：RSA 暗号方式と RSA 署名方式の利用方法

RSA 暗号方式	暗号化	送信者は、RSA 関数 f によって平文 P を変換し、暗号文 $C = f(P) = P^e \bmod n$ を作成する。
	復号	受信者は、RSA 関数の逆関数 f^{-1} により暗号文 C を復号し、平文 $P = f^{-1}(C) = C^d \bmod n$ を入手する。
RSA 署名方式	署名生成	送信者は、データ M を RSA の関数の逆関数 f^{-1} で変換して署名 $S = f^{-1}(M) = M^d \bmod n$ を作成する。
	署名検証	受信者は、署名 S を RSA 関数 f によって変換し、変換結果のデータ $f(S) = S^e \bmod n$ と受信データ M が一致するか否かを確認して署名 S を検証する。

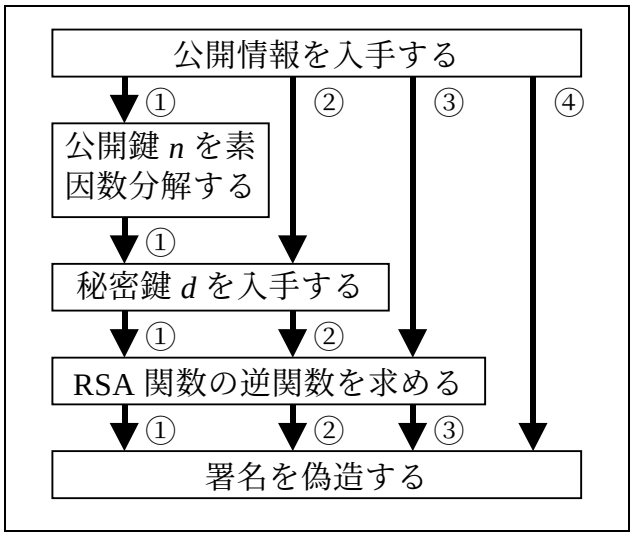
(2)RSA 署名方式の安全性を評価する枠組み

RSA 署名方式の安全性について説明する場合、「RSA 署名方式の安全性は、素因数分解問題の困難性に依拠している」といった表現が用いられることが多い。この表現は正確といえるのだろうか。この点を検証するために、最初に「RSA 署名方式の安全性とは何か」について考える。まず、「RSA 署名方式が安全である」とは、「公開された情報（公開鍵や既存のデジタル署名等）を用いて RSA 署名方式における署名を偽造することが計算量的に困難である」ことを意味することとする。ここで、署名の偽造とは、「『自分以外の者』が生成したと認められる（その『自分以外の者』の公開鍵による署名検証が成立する）署名を、その『自分以外の者』の協力を得ることなく生成すること」と定義する。

なお、本稿では RSA 署名方式自体の安全性に着目し、秘密鍵の管理等の運用上の問題については検討の対象外とする。すなわち、秘密鍵 d を格納したハードウェアから攻撃者が物理的手段によって秘密鍵を読み出すといった手段による署名の偽造は取り上げない。

RSA 署名方式における署名偽造の手順として、理論的にどのようなものが想定可能かを考えてみよう（図 3 参照）。通常想定されるルートとして、公開鍵 n を素因数分解して秘密鍵 d を求め、RSA 関数の逆関数を導出し、署名を偽造するということが、まず考えられる（図 3 の①に相当）。仮に署名の偽造を行うルートとして、このルートしか存在しないのならば、「公開鍵 n の素因数分解を行うことが計算量的に困難であるならば、署名の偽造が困難である」ということができる。しかし、理論的に考えると、n の素因数分解を行うことなく秘密鍵 d を推定し、それを用いて RSA 関数の逆関数を導出し、最終的に署名の偽造を行うルート（図 3 の②に相当）、秘密鍵 d を求めることなく、RSA 関数の逆関数を導出し、それを用いて署名の偽造を行うルート（図 3 の③に相当）、および RSA 関数の逆関数を求めることなく、署名の偽造を行うルート（図 3 の④に相当）を想定することが可能である。もちろん、秘密鍵 d を入手し、RSA 関数の逆関数を導出することなく署名の偽造を行うルートも考えられなくはないが、秘密鍵 d から RSA 関数の逆関数を求めることは容易であるため、あえてそのようなルートを考える必要はない。このように考えると、公開情報から署名の偽造を行う方法は図 3 の①～④のルートに分類される。以下では、RSA 署名方式の安全性をこれらのルートに当てはめながら検討する。

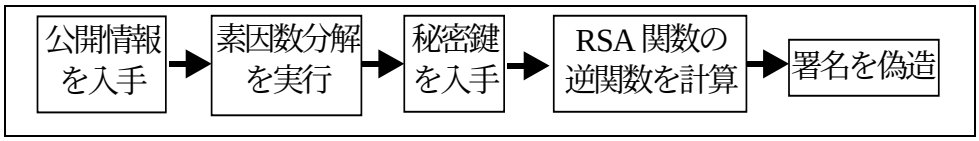
図 3：署名を偽造する 4 つのルート



イ．公開鍵 *n* の素因数分解からはじまるルート

まず、最もオーソドックスと考えられる「公開鍵 *n* を素因数分解するルート」（図 4 参照）については、素因数分解が計算量的に困難であれば、署名の偽造は困難になる。逆に、*n* の素因数分解が容易な場合には、その結果を用いて秘密鍵 *d* を導出し、RSA 関数の逆関数を求めて任意のデータに対する署名を偽造することが可能となる。このため、このルートによる攻撃のみが行われるという前提の下では、素因数分解の困難性は RSA 署名の安全性のための必要十分条件となる。

図 4：公開鍵 *n* を素因数分解するルート

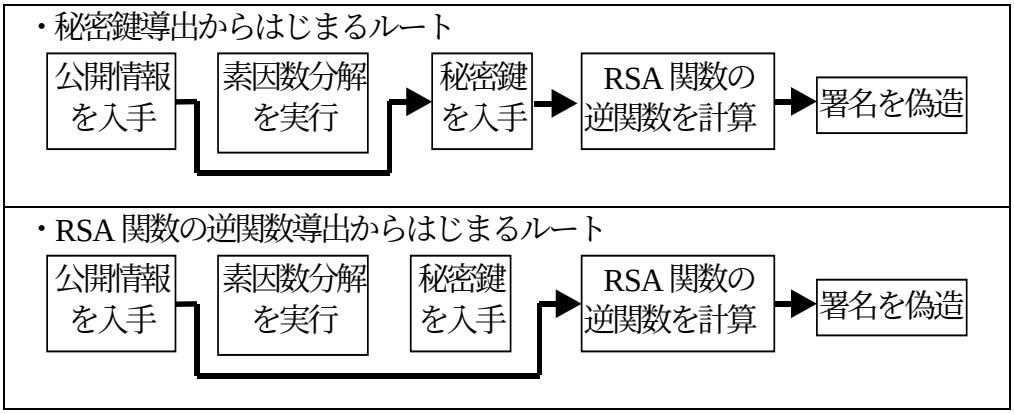


ロ．秘密鍵や RSA 関数の逆関数からはじまるルート

次に、*n* の素因数分解を行わずに、秘密鍵や RSA 関数の逆関数を導出するルート（②、③）について考える（図 5 参照）。こうしたルートによる攻撃の可能性については、RSA 暗号／署名方式が考案されて以来、様々な研究が行われてきたが、これまでのところ、*n* の素因数分解を行うことなく秘密鍵 *d* および RSA 関数の逆関数を導出するという攻撃、または *n* の素因数分解も秘密鍵 *d* の導出も行うことなく RSA 関数の逆関数を導出するという攻撃が成功したという事例

は報告されていないようである。このため、現在では、これらのルートによる有効な攻撃法は存在しないと想定するケースが多い。例えば、最近の公開鍵暗号方式の安全性に関する論文では、素因数分解の困難性を仮定する代りに、RSA 関数の逆関数導出の困難性の仮定（「RSA 関数の一方向性」の仮定と呼ばれる）を置いて議論を進めることが広く行われていることからも、こうした理解が広がっていることが分かる。

図 5：素因数分解を行わずに秘密鍵を導出するルートと RSA 関数の逆関数導出からはじまるルート



ハ．RSA 関数の逆関数を導出することなく署名を偽造するルート

上記ロ．において説明したルートが存在しないと想定して差し支えないのであれば、「RSA 署名方式の安全性は素因数分解の困難性に依拠している」という説明でも問題ないように思われるかもしれない。しかし、RSA 関数の逆関数を導出することなく、公開情報を利用して署名を偽造する攻撃法（図 6 参照）が存在することが知られており、素因数分解の困難性を仮定しても、無条件で RSA 署名方式が安全とはいえない。このルートによる具体的な攻撃法としては、積攻撃、Desmedt-Odlyzko 攻撃、CNS 攻撃等、攻撃者が予め指定したいくつかの署名変換データに対応する署名を署名生成者から入手することができることを前提とした攻撃が挙げられる。これらについては、4. において詳述することとする。

図 6：RSA 関数の逆関数を導出することなく署名を偽造するルート

このように、RSA 署名方式に対する攻撃法とその安全性の関係は、①公開鍵 n の素因数分解からはじまるルートでは、素因数分解の困難性が RSA 署名方式の安全性の必要十分条件となる、②RSA 関数の逆関数を導出することなく署名偽造を行うルートでは、素因数分解の困難性とは無関係に署名を偽造する方法が存在し、別途対策が必要となる、という 2 点に整理することができる。以下では、これらに沿って、RSA 署名方式の安全性についてさらに掘り下げていく。

3．公開鍵 n の素因数分解

(1)素因数分解アルゴリズムの発展

素因数分解問題の困難性とは、合成数 n を素因数分解することは原理的には可能であるものの、n の桁数が大きい場合、実際に素因数 p と q を導出することは計算量的に困難であることを意味する。

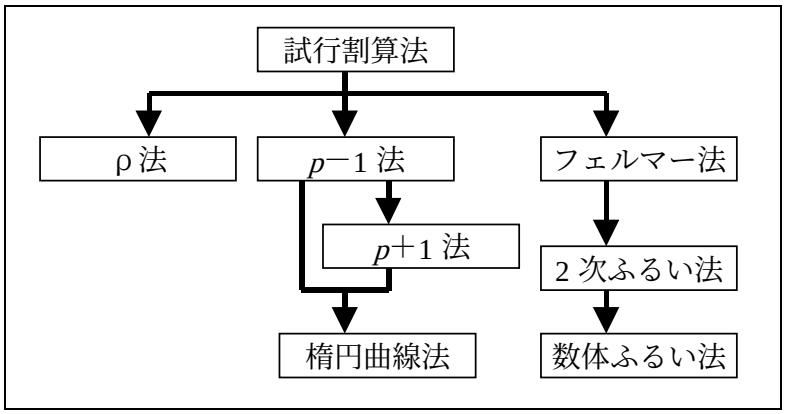
合成数 n の素因数分解を行う最も単純なやり方は、小さい素数で順に n を割っていくという方法である。n は 2 つの素数の積で構成されているため、$\sqrt{n}$ より小さいすべての素数を使って n が割り切れるかどうかをテストしていけば、最終的に n の素因数を求めることができる。この方法は試行割算法（Trial Division）と呼ばれる。試行割算法によって n の素因数分解を行う際に必要となる計算量は、n の桁数を k とすると k の指数関数として表わされ（このようなアルゴリズムは指数時間アルゴリズムと呼ばれる）、k が大きくなると膨大な計算量が必要となる。通常 RSA 署名方式の公開鍵 n として用いられるような桁数の大きい（例えば、桁数が 2 進数で 512 桁や 1024 桁といった）合成数を、試行割算法を用いて素因数分解することは、事実上困難であることが知られている。

素因数分解を試行割算法よりも効率的に行う方法を求めて、整数論の分野では従来から様々な研究が行われてきた。特に、1970 年代以降、コンピューターを利用して n の素因数分解を効率的に実行する方法について盛んに研究が行われるようになった。こうした中、ρ 法（Pollard［1975］）、$p-1$ 法（Pollard［1974］）、フェルマー法（Lehman［1974］）が提案された。これらのアルゴリズムは、いずれも試行割算法と同様に指数関数アルゴリズムであり、大きなサイズの n に対しては有効なアルゴリズムとは言えない。ただし、$p-1$ 法やフェルマー法は、p や q がある特殊な性質を有している場合には効率的に n を素因数分解することができる。

