# デジタル署名生成用秘密鍵の漏洩を巡る問題とその対策

宇根正志（うねまさし）


## 要　旨

*デジタル署名方式は、公開鍵暗号技術に基づいてデータ生成者やデータの一貫性を確認する技術である。インターネット上における電子商取引等を安全に行うためには通信相手や受信データの認証が不可欠であり、デジタル署名方式はそうした機能を果たす技術として活用されている。*

*デジタル署名方式を利用する際には、署名生成用秘密鍵を秘密に管理することが前提となっている。したがって、秘密鍵は内部のデータを物理的・論理的に保護する機構をもつICカード等のハードウエアに格納されることが望ましい。しかし、そうした場合においても、鍵管理方法の欠陥に加え、ハードウエア自体の欠陥や署名方式の欠陥によって秘密鍵が漏洩する可能性は否定できない。*

*通常のデジタル署名方式を利用している限り、漏洩した秘密鍵によって署名が偽造されるとその検知は不可能であり、なりすまし等の不正が行われるおそれがある。こうした不正が発生すると、署名を再度生成しなおす必要が出てくるほか、当該電子認証サービスの信頼が大きく損なわれると考えられる。電子認証サービスを提供していく際には、秘密鍵漏洩の影響を十分考慮し、必要な対策を検討していくことが望まれる。*

*本稿では、まず、デジタル署名生成用の秘密鍵管理に関するPKIの役割と問題点、秘密鍵漏洩の可能性とその影響について整理する。そのうえで、秘密鍵の漏洩を前提とした署名偽造への対策技術について説明し、それらの効果、実現方法、想定環境、セキュリティ要件の比較を行う。*


**キーワード：**秘密鍵漏洩、認証機関、PKI、デジタル署名、電子認証

本稿は、2003年3月7日に日本銀行で開催された「第5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。本稿で示されている内容および意見は筆者個人に属し、日本銀行の公式見解を示すものではない。

宇根正志　日本銀行金融研究所研究第2課（E-mail: masashi.une@boj.or.jp）

## １．はじめに

デジタル署名方式は、公開鍵暗号技術に基づいてデータ送信者や交信データの一貫性の確認を実現する技術であり、インターネットにおける電子商取引を安心して行うために不可欠な技術の1つとなっている。特に、わが国では、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、一定の条件のもと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もつようになり、デジタル署名に対する注目度も一段と高まっている。

デジタル署名方式を利用する際には、署名生成用の秘密鍵をその所有者が厳格に管理し、他者に知られないようにしなければならない。通常のデジタル署名方式を利用する限り、秘密鍵が攻撃者に知られてしまうと、攻撃者がその秘密鍵を用いて署名を偽造し、その秘密鍵の所有者になりすますことが可能になる。また、秘密鍵の漏洩を検知できたとしても正当に生成された署名と偽造された署名を第三者が判別することは不可能であるため、正当な署名を含めて、その秘密鍵によって生成されたすべての署名が信頼できないものとなる。こうした問題が発生すると、新しい秘密鍵と公開鍵を生成したうえで、既存の正当な署名を生成しなおす必要が出てくるほか、PKI（public key infrastructure）・電子認証サービス自体の信頼が大きく低下すると考えられる。

秘密鍵は、一般に、フロッピー・ディスク、パソコンのハード・ディスク、ICカード等に格納して管理される。ただし、フロッピー・ディスクのように内部のデータを保護するための特別な機構を有しない単なる記憶媒体に秘密鍵を格納する場合、データを読み出すことが比較的容易であり、秘密鍵が漏洩する可能性が高い。このため、ハードウエア内部の機密データを物理的あるいは論理的に保護する機構を有し、ハードウエア内部で署名を生成可能なICカード等を利用することが望ましい。こうしたハードウエアにおける具体的な保護機構としては、例えば、ハードウエアのカバーをこじ開けたり化学溶液によって溶かしたりした場合、カバーに設置されたセンサーで攻撃を検知し、メモリー上のデータを自動的に消去するといった対策がよく知られている（Anderson［2001］）。また、秘密鍵の管理方法としては、1人のハードウエア操作者が攻撃者と結託して秘密鍵を不正使用できないようにするために、署名生成を行う際には複数のハードウエア操作者が協力しなければならないようにするといった対策もある。こうした保護機構を有するICカード等のハードウエアが秘密鍵を格納する媒体として今後主流になるとみられる。