その後、1980 年代に入ると、ρ 法や $p-1$ 法等よりも効率的に素因数分解を行うことを可能にするアルゴリズムとして、楕円曲線法（Lenstra［1987］）や数体ふるい法（Lenstra et al.［1990］）が提案された。楕円曲線法は $p-1$ 法から、数体ふるい法はフェルマー法から生み出されたものと位置付けることができる（Koblitz［1997］）。現時点では $n = p \cdot q$（ただし、p と q のサイズはほぼ同じ）タイプの合成数 n の素因数分解に関して、数体ふるい法が最高速のアルゴリズムとされている。

これらの素因数分解アルゴリズムの発展を系図として示せば図 7 のとおりである。以下では、各素因数分解アルゴリズムの概要を見ていくこととする。

図 7：素因数分解アルゴリズムの発展の系図



イ．ρ 法

ρ 法は、1975 年に Pollard によって提案された方法である。

＜基本的な考え方＞
ある適当な数 x_0 を選択した上で、$x_{i+1} = x_i^2+1 \pmod n$に従う数列を作成し、数列の中から選択した x_a と x_b（$a \neq b$）の差と n との最大公約数を求める。1 でも n でもない最大公約数が求められれば、それが n の素因数となる。様々な x_a と x_b に対して同様の試算を行うことによって、n の素因数を探索することができる。

ロ．$p-1$ 法

$p-1$ 法は、1974 年に Pollard によって提案された方法であり、$p-1$ が比較的小さな素数の積で表わされる特殊なケースにおいて効率的に素因数分解を行うことが可能である。

＜基本的な考え方＞
n と互いに素な自然数 a を選び、整数 B を適当に定め、整数 B よりも小さいすべての素数のべき乗の積として計算される k を選ぶ。続いて、 $(a^k \bmod n)-1$ と n の最大公約数を求める。1 でも n でもない最大公約数が求められれば、それが n の素因数となる。様々な値を B、k として同様の試算を行うことにより、n の素因数を探索することができる。

他方、$p+1$ が比較的小さな素数の積で表される場合には、$p-1$ 法を変形した $p+1$ 法と呼ばれる方法が Williams によって提案されている（Williams［1982］）。

ハ．フェルマー法

1974 年に Lehman によって提案された方法であり、$|p-q|$が小さな値になる

ケースにおいて、合成数 n を素因数分解する際に効果的である。

＜基本的な考え方＞
$\sqrt{n}$ の整数部分に 1 を加えた数を t_0 とし、t_0^2-n を計算する。$t_0^2-n=s^2$ となる整数 s が存在しない場合は、t_0 に 1 を加えて再び同じ計算を行い、s が得られるまで同様の計算を繰り返す。このような s が入手できれば、$n=t_i^2-s^2=(t_i-s)(t_i+s)$ より、t_i-s、t_i+s が n の素因数となる。

ニ．楕円曲線法

楕円曲線法は、1987 年に Lenstra によって考案された方法であり、$p-1$ 法や $p+1$ 法において利用されている演算を楕円曲線[3]上の有理点での演算に置き換えたものである。

＜基本的な考え方＞
n より小さい整数 a、x_1、y_1 を無作為に選び、$b=y_1^2-x_1^3-a\cdot x_1 \pmod n$ を計算する。続いて、E を $Y^2=X^3+aX+b \pmod n$で定義される 3 次曲線（楕円曲線）とし、E 上の点(x_1, y_1)を T とする。さらに、小さい素数の小さいべき乗の積であるような数を k として、T の k 倍点 $kT=(x_k, y_k)=(c_k/d_k^2, e_k/d_k^3)$を計算し、$d_k$ と n の最大公約数を算出する。1 でも n でもない最大公約数が求められれば、それが n の素因数となる。様々な値を k に代入したり、様々な楕円曲線 E を利用することによって、n の素因数を探索することができる。

前述の $p-1$ 法や $p+1$ 法では、合成数 n の素因数 p に対して、$p-1$ や $p+1$ が小さな素因数のみによって構成される場合に有効であった。これに対し、楕円曲線法では、$p-1$ 法や $p+1$ 法とは異なり、素因数 p が特殊な性格を満たさなくても適用が可能であるという特徴がある。

ホ．2 次ふるい法

2 次ふるい法は、1984 年に Pomerance によって提案されたものであり、フェ

---

[3] 楕円曲線： $Y^2=X^3+aX+b$ によって表わされる 3 次曲線。楕円曲線上の点に無限遠点 O を追加し、楕円曲線上の点 P, Q に対して以下のように演算を定義すると、楕円曲線上の有理点の集合は有限可換群となる。

(1) $-O=O$
(2) $P+O=O+P=P$
(3) $-P=-(x, y)=(x,-y)$　（$P=(x, y)$）
(4) $P\neq Q$ の場合、$P+Q=-R$ (R は PQ を延長した直線と楕円曲線との交点)
(5) $P=Q$ の場合、$2P=-R$ (R は P における接線と楕円曲線との交点)

ルマー法のアイデアを一般化し、$T^2 = S^2 \pmod n$となる T、S を求めるという方法である（Pomerance［1984］）。

＜基本的な考え方＞

αを $\sqrt{n}$ の整数部分とし、関数 $Q(x) = (x+\alpha)^2 - n$ を導入する。$Q(x)$に様々な整数 x_i を代入し、得られた $Q(x_i)$を素因数分解したとき、小さな素因数 p_j のみによって、$Q(x_i) = p_1^{a_1} \cdot p_2^{a_2} \cdot \cdots \cdot p_r^{a_r}$（$a_j$ は整数）が成立するものを多数集める。このような $Q(x_i)$同士を掛け合わせることによって、$\prod Q(x_i) = \prod p_j^{2 \cdot a_j} \pmod n$ が成立するような $Q(x_i)$のグループを探索する。このようにして選ばれた $Q(x_i)$を用いて、$T = \sqrt{\prod Q(x_i)}$ 、$S = \prod p_j^{a_j}$ とすれば、$T^2 = S^2 \pmod n$ を満足する。こうした T、S の中で $T \neq \pm S \pmod n$となる T、S を見つけることができれば、$T-S$ と n の最大公約数（または $T+S$ と n の最大公約数）は n の素因数となる。

へ．数体ふるい法

数体ふるい法は、1990 年に A. K. Lenstra、H. W. Lenstra、Manasse、Pollard によって提案された方法であり、2 次ふるい法をさらに発展させたものである。T と S のペアを効率的に探索するために、一意分解整域と環の準同型写像の性質を利用している。

＜基本的な考え方＞

$f(x)$を最高次係数が 1 の d 次整数係数既約多項式とし、$f(x) = 0$ の根の一つをαとし、整数環 $\mathbf{Z}$ の元を係数とするαの多項式全体の集合を $\mathbf{Z}[\alpha]$とする。$\mathbf{Z}[\alpha]$は可換環であるが、ここでは $\mathbf{Z}[\alpha]$について「$\mathbf{Z}[\alpha]$の任意の元は既約元の積で書き表せ、その表わし方は一意的であること」という条件が成立する（一意分解整域）と仮定する。φを $\mathbf{Z}[\alpha]$から可換環 $\mathbf{Z}/n\mathbf{Z}$ への環の準同型写像とする。

$$\varphi : \mathbf{Z}[\alpha] \rightarrow \mathbf{Z}/n\mathbf{Z}$$
$$a + b\alpha \rightarrow a + bC \quad （a、b は互いに素な整数）$$

$a+b\alpha$は、$\mathbf{Z}[\alpha]$ 上で既約元の積、$a + b\alpha = p_1 \cdot p_2 \cdot \cdots \cdot p_k$（$p_i$ は既約元）で表わせるので、$\mathbf{Z}/n\mathbf{Z}$ 上では、

$$a + bC = \varphi(a + b\alpha) = \varphi(p_1 \cdot p_2 \cdot \cdots \cdot p_k) = \varphi(p_1) \cdot \varphi(p_2) \cdot \cdots \cdot \varphi(p_k) \pmod n$$

が成立する。$a + bC = t$、$\varphi(p_1) \cdot \varphi(p_2) \cdot \cdots \cdot \varphi(p_k) = s$、とし、このような$(s, t)$の組を多数集める。入手した$(s, t)$の中からいくつかを掛け合わせることによって、$(\prod t_i)^2 = (\prod s_j)^2 \pmod n$が成立するような$(s, t)$を探索する。このようにして選ばれた$(s, t)$を用いて、$T = (\prod t_i)$、$S = (\prod s_j)$とすれば、 $T^2 = S^2 \pmod n$ を満足する。こうした T、S の中で $T \neq \pm S \pmod n$となる T、S を見つけることができれば、$T-S$ と n の最大公約数（または $T+S$ と n の最大公約数）は n の素因数となる。

(2)各素因数分解アルゴリズムの計算量

次に、各アルゴリズムを用いて合成数 $n = p \cdot q$ を素因数分解するために必要な計算量について整理してみよう（表 2 参照）。

表 2：素因数分解アルゴリズムとその計算時間

	提案者 (提案年)	ｱﾙｺﾞﾘｽﾞﾑの ｵｰﾀﾞｰ	計算時間 （O 表記）	計算時間 （L 表記）
試行割算法	— （ — ）	指数時間	$O(\exp(1/2 \cdot \log n))$	$L_n(1; 1/2)$
ρ 法	Pollard （1975）	指数時間	$O(\exp(1/4 \cdot \log n))$	$L_n(1; 1/4)$
$p-1$ 法	Pollard （1974）	指数時間	$O(q_1)$, q_1 は $p-1$ の最大素因数のサイズ	—
$p+1$ 法	Williams （1982）	指数時間	$O(q_2)$, q_2 は $p+1$ の最大素因数のサイズ	—
フェルマー法	Lehman （1974）	指数時間	$O(\exp(1/3 \cdot \log n))$	$L_n(1; 1/3)$
楕円曲線法	Lenstra （1987）	準指数時間	$O(\exp((1+o(1)) \cdot (\log n)^{1/2} \cdot (\log\log n)^{1/2}))$	$L_n(1/2; 1+o(1))$
2 次ふるい法	Pomerance （1983）	準指数時間	$O(\exp((1+o(1)) \cdot (\log n)^{1/2} \cdot (\log\log n)^{1/2}))$	$L_n(1/2; 1+o(1))$
数体ふるい法	Lenstra 他 （1990）	準指数時間	$O(\exp((64/9)^{1/3}+o(1)) \cdot (\log n)^{1/3} \cdot (\log\log n)^{2/3}))$	$L_n(1/3; (64/9)^{1/3}+o(1))$

(注) $O(k^d)$：アルゴリズムを実行するのに必要な計算量のオーダーが k^d であることを表記したもの。

$L_n(r; c) = O(\exp(c \cdot (\log n)^r (\log\log n)^{1-r}))$

$o(1)$は n が大きくなればなるほど 0 に近づく。

試行割算法および ρ 法、$p-1$ 法、$p+1$ 法、フェルマー法は指数時間アルゴリズムであり、n の桁数 k が大きくなると、膨大な計算量が必要となるため、大きな素数の素因数分解には向かない。ただし、同じ指数時間アルゴリズムにおいても、試行割算法、フェルマー法、ρ 法の計算量は $e^{1/2 \cdot k}$、$e^{1/3 \cdot k}$、$e^{1/4 \cdot k}$ と表わされ、これらの中では、ρ 法の計算量が比較的少ない。

他方、楕円曲線法、2 次ふるい法、数体ふるい法は、指数時間アルゴリズムよりも少ない計算量で素因数分解を実行することが可能であり、準指数時間アルゴリズムと呼ばれるカテゴリーに分類される。準指数時間アルゴリズムの計算量は、計算量が k の多項式で表わされる多項式時間アルゴリズムと指数時間アルゴリズムの中間に位置する。

準指数時間アルゴリズムの計算時間については、L 表記（$L_n(r; c)$による計算量の表記）を導入すると理解し易い。L 表記は、準指数時間アルゴリズムの計算量を、指数時間アルゴリズム（計算量は $\exp(c \cdot \log n)$に比例）と、多項式時間アル

ゴリズム（計算量は exp(c･loglog n)に比例）の加重平均で表わすもので、加重平均のパラメータが $r = 1$ であれば指数時間アルゴリズム、$r = 0$ であれば多項式時間アルゴリズムとなる。楕円曲線法、2 次ふるい法、数体ふるい法の計算時間を L 表記で比較すると、それぞれ、$L_n(1/2; 1+o(1))$、$L_n(1/2; 1+o(1))$、$L_n(1/3; (64/9)^{1/3}+o(1))$であり、$r$ が一番小さい数体ふるい法が最も多項式時間アルゴリズムに近い素因数分解アルゴリズムであることが分かる。

現在のところ、数体ふるい法が最速のアルゴリズムであるが、今後、より高速な素因数分解アルゴリズム（L 表記において r が 1/3 よりも小さい準指数時間アルゴリズムや、さらには多項式時間アルゴリズム）が提案される可能性もある。例えば、将来量子コンピューターによる素因数分解を行うことが可能になれば、Shor のアルゴリズムと呼ばれる方法が利用でき、多項式時間で素因数分解を行うことができるとの指摘もある（Shor［1994］）。

(3)安全な公開鍵サイズに関する検討

これまでに提案された素因数分解アルゴリズムによって実際にどの程度の大きさの合成数 n がどの程度のコスト・時間によって素因数分解できるのかについて、様々な検討が行われてきた。その代表的なものとしては、RSA 社が主催する素因数分解コンテストである「RSA Factoring Challenge」、Lenstra と Verheul の分析（Lenstra and Verheul［2001］）、Silverman の分析（Silverman［2001］）の 3 つが挙げられる。

イ．RSA Factoring Challenge

RSA Factoring Challenge は、素因数分解に対する RSA 暗号／署名方式の安全性を実証することを主な目的として開催されてきたコンテストである。RSA 社のホームページに掲載された、いくつかのサイズの異なる合成数（2 つの素数の積）を素因数分解することを競うもので、誰でも挑戦することができる[4]。RSA 社の発表資料によれば、1991 年にコンテストが開始されて以降、これまでに 7 種類の合成数について素因数分解の成功例が報告されている（表 3 参照）。

これらの成功例をみると、1991 年から 1994 年にかけて、330 bit から 425 bit までの合成数が素因数分解されたが、当時、素因数分解成功者が利用していたのは、2 次ふるい法であった。1996 年以降は、数体ふるい法を利用して、より

[4] 詳細は、RSA 社のホームページに掲載されている「The RSA Challenge Numbers」（http://www.rsasecurity.com/rsalabs/challenges/factoring/numbers.html） を参照。

サイズの大きい合成数の素因数分解が行われている。

1999 年 8 月の成功例をみると、オランダの研究機関の暗号研究者を中心とした作業チームが、ワークステーションやパソコン 293 台を利用し、数体ふるい法によって 512 bit の合成数を素因数分解している。実際の計算においては、①数体ふるい法を実行するために必要な多項式を見つけるための事前計算に約 2.2 ヶ月、②公開鍵 *n* の素因数を見つけるための手がかりとなる整数を絞り込むために約 3.7 ヶ月、③上記①によって得た多項式と上記②によって入手した整数を利用して実際に公開鍵 *n* の素因数を計算するために約 1.5 ヶ月、が費やされた。この結果、素因数分解に費やされた時間は、事前計算を含めない場合には約 5.2 ヶ月、事前計算を含める場合には約 7.4 ヶ月となった。

こうした結果を踏まえて、現在 RSA 社は、公開鍵 *n* のサイズとして 1024 bit 以上を推奨するとしている[5]。

表 3：RSA 社による RSA Factoring Challenge の結果

公開鍵のサイズ（bit）	利用された素因数分解アルゴリズム	計算量 (MIPS 年)	素因数分解に成功した年月
330	2 次ふるい法	7	1991 年 4 月
364	2 次ふるい法	75	1992 年 4 月
397	2 次ふるい法	830	1993 年 6 月
425	2 次ふるい法	5000	1994 年 4 月
430	数体ふるい法	500	1996 年 4 月
463	数体ふるい法	2000	1999 年 2 月
512	数体ふるい法	8000	1999 年 8 月

ロ．Lenstra と Verheul の分析

Lenstra と Verheul は、素因数分解に対する RSA 暗号／署名方式の安全性を DES 暗号の安全性と対比させて検討を行った。

DES は、1977 年に米国連邦政府標準暗号に制定された鍵長 56 bit の共通鍵ブロック暗号方式であり、世界中で幅広く利用されていたが、1998 年に 25 万ドルで開発された DES 解読装置によって約 56 時間（計算量は約 50 万 MIPS 年）で解読が可能であることが実証された（Blaze et al.［1996］）。DES は、開発されてから解読されるまでの間、常に高い信頼性を保持していた訳ではなく、1990

[5] 詳細は、RSA 社のホームページに掲載されている FAQ（http://www.rsasecurity.com/rsalabs/faq/3-1-5.html） を参照。

年代の初め頃から、様々な暗号学者が「DES は全数探索法[6]によって解読が可能である」ということを指摘していた。

このような状況を踏まえ、Lenstra らは、DES の信頼性が極めて高かった時期として 1982 年を想定し、1982 年時点の DES と同等の安全性を達成するには RSA 暗号／署名方式においてどのくらいの鍵長のサイズを持てば良いかについて分析を行った。

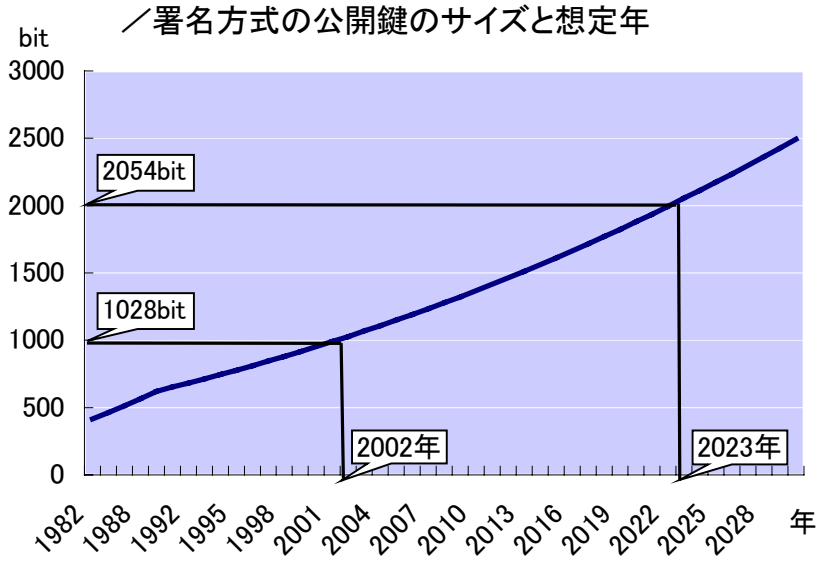
具体的には、次のような仮定の下で試算を行っている。

① 全数探索法による DES の解読に要する計算量は 50 万 MIPS 年であり、1982 年当時においてはこの計算量で安全性が達成されていた。
② 一定の金額で購入できるコンピュータの処理速度は 18 ヶ月で 2 倍になり、1982 年当時の 50 万 MIPS 年に相当する安全性を達成できる計算量もこれと同じペースで増大する。
③ 1999 年 8 月時点で、数体ふるい法によって 512 bit の合成数を素因数分解するのに要した計算量は 8000 MIPS 年である。同じ時点でサイズの異なる合成数を素因数分解するための計算量は、数体ふるい法における合成数のサイズと計算量の関係から推定することができる。
④ 素因数分解法の改良により、同じサイズの合成数を素因数分解するために必要となる計算量は、18 ヶ月で半分になる。

その結果、Lenstra らは、RSA 暗号／署名方式において、2002 年時点の 1028 bit の鍵長と 2023 年時点の 2054 bit の鍵長は、1982 年時点の DES の安全性と同等であるという分析結果を発表している（図 8 参照）。このため、Lenstra らは、「今後 20 年間の利用を前提にするのであれば、鍵長は 2048 bit にすべきである」と主張している。

[6] 全数探索法：入手した暗号文を候補となる鍵で順々に復号することによって真の鍵を探索する方法。

図8：1982年時点のDESの安全性と等価なRSA暗号／署名方式の公開鍵のサイズと想定年



出典：Lenstra and Verheul［2001］

ハ．Silverman **の分析**

Silverman は、素因数分解を行うために必要となるコストをより具体的に考慮した検討を行っている。Silverman は、まず、①全体の予算を 1 千万ドルと想定し、②素因数分解アルゴリズムとして数体ふるい法を採用し、③数体ふるい法の実行に必要な記憶容量を算出した上で、④その記憶容量分の半導体メモリーを購入するために必要なコストを 1 千万ドル（予算）から差し引いて残った資金で調達可能なパソコン（PentiumⅢ 500MHz を搭載）の台数を算出した。次に、これらの装備によって素因数分解を実行した場合、合成数のサイズが 430、760、1020、1620 bit の時に必要な計算時間の見積もりを行った（表 4 参照）。

この結果、Silverman は、1020 bit のサイズの合成数を 1 千万ドルで素因数分解するためには約 3 百万年かかるとの試算を示した上で、「素因数分解に必要とされる計算機のパワーだけではなく、必要とされる記憶装置のコストをも考慮すると、1024 bit の合成数は少なくとも今後 20 年は素因数分解できないと考えて良い」と報告している。

表 4：1 千万ドルのコストで素因数分解を行った場合の鍵長と所要時間の関係

公開鍵のサイズ（bit）	素因数分解に要する時間	利用 PC 数	所要記憶容量
430	5 分未満	100000	わずか
760	600 ヶ月	4300	4Gb
1020	3 百万年	114	170Gb
1620	10^{16} 年	0.16	120Tb

このように、鍵長の耐久年数に対する見解は研究者によって様々であるが、どの程度のサイズの n であれば素因数分解が計算量的に困難であるか否かは、その時々の技術条件、すなわち、素因数分解アルゴリズムの研究の進展やコンピューターのコスト・パフォーマンスの向上等に依存して決定される。このため、RSA 署名方式を利用してシステムの安全を守ろうとする場合、常に最新の技術動向に注意しておく必要がある。

4．RSA 関数の逆関数を導出することなく署名を偽造する攻撃

(1)積攻撃

2. で説明したように、RSA 関数の逆関数を導出することなく、署名を偽造する攻撃が存在する。こうした攻撃には、RSA 署名方式の乗法性を利用したものが多い。RSA 署名方式の乗法性とは、「（mod n において）メッセージ M_1 とメッセージ M_2 の積で表わされるメッセージ $M_1 \cdot M_2$ に対する署名と、M_1 に対する署名と M_2 に対する署名の積は等しい」という性質を指す。

＜RSA 署名方式の乗法性＞

メッセージ M_1 に対する署名を $S(M_1)$、メッセージ M_2 に対する署名を $S(M_2)$ とすると、RSA 署名方式の定義により、メッセージ $M_1 \cdot M_2$ に対する署名を $S(M_1 \cdot M_2)$ は次のように表わされる。

$$\begin{aligned} S(M_1 \cdot M_2) &= (M_1 \cdot M_2)^d \pmod{n} \\ &= (M_1)^d \cdot (M_2)^d \pmod{n} \\ &= [(M_1)^d \pmod{n}] \cdot [(M_2)^d \pmod{n}] \pmod{n} \\ &= S(M_1) \cdot S(M_2) \pmod{n} \end{aligned}$$

このような性質に着目した基本的な攻撃法が積攻撃（multiplicativity attack）である。積攻撃は、攻撃者が署名を偽造したいメッセージを X、署名生成者から正当な署名が得られたメッセージを M とすると、X と M の積 $X \cdot M \pmod{n}$ に対する署名を入手することによって、X の署名を偽造するという攻撃法である。

＜積攻撃＞

(i)攻撃者が署名を偽造したいメッセージを X とする。

(ii)攻撃者は、署名生成者からメッセージ M に対する署名 $S(M)$ を入手した上で、$Y = X \cdot M \pmod{n}$ を計算し、Y に対する署名 $S(Y)$ を入手する。