しかし、こうしたハードウエアに秘密鍵を格納したとしても、秘密鍵の漏洩を完全に防止することは困難である。例えば、デジタル署名方式のアルゴリズムやその実装方法に欠陥が存在し、それに気づいた攻撃者が公開鍵等の情報から秘密鍵を効率的に推定する可能性があるほか、ハードウエアのセキュリティ対策に欠陥があり、攻撃者がハードウエアのメモリーから直接秘密鍵を盗み出す特殊な方法を考案する可能性もある。また、署名方式やハードウエアが安全であっても、

秘密鍵の管理方法に問題が存在し、攻撃者がハードウエアの操作権限を有する者と結託してハードウエアから秘密鍵を盗み出す可能性も否定できない。

秘密鍵が漏洩する可能性があり、その影響が大きいと考えられる以上、仮に秘密鍵が漏洩して署名が偽造されたとしても署名偽造を事後的に検知可能にするなど、秘密鍵漏洩の影響を最小限にするための対策を検討する必要がある。こうした問題意識に基づき、これまでにいくつかの技術が提案されている。代表的なものとして、フォワード・セキュア署名、ヒステリシス署名、実行ハードウエア確認タグ付きデジタル署名、MAC（message authentication code）付きデジタル署名が挙げられる。これらの技術は現時点で研究途上にあり、直ちに実務で使用されているシステムに適用可能であるというわけではない。しかし、電子認証サービスの提供を検討していく場合、秘密鍵漏洩の事後的な対応手段として、こうした技術における今後の研究動向に注目し、必要な対策を検討していくことが重要である。

本稿の構成は次のとおりである。まず、2節においてデジタル署名方式やPKIについて概説したうえで、秘密鍵が漏洩した場合にどのような問題が生じるかについて説明する。3節では、秘密鍵が漏洩する形態を分類したうえで、各形態による秘密鍵の漏洩を防止する技術とその限界について具体的事例を交えながら説明する。4節では、秘密鍵漏洩に対する事後的な対策技術を紹介し、各技術の効果、実現方法、想定環境、セキュリティ要件に関して比較を行う。最後に5節で簡単に結びを述べる。

## 2．秘密鍵漏洩対策におけるPKIの問題点

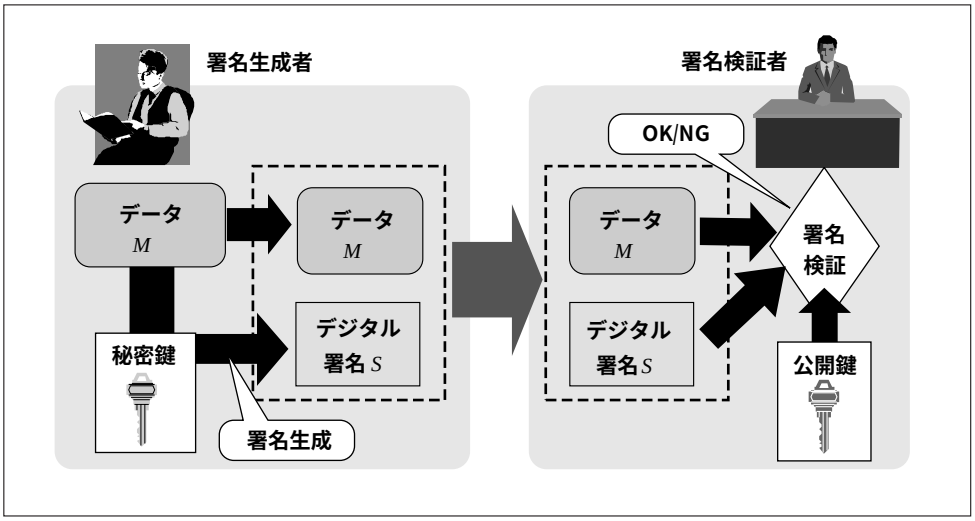
### (1) デジタル署名と公開鍵暗号

デジタル署名は、公開鍵暗号技術を利用することで、署名生成者の確認（ユーザー認証）と、署名対象データの一貫性（integrity）[1]の確認（メッセージ認証）を実現可能とする（Menezes, Oorschot and Vanstone［1997］）。公開鍵暗号は、暗号化用の鍵と復号用の鍵が異なる暗号であり、ある特定のデータが得られない状況で暗号化用の鍵から復号用の鍵を算出すること、および、復号用の鍵を用いることなく暗号文から平文を求めることが計算量的に困難である[2]という性質を有している。暗号化用の鍵（公開鍵）は公開される一方、復号用の鍵（秘密鍵）は秘密鍵の所有者が秘密に管理する必要がある。

1 “integrity”は、「一貫性」のほか、「完全性」と訳される場合もある。

2 「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。公開鍵暗号では、「一方の鍵から他方の鍵を導出することが計算量的に困難」という状況を、効率的に解を求めることが困難とみられている数学の問題を利用して実現している。

**図1　デジタル署名の生成と検証**




デジタル署名方式では、秘密鍵を署名生成用の鍵として利用し、公開鍵を署名検証用の鍵として利用する。あるデータ*M*に対するデジタル署名*S*は、秘密鍵を用いて*M*を一定のアルゴリズム（署名生成アルゴリズム）で変換して生成される（図1参照）。秘密鍵が文字どおり秘密に管理されている限り、秘密鍵を用いた署名生成の変換はその所有者のみが実行可能となる。また、*S*と*M*を入手した署名検証者は、これらのデータと署名生成者の公開鍵を用いて一定のアルゴリズム（署名検証アルゴリズム）を実行することによって*S*の検証を行う。デジタル署名の検証は、その署名の生成に用いられた秘密鍵に対応する公開鍵によって実行可能となる。*S*が公開鍵に対応する秘密鍵によって適切に生成されたものでない場合、あるいは、*S*が生成された後に*M*が改ざんされた場合には、署名検証は成功しない。このため、公開鍵が確かに署名生成者のものであり、かつ、秘密鍵が適切に管理されているという状況において、*S*の検証が成功すれば、「*S*は公開鍵・秘密鍵の所有者によって生成されたものであり、*S*が生成された後に*M*が改ざんされていない（一貫性、あるいは、完全性が確保されている）」ということが確認される。

## （2）PKIの役割

デジタル署名を適切に利用するためには、公開鍵・秘密鍵の所有者を特定すると同時に、その公開鍵に対する秘密鍵が適切に管理されていることを確認する必要がある。PKIはこれらの問題に対応する手段を提供する[3]。

PKIでは、公開鍵・秘密鍵の所有者や署名検証者から信頼される第三者（認証機

3　金融分野におけるPKI関連の技術研究・標準化動向については宇根［2002］を参照。

関、CA：certification authority）が、公開鍵・秘密鍵とその所有者を結び付ける公開鍵証明書（public key certificate）を生成・発行する。公開鍵証明書には、公開鍵や公開鍵・秘密鍵の所有者の識別子に加え、公開鍵証明書の有効期間、公開鍵証明書を発行した認証機関の識別子等のデータが含まれるほか、これらのデータが改ざんされていないことを確認可能にするために認証機関のデジタル署名が含まれる。

認証機関は、公開鍵・秘密鍵の所有者に対して、「秘密鍵が漏洩した」と考えられる場合や、秘密鍵が何らかの原因で利用できなくなった場合にはその旨を認証機関に直ちに通知し、公開鍵証明書の失効申請を行うことを求める。認証機関は、失効申請等に基づいて失効した公開鍵証明書のリストを更新し、定期的または要求に応じて公開鍵証明書の失効情報を配布する。公開鍵証明書の失効情報の主な配布方法としては、公開鍵証明書失効リスト（CRL：certificate revocation list）を配布する方法と、特定の公開鍵証明書の有効性に関する問合せにリアルタイムで回答するOCSP（online certification status protocol）と呼ばれるプロトコルを利用する方法の2つが挙げられる。