(iii)攻撃者は、入手した署名 $S(M)$ の mod n 上での逆数 $S(M)^{-1}$ と $S(Y)$ を掛け合わせることによって、X の署名 $S(X)$ を偽造することができる。

$$\begin{aligned} S(Y) \cdot S(M)^{-1} &= S(X \cdot M) \cdot S(M)^{-1} \pmod{n} \\ &= S(X) \cdot S(M) \cdot S(M)^{-1} \pmod{n} \\ &= S(X) \pmod{n} \end{aligned}$$

積攻撃は、$X \cdot M \pmod{n}$ に対する署名を署名生成者から入手することを前提とした攻撃である[7]。このような前提は現実的なものといえるのだろうか。もちろ

[7] このような攻撃は能動的攻撃（5. 参照）と呼ばれる。能動的攻撃が実際にどれだけ適用可能かについては、従来疑問視する見方が少なくなかった。しかし、1998 年、Bleichenbacher によって、（RSA 署名方式に対してではないが）SSL Ver.3.0 の中で利用されている RSA 暗号方式に対して能動的攻撃の適用が指摘されて以来（Bleichenbacher

ん、ここでは署名生成者が自分の意志に反するメッセージ X そのものに署名を生成してしまう事態は考えない。しかし、X と M の選び方によっては、$X \cdot M \pmod{n}$が署名をするのに不自然でないようなデータとなる可能性は否定できない。

そもそも、デジタル署名方式がどのような環境においても安全に利用されるためには、署名生成者が自らの意思で署名した以外の情報について、デジタル署名が生成されてしまうことは望ましいことではなく、既知の署名 $S(M_1)$、$S(M_2)$ の積によって異なる署名 $S(M_1 \cdot M_2)$が計算できてしまうということ自体が問題といえる。このため、積攻撃に対して耐性を持つような署名方式が研究されるようになった。

(2)積攻撃への対応法

積攻撃を回避するために、署名対象のメッセージに何らかの変形を施したデータを署名変換データとする方法が提案されており、これらは、①メッセージにハッシュ関数による変換を施す方法、②メッセージの前後にパディングデータ（メッセージとは関係のないデータ）を付加する方法、③ハッシュ関数とパディングの両方を利用する方法、の 3 通りに大別できる。ただし、これらの改良方式の中には、新たな攻撃法が提案されているものも存在する。上記 3 種類の改良方式に対する積攻撃の有効性について予め整理すると、表 5 のとおりである。

［1998］)、こうした攻撃の実現の可能性を軽視すべきではないという見方が強まっている。

表 5：積攻撃に対する RSA 署名方式とその改良方式の耐性

	単純な RSA 署名方式	パディングを利用する方法	ハッシュ関数を利用する方法	ハッシュ関数とパディングの両方を利用する方法
署名生成	$S(M) = M^d \bmod n$	$S(M) = (R(M))^d \bmod n$（R はパディングデータを付加する関数）	$S(M) = (\mathrm{H}(M))^d \bmod n$（H はハッシュ関数）	$S(M) = (T(M))^d \bmod n$（T はハッシュ関数とパディングの両方を利用する関数）
積攻撃を適用するための条件	$Y = X \cdot M \pmod n$ となる Y を探索。	$R(Y) = R(X) \cdot R(M) \pmod n$ となる Y を探索。	$H(Y) = H(X) \cdot H(M) \pmod n$ となる Y を探索。	$T(Y) = T(X) \cdot T(M) \pmod n$ となる Y を探索。
積攻撃に対して期待される耐性	上の式を満たす Y を容易に探索することができ、積攻撃は適用可能。	$Z = R(X) \cdot R(M) \pmod n$ となる Z を求めることができても、$Z = R(Y)$ となるメッセージ Y を求めることは困難であるため、積攻撃は適用困難であると期待される。	ハッシュ値 $H(Y)$からメッセージ Y を求めることは困難であるため、積攻撃は適用困難であると期待される。	ハッシュ関数を利用する方法とパディングを利用する方法は、各々積攻撃の適用が困難であると期待されることから、両方を組合せた方法も積攻撃は適用困難であると期待される。
既存の攻撃	上記のように、積攻撃が適用可能。	De Jonge と Chaum の攻撃、Girault-Misarsky 攻撃、Misarsky 攻撃によって、積攻撃が適用可能であることが示されている。	積攻撃の適用例は、これまでに発表されていない。しかし、RSA 署名方式の乗法性を利用した別の攻撃として、Desmedt-Odlyzko 攻撃が提案されている。	積攻撃の適用例は、これまでに発表されていない。しかし、RSA 署名方式の乗法性を利用した別の攻撃として、CNS 攻撃が提案されている。

(注)X：攻撃者が署名を偽造したいメッセージ。
M：署名生成者が署名したメッセージ。

イ．パディングを利用する方法とその攻撃法

署名対象のメッセージが鍵長のサイズに満たなかった場合には、例えば鍵長のサイズに満たない部分に「0」を挿入することが考えられる。このように、メッセージ以外の部分にメッセージとは関係のないデータを付加する方法のことを一般に「パディング」と呼ぶ。

署名変換データを生成する際にパディングを利用することによって、積攻撃を防ぐ効果が期待できる。例えば、n が 1024 bit であり、768 bit のメッセージから 1024 bit の署名変換データを生成する際に、メッセージを右にシフトさせてメッセージ以外の部分に「0」を連結するというパディング方法を考える。このようなパディングを行う関数を R とし、積攻撃が可能か否かを検討する。積攻

撃を利用するためには、

$$R(Y) = R(X)\cdot R(M) \pmod{n}$$

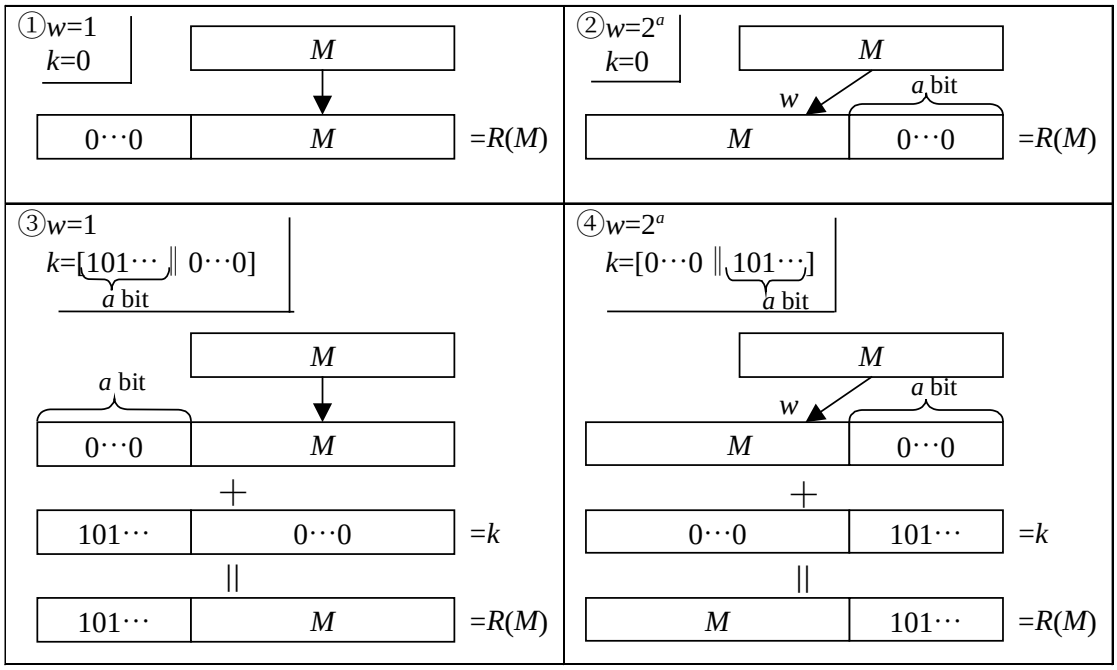
を満たす Y を見つけなければならない。しかし、積攻撃の手順として紹介したように、$Z = R(X)\cdot R(M) \pmod{n}$を計算しても、上記関係式を満たす Y を導出することはできない。n は 1024 bit なので、このようにして算出した Z は 1024 bit の値となり、そこから $Z = R(Y)$となるような 768 bit の Y を推定することは困難であるからである。$R(X)\cdot R(M) \pmod{n}$の上位 256 bit が偶々「0」になれば、残りの 768 bit を Y として利用できるが、このように、偶然適切な Y を入手できる確率は十分小さいと考えられる。

基本的なパディングのバリエーションは次の 4 つが考えられる。

①メッセージを右にシフトさせ、残りの部分に「0」を連結する方法
②メッセージを左にシフトさせ、残りの部分に「0」を連結する方法
③メッセージを右にシフトさせ、残りの部分に「0」以外の値を連結する方法
④メッセージを左にシフトさせ、残りの部分に「0」以外の値を連結する方法

M の前後に特定の値を連結するというパディング方法は、③と④の方法を組み合わせることによって得られる。これらの方式は、一見すると全く異なる方法のように見えるが、すべて $R(M) = w\cdot M+k$（w、k は定数）という式で表わされる。例えば、①のパディング方法を実現する場合、$w = 1$、$k = 0$ となり、②のパディングを実現する場合には、$w = 2^a$、$k = 0$ となる（図 9 参照）。

図 9：基本的なパディング方法



(注)“‖”は、データの連結を意味する。

当初、このようなパディングを利用した RSA 署名方式は積攻撃に対して有効であると見られていたが、①～③のパディング方法を用いた RSA 署名方式に対して積攻撃が有効であることを 1985 年に De Jonge と Chaum が示した（De Jonge and Chaum［1985］）。De Jonge と Chaum は、$X \cdot M \pmod{n}$によって Y を求めるのではなく、パディング方法に着目した演算を行い、ユークリッドの互除法を応用したアルゴリズムを適用することによって、積攻撃を適用可能なメッセージ Y を効率的に探索できることを明らかにした。

また、Girault と Misarsky は、1997 年に De Jonge と Chaum の攻撃を発展させた攻撃法（Girault-Misarsky 攻撃）を発表した（Girault and Misarsky［1997］）。Girault-Misarsky 攻撃は、①～④のパディング方法を一般化した方法、すなわち、メッセージの左右にパディングデータを連結する方法を利用した RSA 署名方式にも適用可能である（De Jonge と Chaum の攻撃は Girault-Misarsky 攻撃の 1 形態となる）。Girault-Misarsky 攻撃の概要は以下のとおりである。

＜Girault-Misarsky **攻撃**＞
(i) 攻撃者が署名を偽造したいメッセージ（パディングを利用する前のサイズ）を X とする。
(ii) z を $z = k/w\{1-(w\cdot X+k)\} \bmod n$ とする。
(iii) ユークリッドの互除法を応用したアルゴリズムを利用して、
$$(w\cdot X+k)\,M = Y+z \pmod n$$
を満たす(M, Y)を探索する。
(iv) 探索された(M, Y)は
$$(w\cdot X+k)(w\cdot M+k) = (w\cdot Y+k) \pmod n$$
満たすので、$R(X)\cdot R(M) = R(Y) \pmod n$ が成立する M、Y を入手することができる。

Girault と Misarsky は、この攻撃の発表と同時に、それを回避するような新しいパディング方法も提案した。そのパディング方法は、メッセージおよびメッセージを剰余演算によって変換した値をそれぞれ分割し、分割後の値を一定の手順によって組み合わせるという方法である。この方法は、メッセージ復元型のデジタル署名方式の国際標準である ISO 9796-3 に採用され、標準化作業が行われていた。しかし、1997 年、このパディング方法に対する有効な攻撃法が Misarsky 自身によって発見された（Misarsky 攻撃、Misarsky［1997］）。このため、1999 年 4 月に ISO 9796-3 の標準化作業が中止され、ISO 9796-3 は廃止されている。

このように、パディングによって署名変換データに冗長性を付与し、積攻撃を回避しようとする試みに対しては、次々に有効な攻撃法が提案されてしまった。このため、現在では、このアプローチで安全性を高めようとする研究は行われておらず、ハッシュ関数とパディングの両方を利用する方法が研究されている。

ロ．ハッシュ関数を利用する方法とその攻撃法

ハッシュ関数とは、与えられた任意のメッセージから固定長の疑似乱数を生成する関数のことであり、ハッシュ関数を利用して生成されたデータは「ハッシュ値」と呼ばれている。安全なハッシュ関数は、ハッシュ値から元のメッセージを求めることが計算量的に困難であるという性質（一方向性と呼ばれる）を持っている。このため、攻撃者はハッシュ値から元のメッセージを容易に算出することができず、積攻撃を回避することが期待できる。

ここで、H をハッシュ関数とし、メッセージに対してハッシュ関数による変

換を施した場合の RSA 署名方式に積攻撃を適用するケースを考える。積攻撃を利用するためには、

$$H(Y) = H(X)\cdot H(M) \pmod{n}$$

となるメッセージ Y を見つけなければならない。しかし、ハッシュ関数の一方向性により、$H(X)\cdot H(M) \pmod{n}$によって求められる $H(Y)$に対応する Y を求めることは困難である。このため、ハッシュ関数を用いる方法では、積攻撃に対しては有効であると期待されていた。

しかし、このような方法に対しては、RSA 署名方式の乗法性を利用した別の攻撃法が見つかっている。1986 年、Desmedt と Odlyzko は、ハッシュ関数を利用した方式に対する攻撃法（Desmedt-Odlyzko 攻撃）を提案した（Desmedt and Odlyzko［1986］）。 Desmedt-Odlyzko 攻撃は、署名が小さな素因数に分解できる場合に適用可能となる。攻撃の概要は以下のとおりである。

＜Desmedt-Odlyzko **攻撃**＞

(i)攻撃者は、メッセージのハッシュ値が比較的小さな素数に素因数分解可能となるメッセージ M を選択する。例えば、ハッシュ関数を H とすると、M のハッシュ値 $H(M)$が

$$H(M) = p_1\cdot p_2\cdot \cdots \cdot p_k$$（p_i は素数、k は正の整数）

となる場合を考える。

(ii)攻撃者は、上記の条件を満たす M を大量に選択し、署名生成者から各 M に対する署名 $S(M)$を入手する。$S(M)$は、

$$S(M) = H(M)^d \bmod n = p_1^d\cdot p_2^d\cdot \cdots \cdot p_i^d \pmod{n}$$ (ただし、$1 \leqq i \leqq k$)

と表わされる。

(iii)攻撃者は、入手した複数の署名を使って、各素数 p_i を d 乗して mod n を計算した値（$p_i^d \bmod n$）を得る。

(iv)攻撃者は、入手した $p_i^d \bmod n$ の値を組合せることによって、ハッシュ値が小さな素数の積となる任意のメッセージに対する署名を偽造することができる。

本攻撃は、メッセージのハッシュ値が比較的短く、それらを素因数分解したり、署名を偽造するための演算が困難ではない、ということが前提とされている。実際、従来のシステム実装で通常利用されてきたハッシュ関数は、ハッシュ値のサイズが 160 bit 程度と比較的短いものであるため、これをそのまま署名対象データとして利用した場合、本攻撃が前提としている計算を行うことは困難とは言えない。逆に、ハッシュ値のサイズを大きくすれば、本攻撃を適用するための計算量や所要データ量が増大するので、本攻撃を受けにくくすることが可能になる。そのような考え方に基づいて提案された RSA 署名方式として、

RSA-FDH 署名方式が挙げられる。これは、通常利用されるハッシュ関数ではなく、ハッシュ値のサイズが公開鍵 n と同一のサイズとなるような「フル・ドメイン・ハッシュ関数」を利用するもので、一定の数学的な仮定に基づき、安全性証明が可能となっている[8]。ただし、RSA-FDH 署名方式を実現するためには、通常よりも大きなハッシュ値を出力するハッシュ関数が必要となり、後述の RSA-PSS 署名方式に比べて実用性に劣るとされている。

ハ．ハッシュ関数とパディングの両方を利用する方法

ハッシュ関数とパディングの両方を利用した代表的な署名方式には、PKCS#1 Ver.1.5 に記載されている署名方式（以下、PKCS#1-1.5 署名方式）と、ISO9796-2 に採用されている方式（以下、9796-2 署名方式）が挙げられる。これらの方式に関する概要は以下のとおりである。

(イ)PKCS#1-1.5 署名方式

PKCS#1 は、RSA 社が作成するデータ守秘方式およびデジタル署名方式の利用方法に関する業界標準であり、現在まで何度か改訂が行われている。PKCS#1-1.5 署名方式は、PKCS#1 の Ver.1.5 において初めて規定された署名方式であり、事実上の標準として広く実装されている。例えば、S/MIME[9]には PKCS#1-1.5 署名方式が実装されている。

なお、現在、PKCS#1 は Ver.2.1 が最新のバージョンであり、PKCS#1-1.5 署名方式が引続き規定されているほか、後述する RSA-PSS 署名方式も規定されている（表 6 参照）。

---

[8] デジタル署名方式に対する安全性証明については、5. を参照。

[9] S/MIME（Secure Multipurpose Internet Mail Extension）：RSA data security 社によって提案された暗号通信、認証等のセキュリティ機能が付加された電子メール用プロトコル。Microsoft 社が無償で提供している Outlook Express 等を初めとする各種メールソフトで利用可能である。現在、IETF（Internet Engineer Task Force）の S/MIME ワーキング・グループにおいて標準化が進められている。

表 6：PKCS#1 に規定されている暗号アルゴリズムの推移

PKCS#1	デジタル署名方式	データ守秘方式	提案年
Version1.5	ハッシュ関数とパディングの両方を利用した方式	パディングを利用する方式	1993 年
Version2.0	上に同じ	RSA-OAEP を追加	1998 年
Version2.1	RSA-PSS 署名方式を追加	上に同じ	2001 年

これまでのところ、PKCS #1-1.5 署名方式に対しては、素因数分解よりも有効な攻撃法は提案されていないものの、後述する RSA-PSS 署名方式のように、一定の仮定の下で、安全性が証明されている訳ではないことに注意が必要である。署名生成・検証手順は以下のとおりである。

＜PKCS #1-1.5 **署名方式**＞

【署名生成】メッセージを M、ハッシュ関数を H（ハッシュ値のサイズを $8h$ bit）、SR を署名変換データ（$8k$ bit）とする。

(i)メッセージ M のハッシュ値 $H(M)$を計算する。

(ii)ハッシュ関数の ID データとハッシュ値 $H(M)$を連結させたデータ T（$8t$ bit）を生成する。

(iii)パディングデータ PS のサイズは $8(k-t)$ bit となり、上位 16 bit の値を「0000 0000 0000 0001」、下位 8 bit の値を「0000 0000」と定め、残りの bit をすべて「1」とする。すなわち、

$$PS = [\underbrace{0000\ 0000\ 0000\ 0001}_{16\ \text{bit}} \| \underbrace{1111\cdots1111}_{8(k-t-3)\text{bit}} \| \underbrace{0000\ 0000}_{8\text{bit}}]$$

とする。

(iv) $SR = [PS \| T]$とする。

(v)上記 SR を秘密鍵 d で変換して署名 $S = SR^d \bmod n$ を生成する。

【署名検証】署名 S と署名検証鍵 e を用いて $S^e \bmod n$ を計算した上で、メッセージ M、ハッシュ関数 H を用いて $S^e \bmod n$ が SR のデータ形式（上記(iv)）を満足していることを確認。

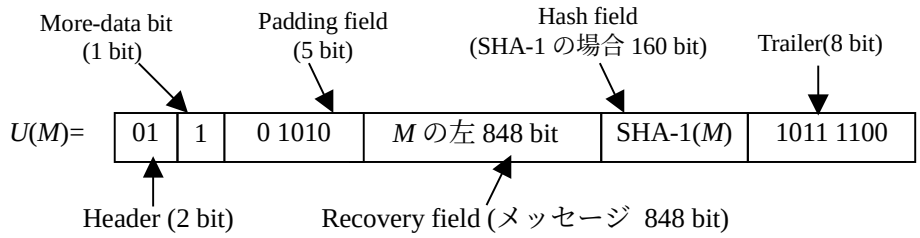
(**ロ**)ISO 9796-2 **署名方式**

ISO 9796-2 は、ハッシュ関数を利用したメッセージ復元型のデジタル署名方式の国際標準であり、Girault-Misarsky 攻撃に対しても、十分な安全性を有するものとして提案された。ISO 9796-2 では、公開鍵 n のサイズが 1024 bit の場合、署名から最大 848 bit のメッセージを復元することが可能である。ISO 9796-2 の署名生成・検証手順について、以下では、公開鍵 n のサイズを 1024 bit、ハッシュ

値のサイズを 160 bit、メッセージのサイズを 848 bit として説明する[10]。

＜ISO 9796-2＞

【署名生成】メッセージ M を用いて以下の署名変換データ $U(M)$ を生成し $S = U(M)^d \bmod n$ によって M に対する署名 S を生成する。



(i) Header：$U(M)$の左から 2 bit 分のデータであり、常に”01”。
(ii)More-data bit：メッセージ全体が復元できる場合は”0”、そうでない場合は”1”。
(iii)Padding field：$U(M)$を 1024 bit にするためのパディングデータ”01010”。
(iv)Recovery field：メッセージのうち復元可能なデータ（左から 848 bit 分）。
(v)Hash field：メッセージのハッシュ値 SHA-1(M)（サイズは 160 bit）。
(vi)Trailer：ハッシュ関数の属性を表すデータ。SHA-1 の場合は”1011 1100”。

【署名検証】$U(M)' = S^e \bmod n$ を計算し、$U(M)'$が ISO 9796-2 規定のフォーマットに従うことを確認した後、メッセージ M を復元。

ISO 9796-2 は 1997 年 9 月に国際標準として成立したが、1999 年 4 月に、Coron、Naccache、Stern によって、n の素因数分解よりも効率的な攻撃法が提案された（CNS 攻撃法、Coron, Naccache, and Stern［1999］）[11]。Coron らは、ISO 9796-2 の安全性上の欠点について、「攻撃者が比較的容易に推定できる bit が署名変換データに多く含まれている」と指摘した。CNS 攻撃の概要については以下のとおりである。

[10] これ以上のサイズのメッセージに対する署名を生成する場合には、メッセージのうち左から 848 bit 分のデータのみが復元可能となる。

[11] CNS 攻撃法の詳細については、宇根［1999］を参照。

＜CNS 攻撃＞

【攻撃のアイデア】

署名変換データ $U(M)$が小さな素因数のみから構成されるメッセージ M をみつける。ただし、$U(M)$を直接素因数分解することは困難なため、代わりに、$a \cdot n - 2^8 U(M)$が小さな素因数のみから構成され、そのサイズが十分小さな合成数になる M と a をみつける。

【攻撃の手順（例 n：1024 bit、M：1024 bit、ハッシュ値 $H(M)$：160 bit）】

(1) $a \cdot n - 2^8 U(M)$**の生成**

$U(M)$は左から 8 bit 分が”0110 1010”となる。

$U(M)$=	01	1	0 1010	M の左 848 bit	$H(M)$	1011 1100

法 n の左 8 bit が”0110 1010”となっていない場合、適当な a を選択して n を a 倍し、左 8 bit が”0110 1010”となる 1032 bit（ = 1024 bit+ 8 bit）のデータ $a \cdot n$ を生成する。データ $a \cdot n$ の右 176 bit の部分を y とし、左 8 bit ”0110 1010”と y の間の部分（848 bit）を x とする。

$a \cdot n$=	0110 1010	x (848 bit)	y (176 bit)

このとき、M の左 848 bit を x に設定すると、$a \cdot n - 2^8 U(M)$ のサイズは、ハッシュ値が 160 bit であることから、176 bit 以下となる。

$a \cdot n - 2^8 \cdot U(M)$=	0110 1010	x（848 bit）	y（176 bit）	
—	0110 1010	x	$H(M)$	1011 1100 0000 0000