公開鍵を利用する際には、こうした仕組みを前提として、①その公開鍵に対応する公開鍵証明書を入手したうえで、②CRL等の失効情報を用いて公開鍵証明書が有効であることを確認し、③予め入手していた認証機関の公開鍵を用いて公開鍵証明書に含まれる認証機関のデジタル署名を検証するという手続を実行することとなる。これらの手続が成功した場合、公開鍵証明書が有効であり、その公開鍵に対応する秘密鍵が適切に管理されているとみなされる。

### （3）秘密鍵漏洩によるデジタル署名の偽造の影響

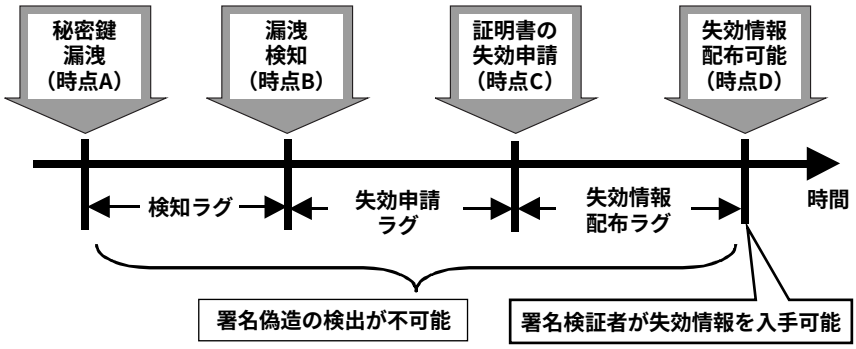
デジタル署名を実現するうえで、PKIは公開鍵・秘密鍵とその所有者を対応づけるという重要な役割を果たす。しかし、秘密鍵が何らかの原因によって漏洩し、攻撃者がそれを入手した場合、PKIの枠組みだけでは解決困難な問題が存在する。

#### イ．偽造されたデジタル署名の不正使用

まず、秘密鍵が攻撃者に漏洩した時点から、対応する公開鍵証明書の失効情報が配布されるまでにタイムラグが存在し、その間、偽造されたデジタル署名が正当なデジタル署名として誤って受け入れられてしまう可能性がある（小森・松浦・須藤［2001a］）。

こうしたタイムラグとして次の3種類が存在する（図2参照）。まず、攻撃者が秘密鍵を入手した時点（時点A）から、その秘密鍵の所有者が秘密鍵漏洩の事実に気づく（時点B）までの間にタイムラグが存在する（以下、検知ラグという）。例えば、攻撃者がハードウエアの操作者と結託し、ハードウエアにおける鍵管理の不備を巧みに利用してハードウエアから秘密鍵を盗み出した場合、署名偽造による具体的な被害が表面化しない限り秘密鍵の漏洩は検知されがたいと考えられる。こうしたケースでは検知ラグが大きくなる。

**図2　失効処理に伴うラグ**




備考：小森・松浦・須藤［2001］の図1を基に作成。

また、秘密鍵の所有者が秘密鍵の漏洩を検知して（時点B）から、その秘密鍵に対応する公開鍵証明書の失効申請を認証機関に対して行う（時点C）までの間にタイムラグが存在する（以下、失効申請ラグという）。ただし、IETF（Internet Engineering Task Force）の証明書ポリシーのガイドラインRFC 2527（Chokhani *et al.*［2001］）等では、「秘密鍵の所有者は、秘密鍵の漏洩を検知した場合、直ちに認証機関に報告しなければならない」という項目を証明書ポリシーに織り込むことが記述されており、こうした規定が守られるならば失効申請ラグを小さくすることができる。

公開鍵証明書の失効申請を認証機関に対して行って（時点C）から、認証機関がその公開鍵証明書を失効させて失効情報をCRLに反映する、または、OCSPによって失効情報を配信可能な状態にする（時点D）までの間にもタイムラグが存在する（以下、失効情報配布ラグという）。CRLを定期的に更新して失効情報を配布するケースでは、失効情報がリアルタイムでCRLに反映されず、時点Cのタイミングによっては失効情報配布ラグが大きくなる。OCSPによって失効情報を配布するケースでは、署名検証者が失効情報をリアルタイムで入手可能であり、失効情報配布ラグを小さくすることができる。