$= \quad y - [H(M) \| 1011\ 1100\ 0000\ 0000] \leqq 2^{176}$

(2)M **の選択**

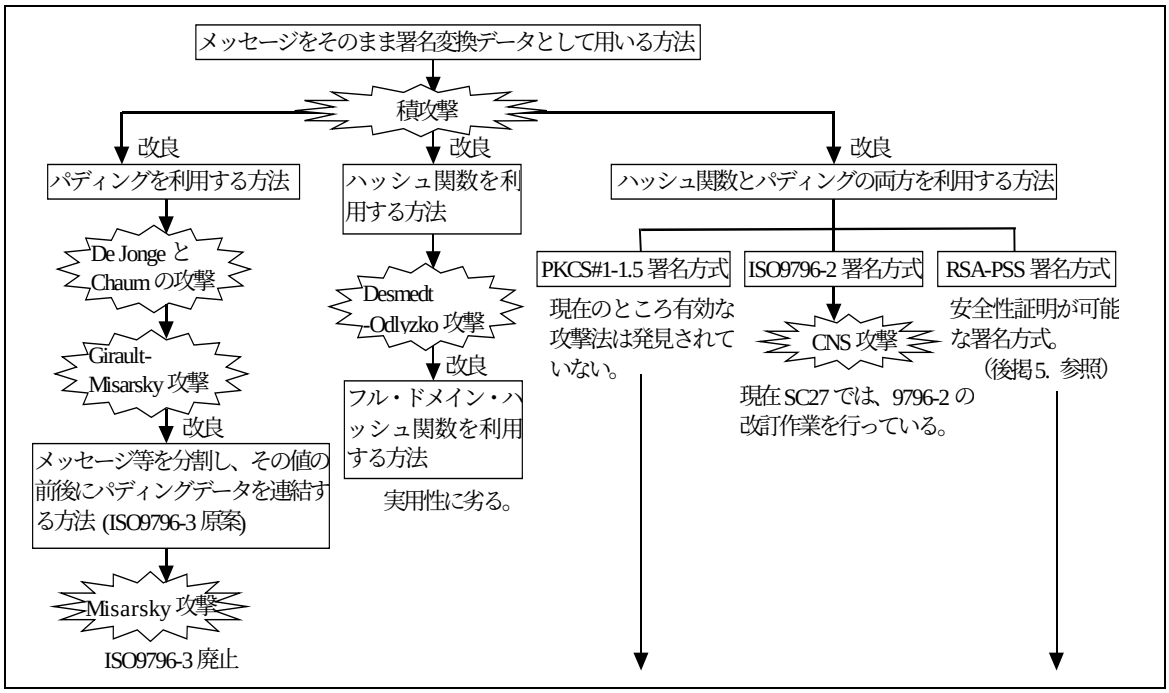
攻撃に利用するメッセージ M は、左 848 bit の部分が x と等しくなるように設定され、残りの 176 bit の部分（z とする）を攻撃者が選択することができる。したがって、M の取り得る値の総数は 2^{176} 個となる。攻撃者は、上記 $a \cdot n - 2^8 U(M)$が小さな素因数のみから構成されるように M を選択する。

(3)M **に対する署名の入手と別の署名の偽造**

攻撃者は、選択した M を署名生成者に送付する。署名生成者は、M に対する署名 $U(M)^d \bmod n$ を生成・返送する。入手した署名は小さな素因数から構成されており、Desmedt-Odlyzko の攻撃法と同様の方法によって、別のメッセージに対する署名を偽造できる。

なお、これまでに提案された RSA 署名方式における署名変換データの生成方法と、それに対する主な攻撃法について整理すると、図 10 のとおりである。現在では、「パディングを利用する方法」および「ハッシュ関数を利用する方法」については、有効な攻撃法が存在しているか、実用性に問題があるため、「ハッシュ関数とパディングの両方を用いる方法」が主流となっている。

図 10：RSA 署名方式における署名変換データの生成方法とそれに対する主な攻撃法



5．RSA-PSS 署名方式の提案

(1)安全性が証明可能な署名方式の必要性

4. で説明したように､RSA 署名方式の署名変換データの生成方法には､パディングやハッシュ関数を利用する様々な方法が提案されてきたが、新たな方法が提案される都度、新たな攻撃法が発見される場合が多かった。PKCS#1-1.5 署名方式においては､現在のところ有効な攻撃法は発見されていないが､過去に様々な攻撃法が発見されてきたように、新たな攻撃法が発見されないとは限らない。

従来提案されてきた様々な署名変換データの生成方法の多くは、確実に署名方式の安全性に結び付くことを示す数学的な根拠に基づいている訳ではなく、あくまで既存の攻撃法や分析手法を基に経験的に改良を施したものであった。CNS 攻撃の発見は、こうした経験的な改良に基づく安全性対策が不十分であり得ることを示唆するものであった。このため、デジタル署名技術の研究者や利用者の間では、「デジタル署名方式の安全性評価には、経験的な分析だけではなく、数学的な根拠が示されることが必要である」という考え方が強まることとなった。

こうした問題が広く認識される以前から、デジタル署名方式の数学的な安全性証明に関する研究が進められてきてはいたが、実用性を兼ね備えた署名方式が提案されるには至っていなかった。しかし、最近では、既存の実用性が高いデジタル署名方式を改良する形で、高い実用性を維持しつつ、数学的に安全性が証明可能なデジタル署名方式がいくつか提案されている。

(2)デジタル署名方式に対する攻撃のタイプと達成度

安全性が証明可能なデジタル署名方式について検討を行うために、まず、デジタル署名方式の実装環境が安全性に及ぼす影響について、①攻撃者が利用可能な情報（攻撃のタイプ）と、②署名偽造の程度（攻撃の達成度）に分類して説明する。

イ．攻撃のタイプ

デジタル署名方式への攻撃は、大きく 2 種類（受動的攻撃と能動的攻撃）に分類することができ、能動的攻撃はさらに 2 種類（一般選択文書攻撃と適応的選択文書攻撃）に分類される（表 7 参照）。

表 7：攻撃のタイプの分類

攻撃のタイプ		内容
受動的攻撃		署名検証鍵(公開鍵)のみを利用して署名を偽造するという攻撃。
能動的攻撃	一般選択文書攻撃	攻撃者が予め指定したいくつかの文書に対応する署名を入手できる場合の攻撃（ただし、署名生成者に署名させる文書を攻撃に先立ってすべて選択しなければならない）。
	適応的選択文書攻撃	署名の入手に当って、署名生成者に署名させる文書の選択を、それまでに入手した署名とそれに対応する文書に関する情報を参考にしながら決定することができる場合の攻撃。

受動的攻撃は、攻撃者が公開情報である署名検証鍵（公開鍵）のみを利用して行う攻撃である。このため、実用的なデジタル署名方式は受動的攻撃に対して十分な安全性を確保することが必要である。他方、能動的攻撃は、攻撃者が自分にとって都合の良い文書に対する署名を入手して行う攻撃であり、実行可能性は受動的攻撃よりも低い。特に、適応的選択文書攻撃では、一旦入手した署名とそれに対応する文書を分析した後、その分析結果を踏まえて新たに文書を選択し、その文書に対する署名を入手することを前提としている。

ロ．攻撃の達成度

デジタル署名方式に対する攻撃の達成度は、一般的偽造と存在的偽造の 2 種類に分類できる（表 8 参照）。

表 8：攻撃の達成度の分類

達成度	内容
一般的偽造	任意の文書に対してデジタル署名を偽造できる。
存在的偽造	ある特定の文書に対してデジタル署名を偽造できる。

一般的偽造はどのような文書に対する署名も偽造可能であるということであり、実現可能性の高い攻撃によって一般的偽造が可能になるとすれば、そのデジタル署名方式は安全性の面で利用に供することが不適切であることになる。また、存在的偽造は、すべての文書に対して署名の偽造が可能となるわけではないが、ある特定の文書に対しては署名の偽造が可能であるということである。このため、存在的偽造は、一般的偽造よりも攻撃の達成度が低い。

このような整理を行うと、RSA 署名方式における公開鍵 n を素因数分解する攻撃は「受動的攻撃によって一般的偽造を可能にするもの」に相当し、RSA 関数の逆関数を導出することなく署名を偽造する攻撃は、「能動的攻撃によって存在的偽造を可能にするもの」に相当する。

デジタル署名方式の安全性の観点からは、「存在的偽造さえも不可能である」署名方式が望ましい。また、最も強力な攻撃は適応的選択文書攻撃であることから、「適応的選択文書攻撃によっても存在的偽造が不可能」なデジタル署名方式が最も望ましいと言える。

(3)RSA-PSS 署名方式とは

RSA-PSS 署名方式とは、RSA 署名方式の署名変換データの生成において、パディングとハッシュ関数を用いて安全性を向上させたものであり、1996 年に Bellare と Rogaway によって提案された（Bellare and Rogaway［1996］）。PKCS#1-1.5 署名方式と異なるのは、RSA-PSS 署名方式はハッシュ関数がランダム・オラクルモデル[12]に従うことと、RSA 関数の一方向性を仮定することにより、適応的選択文書攻撃によっても存在的偽造が不可能となることが数学的に証明されていることである。

RSA-PSS 署名方式は、単に「PSS 署名方式」と呼ばれることも多いため、RSA 署名方式とは別個の署名方式であるかのように誤解されている面もあるが、RSA 署名方式の一つのバリエーションと位置付けるべきものである。RSA 署名方式自体、署名偽造を回避するために、パディングやハッシュ関数を組み合せる改良が繰り返されてきたが、いわばそうした改良の最終的な姿として、証明可能安全性を持つ署名方式が考案されたものである。

RSA-PSS 署名方式は、メッセージと署名の両方を受信者に送信する方式（署名添付型）であるが、Bellare と Rogaway は、RSA-PSS 署名方式をベースに改良を加えることによって、署名と復元できないメッセージの部分を受信者に送信し、それらを用いてメッセージを復元する方式（メッセージ復元型）も提案している。この方式は RSA-PSS-R 署名方式と呼ばれている。

なお、RSA-PSS 署名方式については、これまでに、署名生成・検証時の計算方法が異なる 3 つのバージョンが提案されてきている。以下では、RSA-PSS 署名方式の 3 つのバージョンを、提案された順に RSA-PSS96 署名方式、RSA-PSS98 署名方式、RSA-PSS2000 署名方式と記述することとし、各 RSA-PSS 署名方式をベースにしたメッセージ復元型の署名をそれぞれ RSA-PSS96-R 署名方式、RSA-PSS98-R 署名方式、RSA-PSS2000-R 署名方式と記述することとする（表 9 参照）。

---

[12] ランダム・オラクル・モデル：ハッシュ関数が、入力値に対して真正な乱数を出力し、かつ、同じ入力値に対して同じ乱数を出力するという性質を満足するモデルのこと。

表 9：RSA-PSS 署名方式のバージョン

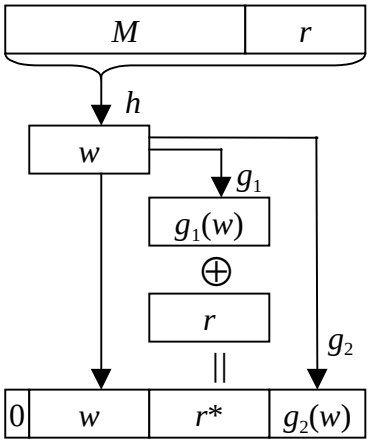
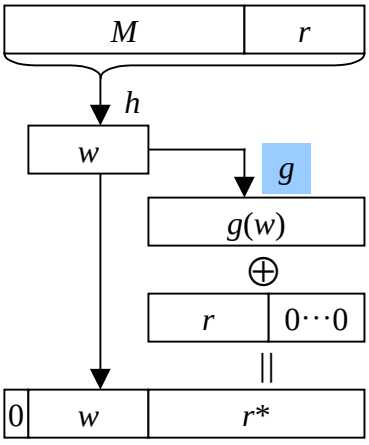
	分類	提案者	提案年
RSA-PSS96 署名方式	署名添付型	Bellare、Rogaway	1996 年
RSA-PSS96-R 署名方式	メッセージ復元型		
RSA-PSS98 署名方式	署名添付型	Bellare、Rogaway	1998 年
RSA-PSS98-R 署名方式	メッセージ復元型		
RSA-PSS2000 署名方式	署名添付型	（IEEE P1363）	2000 年
RSA-PSS2000-R 署名方式	メッセージ復元型		

これらのバージョンの差異はさほど大きなものではないが、RSA-PSS 署名方式の安全性評価や実装に当っては、バージョンの違いについても認識しておく必要がある。以下では、各バージョンにおける署名変換データの生成方法にスポットを当てて、その差異を整理することとする。