### ロ．既存の正当なデジタル署名の信頼喪失

秘密鍵が漏洩した直後にその秘密鍵に対応する公開鍵証明書の失効情報を署名検証者に配布できたとしても（すなわち、認知や失効処理のラグを無視できるほど小さくすることができたとしても）、「秘密鍵が漏洩する以前にその秘密鍵によって正当に生成されたすべての署名の信頼が失われてしまう」という深刻な問題の発生を回避することは困難である。

こうした既存の署名の信頼喪失は、デジタル署名が押印や手書き署名に代わって文書の真正性を担保する手段として広く利用されているような環境においては、大

きな影響をもたらすこととなる。署名生成者にとっては、新しい秘密鍵・公開鍵ペアを生成し、その公開鍵に対応する公開鍵証明書を認証機関から入手したうえで、信頼を喪失した既存の署名のすべてについて新しい秘密鍵を用いた署名を生成しなおすことを余儀なくされる。また、当該電子認証サービスの提供者にとっては、それまでに勝ち得ていた安全性に対する信頼が損なわれることにつながる。こうした影響によって、例えば、電子認証サービスを活用したさまざまな電子的な取引を安心して行うことができなくなるほか、過去の署名付きデータが信頼できなくなる等、社会に大きな損失をもたらすことも懸念される。

## 3．秘密鍵の漏洩を未然に防ぐための対策と限界

### (1) 秘密鍵漏洩の形態

以下では、秘密鍵を格納する媒体として、内部の機密データを保護すると期待される機構を有し、内部で署名生成処理等を実行可能なICカード等のハードウエアを想定する。こうしたハードウエアを採用する場合、次のような形態で秘密鍵が漏洩する可能性がある[4]。

#### イ．漏洩形態1：鍵管理方法の欠陥

漏洩形態1としては、鍵管理の方法自体は安全であるものの、運用が適切に行われていない場合や、鍵管理の方法自体にセキュリティ上の欠陥が存在する場合において、攻撃者がそれらの事実に気づいて秘密鍵をハードウエアから盗み出すという状況が想定される。

鍵管理の運用が適切に行われていないケースとして次のようなものが考えられる。例えば、秘密鍵をハードウエア外部で生成し、ハードウエアに入力するという方法が利用される場合を想定する[5]。このとき、秘密鍵をハードウエアに入力した後でハードウエア外部に存在する秘密鍵のコピーが完全に消去されず、攻撃者の手に渡ってしまう事態が考えられる。また、ハードウエアに秘密鍵を入力・設定するコマンドや秘密情報が適切に管理されておらず、攻撃者がそれらの情報を入手し、

4 攻撃者がハードウエアとそれを実行可能にするデータ（例えばパスワード）を入手した場合も、攻撃者はハードウエアの正当な持ち主になりすまして署名を偽造可能である。このようなタイプのなりすましへの対策として、バイオメトリック認証技術等、さまざまな本人認証技術の研究が行われている（中山・小松［2000］）。こうした本人認証技術は署名偽造対策の要素技術の1つとして位置づけることができるが、本稿では、秘密鍵の漏洩にともなう署名偽造への対策技術に焦点を当てて検討を行っており、本人認証技術に関しては検討対象外とする。

5 例えば、代表的な暗号ハードウエア・モジュールであるIBM 4758では、RSA署名用の秘密鍵・公開鍵をハードウエア内部で生成する機能を有しているだけでなく、外部から鍵を入力する機能も有しており、アプリケーションに応じて鍵の設定方法を選択できるようになっている（IBM［2001］）。

ハードウエアの所有者になりすまして新たに秘密鍵を入力・設定するといった事態も考えられる。

一方、鍵管理の方法自体に問題が存在するケースとしてもさまざまなものが想定される。例えば、外部で秘密鍵を生成してハードウエアに入力・設定する際に複数のハードウエア操作者が協力しないと実行できないという方法を採用したものの、プロトコルの不備によって1人のハードウエア操作者が秘密鍵の入力・設定を実行できてしまうといった事態が考えられる。