イ．RSA-PSS96(-R)署名方式から RSA-PSS98(-R)署名方式への改良

RSA-PSS98 署名方式は、RSA-PSS96 署名方式が提案されてから 2 年後に Bellare と Rogaway が提案したものであり（Bellare and Rogaway［1998］）、RSA-PSS96 署名方式で利用される 2 つの拡張関数 g_1、g_2 を 1 つの拡張関数 g で代替し、署名生成・検証時に必要なハッシュ演算の数を削減している点が特徴である。RSA-PSS96 署名方式と RSA-PSS98 署名方式の署名生成方法は図 11 のとおりである。

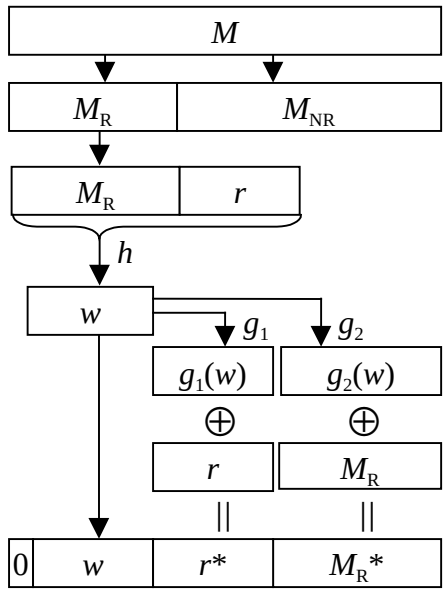
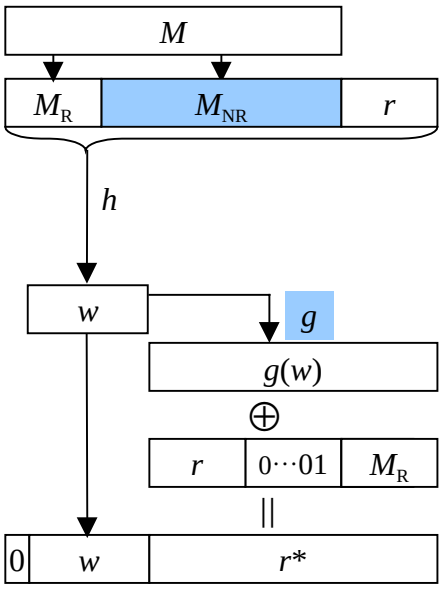
図 11：RSA-PSS96 署名方式と RSA-PSS98 署名方式の署名生成方法

＜RSA-PSS96 署名方式＞	＜RSA-PSS98 署名方式＞
平文を M（k bit）とし、乱数を r（k_r bit）を生成する。k_h bit のハッシュ値を出力するハッシュ関数を h とし、k_h bit のデータを入力すると k_r bit のデータを出力する拡張関数を g_1、k_h bit のデータを入力すると $(k-k_r-k_h-1)$ bit のデータを出力する拡張関数を g_2 とする。 以下の計算①～③を行い、デジタル署名 S を生成する。 ①$w = h(M \| r)$, ②$r^* = g_1(w) \oplus r$, ③$S = U(M)^d \bmod n$ （ただし、$U(M) = (0 \| w \| r^* \| g_2(w))$）	平文を M（k bit）とし、乱数を r（k_r bit）を生成する。k_h bit のハッシュ値を出力するハッシュ関数を h とし、k_h bit のデータを入力すると $(k-k_h)$ bit のデータを出力する拡張関数を g とする。 以下の計算①～③を行い、デジタル署名 S を生成する。 ①$w = h(M \| r)$, ②$r^* = g(w) \oplus (r \| 0\cdots0)$, ③$S = U(M)^d \bmod n$ （ただし、$U(M) = (0 \| w \| r^*)$）
	

RSA-PSS98-R 署名方式においても、RSA-PSS96-R 署名方式で利用していた 2 つの拡張関数 g_1、g_2 を 1 つの拡張関数 g で代替することによって、署名生成・検証時の効率性の向上が図られている。また、メッセージのうち、署名からメッセージを復元できる部分を M_R、メッセージを復元できない部分を M_{NR} とすると、

RSA-PSS98-R 署名方式では、署名生成時に RSA-PSS96-R 署名方式では利用されていなかった M_{NR} を利用している。RSA-PSS98-R 署名方式と RSA-PSS96-R 署名方式の署名生成方法は図 12 のとおりである。

図 12：RSA-PSS96-R 署名方式と RSA-PSS98-R 署名方式の署名生成方法

＜RSA-PSS96-R 署名方式＞	＜RSA-PSS98-R 署名方式＞
以下の計算①~④を行い、デジタル署名 S を生成する。 ①$w = h(M_R \| r)$, ②$r^* = g_1(w) \oplus r$, ③$M_R^* = g_2(w) \oplus M_R$, ④$S = U(M)^d \bmod n$ （ただし、$U(M) = (0 \| w \| r^* \| M_R^*)$）	以下の計算①~③を行い、デジタル署名 S を生成する。 ①$w = h(M \| r)$, ②$r^* = g(w) \oplus (r \| 0\cdots01 \| M_R)$, ③$S = U(M)^d \bmod n$ （ただし、$U(M) = (0 \| w \| r^*)$）
	

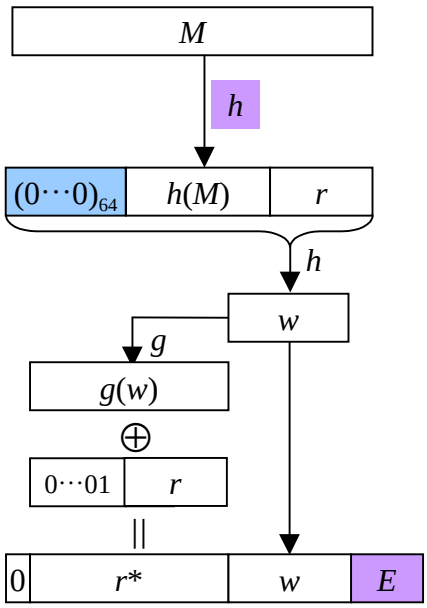
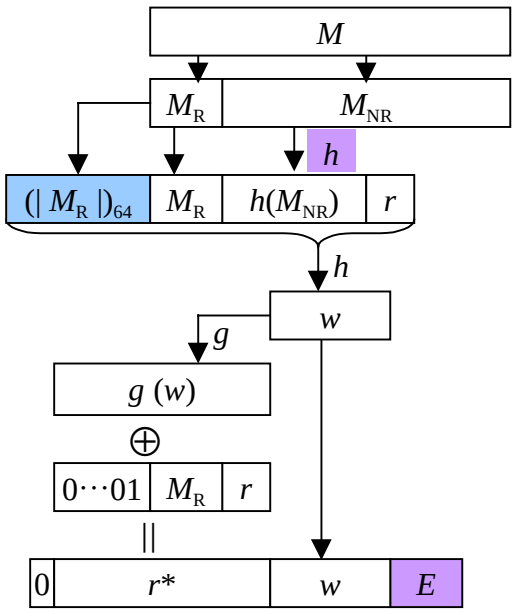
ロ．RSA-PSS98(-R)署名方式から RSA-PSS2000(-R)署名方式への改良

RSA-PSS2000(-R)署名方式は、Bellare と Rogaway が提案したものではなく、RSA-PSS98(-R)署名方式が提案されてから 2 年後に、IEEE P1363[13]において提案された署名方式である（Institute of Electrical and Electronics Engineers［2001］）。RSA-PSS2000(-R)署名方式では、ハッシュ関数 h による変換が 2 回施されている

[13] IEEE（Institute of Electrical and Electronic Engineers）P1363：IEEE は、会員数 32 万人以上を擁する電気・関連技術全般を対象分野とする学会。関連分野における新しい技術の発表の場を提供するほか、関連分野の技術を利用する際にリファレンスとなる国際標準の策定も行っている。IEEE P1363 は、公開鍵暗号技術を利用した鍵配送やデジタル署名等に関する国際標準化プロジェクトである。

が、これは、①RSA-PSS98 署名方式と RSA-PSS98-R 署名方式の両方を同一のプログラムで実装できるようにする、②安全性を向上させる、という 2 つの目的を達成するためである。RSA-PSS2000-R 署名方式において、M_R の値が 0 の場合が RSA-PSS2000 署名方式に相当する。また、RSA-PSS2000(-R)署名方式については、Bellare と Rogaway のオリジナルの証明と Coron が報告した研究（Coron［2001］）をベースにすることによって、数学的に安全性が証明可能であることが示されている（Jonsson［2000］）。RSA-PSS2000(-R)署名方式の署名生成方法は図 13 のとおりである。

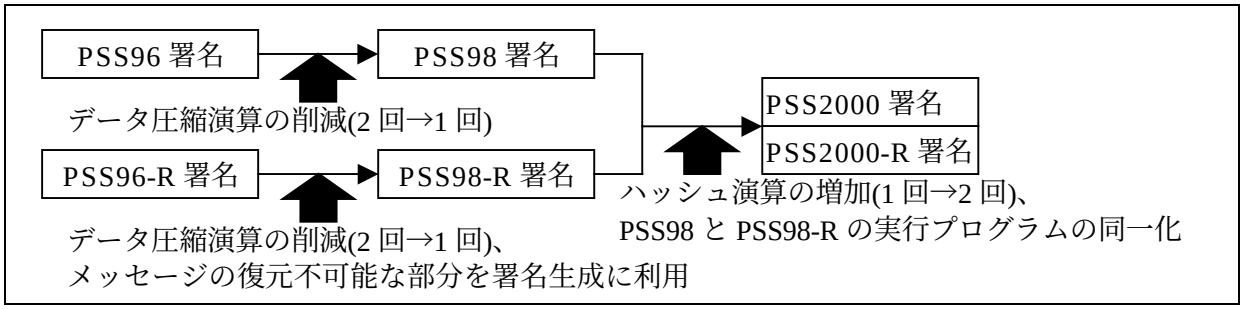
図 13：RSA-PSS2000 署名方式と RSA-PSS2000-R 署名方式の署名生成方法

＜RSA-PSS2000 署名方式＞	＜RSA-PSS2000-R 署名方式＞
E (k_E bit)を 1011 1100 (8 bit)または HashID ‖ 1011 1100 (16 bit)の定数とする。以下の計算①~③を行い、デジタル署名 S を生成する。 ①$w = h(\ (0\cdots0)_{64} \| h(M) \| r)$, ②$r^* = g(w) \oplus (0\cdots01 \| r)$, ③$S = U(M)^d \bmod n$ （ただし、$U(M) = (0 \| r^* \| w \| E)$ ）	E (k_E bit)を 1011 1100（8 bit）または HashID ‖ 1011 1100　(16 bit)の定数とする。以下の計算①~③を行い、デジタル署名 S を生成する。 ①$w = h(\ (\lvert M_R \rvert)_{64} \| M_R \| h(M_{NR}) \| r)$, ②$r^* = g(w) \oplus (0\cdots01 \| M_R \| r)$, ③$S = U(M)^d \bmod n$ （ただし、$U(M) = (0 \| r^* \| w \| E)$ ）
	

（注）$(0\cdots0)_{64}$：「0」を 64 bit 分並べたもの。
$(|M_R|)_{64}$：M_R のサイズを 64 bit で表現したもの。

以上説明してきたRSA-PSS署名方式のバージョンの推移をまとめると、図14のとおりである。

図14：RSA-PSS署名方式のバージョンの推移



(4)標準化プロジェクト等におけるRSA-PSS署名方式の採用状況

現在、RSA-PSS 署名方式は様々な標準化プロジェクト等の対象となりつつある（表10参照）。以下では、CRYPTREC、NESSIE、PKCS#1 Ver.2.1、ISO 9796-2の動向について見ておくこととする。