**ロ．漏洩形態2：ハードウエアの解析**

漏洩形態2としては、攻撃者のみが秘密鍵を格納するハードウエアから破壊型・非破壊型解析によって秘密鍵を盗み出す方法を発見し、ハードウエアに一時的にアクセスして秘密鍵を盗み出すという状況が想定される[6]。

破壊型解析は、ハードウエアに物理的な変形を加えることによって解析を行う攻撃法である。代表的な破壊型解析としてはプローブ解析が挙げられる（情報処理振興事業協会［2000］）。プローブ解析は、ハードウエアのパッケージを除去したうえで、内部の回路に直接プローブ（回路測定用電極）を当てて回路の電気信号を探り、秘密鍵等のデータを盗み出すというものである。

非破壊型解析は、ハードウエアに物理的な変形を加えず、モジュールの通常の利用法を応用して解析を行う攻撃法である。代表的な非破壊型解析としては、故障利用解析とサイド・チャネル解析が挙げられる。故障利用解析は、加熱や放射線照射等によってハードウエア内部の回路に意図的に故障を起こさせ、ハードウエアの正常時の出力データや故障時の出力データ等を利用して回路に格納されているデータを推定するという攻撃である（Boneh, DeMillo and Lipton［1997］ほか）。サイド・チャネル解析は、暗号処理を実行する際にハードウエアから漏洩する（正規の入出力以外の）データ（例えば、暗号処理時間、消費電力、周波数）を解析し、ハードウエア内部に格納されているデータを推定するという攻撃法である。代表的なサイド・チャネル解析としては、タイミング解析[7]、電力解析[8]、テンペスト（TEMPEST）解析[9]が挙げられる。

6 攻撃者がハードウエアにアクセスする形態としては、ハードウエアを物理的に盗用するケースのほか、ハードウエアがネットワークに接続されている状況において、そのネットワーク経由でハードウエア内部に不正アクセスするケースも考えられる。

7 **タイミング解析**：暗号処理時間が処理の最適化等のために秘密鍵に依存して変化する点に着目し、処理時間を統計的に解析してハードウエアに格納されている秘密鍵を推定するという攻撃法（Kocher［1996］ほか）。

8 **電力解析**：処理時におけるハードウエア・モジュールの消費電力が秘密鍵等のデータや処理内容と関係している点に着目し、消費電力を観察することによって秘密鍵等を推定するという攻撃法（Kocher, Jaffe and Jun［1999］ほか）。最近では、電力解析と故障利用解析を組み合わせたフォールト・パワー（fault-power）解析と呼ばれる手法も提案されている（桶屋・櫻井［2002］）。

9 **テンペスト解析**：暗号モジュールおよびその周辺機器から暗号処理中に放出される電磁波を捕捉し、その情報を基にして暗号処理の内容や秘密鍵等を推定するという攻撃法（NIST［2001］）。

### ハ．漏洩形態3：デジタル署名方式やその利用方法の欠陥

漏洩形態3として、デジタル署名方式あるいはその利用方法にセキュリティ上の致命的な欠陥があり、秘密鍵を格納するハードウエアにアクセスすることなく、公開鍵や既存の署名等の公開情報から秘密鍵を効率的に導出する方法を攻撃者のみが知るという状況が想定される。

このタイプの欠陥としては、そもそもデジタル署名のアルゴリズム自体に欠陥が存在していた場合が考えられるほか、そのアルゴリズム自体は安全であっても、実装時に作製された署名生成用プログラムに重大なバグが存在し、安全性の低いシステムになってしまう場合が考えられる。また、署名方式の安全性の根拠としていた数学的問題に対して新しい効率的な解法が考案されたり、飛躍的に高い計算能力を有するコンピュータが実用化されたりする場合も考えられる。例えば、現在盛んに研究が進められている量子コンピュータが実用化された場合、RSA署名やDSAなどの安全性の根拠となっている素因数分解問題や離散対数問題が容易に解けてしまうとみられている（例えば田中・岡本［2002］）。ただし、このような飛躍的に高い計算能力を有するコンピュータが実用化されることがなくても、コンピュータのコスト・パフォーマンスが徐々に向上した結果、公開鍵から秘密鍵を容易に導出可能となる場合も考えられる（洲崎・松本［2001］）。

## （2）秘密鍵の各漏洩形態への対策と限界

### イ．漏洩形態1への対策と限界

形態1による秘密鍵の漏洩を未然に防ぐためには、デジタル署名のアプリケーションにおけるセキュリティ要件に合致した鍵管理の方法を適切に選択するとともに、その運用を適切に実施することが必要である。しかし、鍵管理の方法自体にセキュリティ上の問題がないか否かを厳密に評価することは容易でないほか、認証機関の内部者と攻撃者との結託や秘密鍵の所有者による誤操作など、鍵管理の適切な運用が損なわれる可能性は否定できない。

定評ある暗号ハードウエア・モジュールにおいても鍵管理の方法自体にセキュリティ上の問題が存在し、ハードウエアから秘密鍵が漏洩する可能性があることを示した事例として、IBM 4758（Model 002、IBM［2002b］）における鍵管理の不備を指摘したボンドとクレイトンの研究が挙げられる（Bond［2001］、Clayton and Bond［2002］）。IBM 4758はFIPS 140-2（NIST［1994, 2001］）[10]のセキュリティ・レベル4の認定を受けた暗号モジュールであり、ハードウエア操作者の確認、鍵管理、暗号化・復号、MAC生成・検証等の処理をCCA（Common Cryptographic Architecture、

10 FIPS 140-2（Security Requirements for Cryptographic Modules）：米国連邦政府機関が利用する暗号モジュールのセキュリティ要件を規定する米国政府調達基準。NIST（National Institute of Standards and Technology）によって策定・管理されている。FIPS 140-2では、暗号モジュールに関して4段階のセキュリティ・レベル（レベル1～4）が設定されており、各レベルに応じたセキュリティ要件が規定されている。

IBM［2001, 2002a］)[11]と呼ばれるソフトウエアを用いて行う。CCAは、暗号アルゴリズムとして公開鍵暗号ではRSA暗号、デジタル署名にはRSA署名[12]を採用しているほか、共通鍵暗号としてはトリプルDES（CBCモード）を採用している。また、鍵の管理では、鍵の種類ごとに別々の鍵を生成して使用するという仕組みを利用している。例えば、通信データ（例えば暗証番号）の暗号化に関する鍵としては、通信データの暗号化用の鍵（data key）*DK*、*DK*を外部に転送する際に*DK*を暗号化するための鍵（transport key）*TK*、*TK*を暗号化するためのマスター鍵（master key）*MK*という3種類の鍵が準備される。これらの鍵はMAC生成・検証用や鍵配送用にも準備される。

ボンドらの攻撃のアイデアは、鍵の設定に関する特定の権限を有する操作者1人が攻撃者と結託するという前提のもとで、*TK*を効率的に推定し、その*TK*によってIBM 4758に格納される鍵を盗み出すというものである（読み出した鍵によって通信データ等を解読する）。攻撃の概要は次のとおりである。

**〈ボンドらによるIBM 4758への攻撃法〉**

**【想定環境】**

・通信データの暗号化用にはシングルDESが利用され、*DK*等を暗号化するための鍵*TK*には2-keyトリプルDESが利用される。*TK* = [*TK*1, *TK*2] であり、*DK*、*TK*1、*TK*2はそれぞれ64ビットである。

・シングルDESおよびトリプルDESの鍵は、2人の操作者が別々に有する鍵パーツ（key parts）の排他的論理和によって生成される。

　— *DK*の場合、2つの鍵パーツ*DKP*1、*DKP*2の排他的論理和として生成される（すなわち*DK* = *DKP*1⊕*DKP*2）。

　— *TK*の場合、2つの鍵パーツ [*TK*1*P*1, *TK*2*P*1]、[*TK*1*P*2, *TK*2*P*2] の排他的論理和として生成される（すなわち、[*TK*1, *TK*2]= [*TK*1*P*1⊕*TK*1*P*2, *TK*2*P*1⊕*TK*2*P*2]）。