表10：各標準化プロジェクトに提案されているRSA-PSS署名方式

	RSA-PSS		
	96	2000	2000-R
CRYPTREC（2000年）	○	―	―
CRYPTREC（2001年）	―	○	―
NESSIE	―	○	―
PKCS#1 Ver.2.1	―	○	―
ISO 9796-2	―	―	△

（注）△：RSA-PSS署名方式をベースに策定作業が行われている。

イ．CRYPTREC

CRYPTRECは、わが国の電子政府システムで利用される暗号アルゴリズムの安全性評価を行うことを目的としたプロジェクトである。2000年4月から活動を開始し、2001年3月には、暗号技術評価報告書（CRYPTREC Report 2000）を完成させている。CRYPTRECにおける暗号の安全性評価の対象は広範であり、共通鍵ブロック暗号に加え、ストリーム暗号、デジタル署名方式、公開鍵暗号方式、疑似乱数生成も対象とされている。CRYPTRECは、2000年度に引続き2001年度も活動を続行しており、安全性評価を行った暗号技術の中から、電子政府システムで利用される暗号アルゴリズムに関する推奨暗号リストを作成するこ

とが予定されている。

CRYPTREC は 2000 年度の活動において、RSA 社から提案された RSA-PSS96 署名方式の評価を行い、CRYPTREC Report 2000 には「RSA 関数の一方向性およびランダム・オラクル・モデルを仮定することにより、適応的選択文書攻撃に対して存在的偽造が不可能であることが証明されている。RSA-PSS 署名方式の安全性は、RSA 自体の安全性、並びに、素因数分解問題の困難性に大きく依存するため、これらの問題を解くアルゴリズム、及び、計算機能力には、今後も十分に注意を払っていく必要がある」と記述されている。なお、2001 年度は RSA-PSS2000 署名方式の評価が行われている。

ロ．NESSIE

NESSIE（New European Schemes for Signature, Integrity and Encryption）は、欧州域内で利用される暗号アルゴリズムの標準化を目指して開始されたプロジェクトであり、ノルウェー・ベルゲン大学の Knudsen やイスラエル・テクニオンの Biham らが中心となって、暗号アルゴリズムの安全性や処理速度等に関する評価を進めている。

NESSIE における暗号アルゴリズムの評価の対象は、共通鍵ブロック暗号、ストリーム暗号、デジタル署名、公開鍵暗号アルゴリズム等である。

NESSIE に RSA-PSS 署名方式として提案されている署名方式は､RSA-PSS2000 署名方式である。

ハ．PKCS#1 Ver.2.1

PKCS（Public-Key Cryptosystem Standard）は、RSA 社が作成する公開鍵暗号の業界標準であり、現在 PKCS #1 をはじめとする様々な仕様が定められている。PKCS シリーズは、RSA 暗号／署名方式を利用する際のデータ変換方法、守秘、署名、鍵管理等について仕様を定めている。ただし、PKCS は RSA 社が独自に制定する業界標準であって、国際的な標準化検討委員会等によって定められた標準ではないことに注意が必要である。

PKCS#1 Ver.2.1 には、RSA-PSS2000 署名方式が記載されている。

ニ．ISO 9796-2

ISO9796-2 は、汎業界向けの情報セキュリティ技術の標準化を担当する ISO/IEC JTC1/SC27 が策定したデジタル署名の国際標準の一つである。本標準は、ハッシュ関数とパディングを利用したメッセージ復元型のデジタル署名方式を

規定している。

ISO の標準は 5 年ごとに見直しが行われるが、本標準は現在その見直し時期を迎えており、SC27 では RSA-PSS2000-R 署名方式を参考にしながら、改訂作業を進めている。

6．おわりに

本稿では、RSA 署名方式の変遷と標準化動向について解説した。これまでの RSA 署名方式に関する研究の流れを見ると、公開鍵 n のサイズおよび素因数の性質や、署名変換データの構成方法に応じて、様々な攻撃法が提案されてきたことが分かる。RSA 署名方式は、このような攻撃とこれを踏まえた改良が繰り返されることによって発展してきたといえよう。現在最も広く利用されている RSA 署名方式は、PKCS#1-1.5 署名方式であると見られる。PKCS#1-1.5 署名方式に対しては、これまでのところ、その署名変換データの構成方法に関する安全性上の問題点は指摘されておらず、十分なサイズの公開鍵を利用する限り、安全であると見られている。しかし、PKCS#1-1.5 署名方式は、安全性が数学的に証明されている訳ではなく、将来、有効な攻撃法が発見されないとも限らない。このような背景を踏まえ、ISO や IEEE 等において、安全性が証明可能な署名方式である RSA-PSS 署名方式の標準化が進められている。安全性が証明可能な署名方式は、利用者の立場からもより強い信頼性や安心感を得ることができるものと考えられる。

こうした事情を踏まえると、RSA 署名方式については、今後、RSA-PSS 署名方式の利用が拡大していくことも考えられる。ただし、RSA-PSS 署名方式においても、実装上の観点から改良が行われてきており、いくつかのバージョンが存在している。学術的な意味で RSA-PSS 署名方式という場合、Bellare と Rogaway の論文で最初に提案された RSA-PSS96 署名方式を指すことが多いが、実際に標準化等の対象とされているものは RSA-PSS2000 署名方式である。RSA-PSS 署名方式の安全性評価や実装に当っては、このようなバージョンの違いにも注意する必要があろう。

【参考文献】

宇根正志、「RSA 署名に対する新しい攻撃法の提案について―Coron-Naccache-Stern の攻撃法―」、『金融研究』第 18 巻別冊第 1 号、pp.51-83、日本銀行金融研究所、1999 年 9 月

―――、「金融分野における PKI：技術的課題と研究・標準化動向」、第 4 回情報セキュリティシンポジウム提出論文、日本銀行金融研究所、2002 年 2 月

―――・岡本龍明、「公開鍵暗号の理論研究における最近の動向」、『金融研究』第 18 巻第 2 号、pp.195-251、日本銀行金融研究所、1999 年 4 月

―――・―――、「最近のデジタル署名における理論研究動向について」、『金融研究』第 19 巻別冊第 1 号、pp.55-104、日本銀行金融研究所、2000 年 4 月

岡本龍明・山本博資、『現代暗号』、産業図書、1997 年

楠田浩二・櫻井幸一、「公開鍵暗号方式の安全性評価に関する現状と課題」、日本銀行金融研究所ディスカッションペーパーシリーズ、No.97-J-11、日本銀行金融研究所、1997 年 7 月

情報処理振興事業協会セキュリティセンター、『暗号技術評価報告書 CRYPTREC REPORT 2000』、2001 年 3 月

Bellare, M., and P. Rogaway, "The exact security of digital signatures― How to sign with RSA and Rabin", *Proceedings of EUROCRYPT '96*, LNCS 1070, pp.10-18, Springer-Verlag, 1996.

―――, and ―――, "PSS: Provably Secure Encoding Method for Digital Signature," Submission to IEEE P1363, 1998.

Blaze, M., W. Diffie, R. L. Rivest, B. Schneier, T. Shimomura, E. Thompson, and M. Wiener, "Minimum key Lengths For Symmetric Ciphers To provide Adequate Commercial Security," *A Report By An Ad Hoc Group Of Cryptographers And Computer Scientists*, January 1996.

Bleichenbacher, D., "Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS#1," *Proceedings of CRYPTO '98*, LNCS 1462, pp.1-12, Springer-Verlag, 1998.

Buchmann, J. A., *Introduction to Cryptography*, Springer-Verlag, 1999. (ヨハネス・ブーフマン著、林芳樹訳、『暗号理論入門』、シュプリンガー・フェアラーク、2001 年)

Coron, J.S., "Optimal security proofs for PSS and other signature schemes", 2001. (http://eprint.iacr.org/2001/062.pdf)

―――, D. Naccache, and J. P. Stern "On the Security of RSA Padding," *Proceedings of CRYPTO '99*, LNCS 1666, pp.1-18, Springer-Verlag, 1999.

Davis, J. A., and D. B. Holdridge, “Factorization using the quadratic sieve algorithm,” *Proceedings of CRYPTO ’83*, pp.103-113, Springer-Verlag, 1983.

De Jonge, W., and D. Chaum, “Attacks on Some RSA Signatures,” *Proceedings of CRYPTO ’85*, LNCS 218, pp.18-27, Springer-Verlag, 1986.

Desmedt, Y. G., and A. M. Odlyzko, “A chosen text attack on the RSA cryptosystem and some discrete logarithm schemes,” *Proceedings of CRYPTO ’85*, LNCS 218, pp.516-522, Springer-Verlag, 1986.

Girault, M., and J. F. Misarsky, “Selective Forgery of RSA Signatures Using Redundancy,” *Proceedings of EUROCRYPT ’97*, LNCS 1233, pp.495-507, Springer-Verlag, 1997.

Institute of Electrical and Electronics Engineers, “IEEE P1363a/D9 (Draft Version 9) Standard Specifications for Public Key Cryptography: Additional Techniques,” June 2001.

Jonsson, J., “Security Proofs for the RSA-PSS Signature Scheme and Its Variations－Draft 1.1,” 2001. (http://eprint.iacr.org/2001/053.pdf)

Koblitz, N., *A Course in Number Theory and Cryptography*, Springer-Verlag, 1997. (ニール・コブリッツ著、櫻井幸一訳、『数論アルゴリズムと楕円暗号理論入門』、シュプリンガー・フェアラーク、1997 年)

Lehman, R. S., “Factoring large integers,” *Mathematics of Computation*, Vol.28, pp.637-646, 1974.

Lenstra, A. K., H. W. Jr. Lenstra, M. S. Manasse, and J. M. Pollard, “ The number field sieve,” *Proceedings of ACM Annual Symposium on Theory of Computing*, pp.564-572, 1990.

———, and E. R. Verheul,“ Selecting Cryptographic Key Sizes,” *Journal of Cryptology*, Vol.14, No.4, pp.255-293, Springer-Verlag, 2001.

Lenstra, H. W. Jr., “Factoring Integers with Elliptic Curves,” *Annals of Mathematics*, Vol.126, pp.649-673, 1987.

Menezes, A. T., P. C. Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptology*, CRC Press, 1997.

Misarsky, J. F., “A Multiplicative Attack Using LLL Algorithm on RSA Signatures with Redundancy,” *Proceedings of CRYPTO ’97*, LNCS 1294, pp.221-234, Springer-Verlag, 1997.

———, “How (Not) to Design RSA Signature Schemes,” *Proceedings of PKC ‘98*, LNCS 1431, pp.14-28, Springer-Verlag, 1997.

Pollard, J. M. “Theorem on factorization and primality testing,” *Mathematical Proceedings of the Cambridge Philosophical Society*, Vol.76, pp.521-528, 1974.

———, “A Monte Carlo method for factorization,” *BIT*, Vol.15, pp.331-334, 1975.

Pomerance, C., “Analysis and comparison of some integer factoring algorithms,” in *Number Theory and Computers, Math. Centrum Tracts*, No.154, Part Ⅰ and No.155, Part Ⅱ, pp.89-139, 1983.

Rivest, R.L., A. Shamir, and L. Adleman, “A method of obtaining digital signatures and public key cryptosystems,” *Communications of the ACM*, Vol.21, No.2, pp.120-126, 1978.

RSA Laboratories, “PKCS#1 v1.5: RSA Cryptography Standard,” 1993.

———, “PKCS#1 v2.0: RSA Cryptography Standard,” 1998.

———, “PKCS#1 v2.1: RSA Cryptography Standard,” 2001.

———, “RSA-PSS Signature Scheme with Appendix,” Submission to the NESSIE project, September 2000.

Shor, P., “Algorithms for Quantum Computation: Discrete Logarithms and Factoring,” *Proceedings of 35th Annual Symposium on Foundations of Computer Science*, pp.124-134, 1994.

Silverman, R. D. , “A Cost-Based Security Analysis of Symmetric and Asymmetric Key Lengths,” *RSA Laboratories Bulletin*, No.13, April 2000 (Revised November 2001). (http://www.rsasecurity.com/rsalabs/bulletins/bulltein13.html)

Silverman, J. H., and J. Tate, *Rational Points on Elliptic Curves*, Springer-Verlag, 1994. (J. H.シルヴァーマン､J.テイト著、足立恒雄、木田雅成、小松啓一、田谷久雄訳、『楕円曲線論入門』、シュプリンガー・フェアラーク、1995)

Williams, H. C., “A *p*+1 method of factoring,” *Proceedings of CRYPTO ’83*, pp.87-102, Springer-Verlag, 1983.