・攻撃者と結託する操作者は以下のコマンドの実行権限を有している。

　<1>Combine_Key_Parts（あるいはKey_Parts_Import）：2-key トリプルDESの2つの鍵（それぞれ64ビット）を設定、また、鍵パーツの排他的論理和を計算。

　<2>Data_Key_Export：*DK*を*TK*によって暗号化し、コンソールに表示。

　<3>Load_First_Key_Parts：鍵パーツをハードウエア外部から設定。

　<4>Encipher：通信データを*DK*によって暗号化。

・攻撃者の目的は*TK*を導出することである。

11 研究対象となったCCAはRelease 2.40（IBM［2001］）であり、2002年5月に発表されたRelease 2.41（IBM［2002a］）では、指摘された脆弱性を回避するための対策が講じられている。

12 RSA暗号の利用方法としては米国金融界の国内標準ANS X9.31が参照されているほか、RSA署名の利用方法としては、ANS X9.31のほか、ISO 9796-1、PKCS #1 Ver. 1.5 or 2.0が参照されている（IBM［2001］）。

**【攻撃の手順〈3つのフェーズから構成される〉】**

**（フェーズ1　*DK*の探索〈図3参照〉）**

（1）*DK*の鍵パーツの一部*DK*1をハードウエア内部で生成する。ただし、*DK*1を攻撃者は知ることができない。

（2）“Load_First_Key_Parts”コマンドを用いて鍵パーツ*DK*2を“ 0 ”に設定する。

（3）“Combine_Key_Parts”コマンドを用いて、*DK*1と*DK*2の排他的論理和を計算し、*DK*を生成する。*DK* = *DKP*1⊕*DKP*2= *DK*1となる。

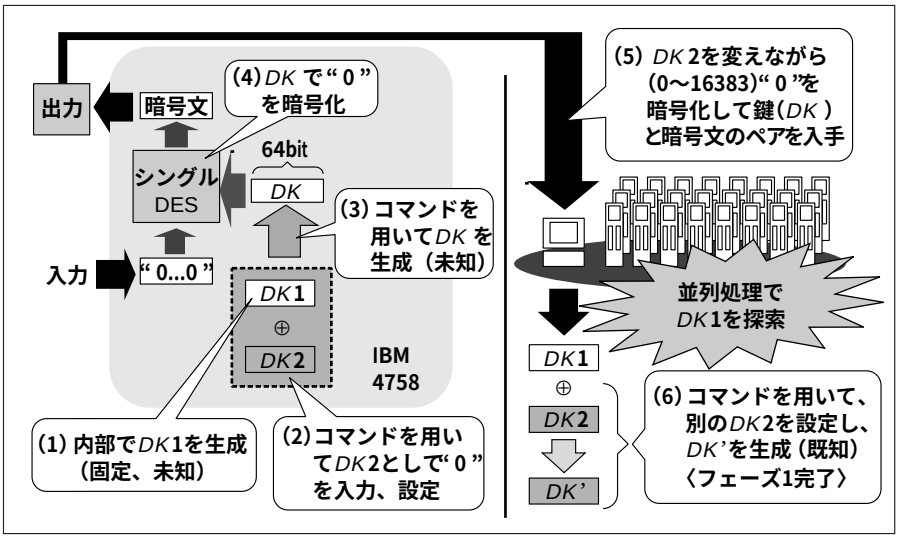
（4）“Encipher”コマンドを用いて、*DK*によってデータ“ 0 ”を暗号化する。

（5）上記（2）における*DK*2の値を“ 1 ”から“ 16383 (= $2^{14}$ – 1) ”に変えながら上記（2）～（4）の処理を繰り返し、平文“ 0 ”に対する複数の暗号文を生成する（本処理は約10分で実行可能）。

　次に、入手した複数の暗号文を用いて専用装置によって*DK*1を探索する。この結果、約1日で*DK*1を探索可能である。

（6）“Combine_Key_Parts”コマンドを用いて、（5）で得た*DK*1と新たに設定した*DK*2の排他的論理和を計算して*DK’*を新たに設定する。この結果、攻撃者はハードウエアに格納されている*DK’*を得る。

**図3　*DK*の探索・設定手順〈フェーズ1〉**




**（フェーズ2　*TK*の設定・探索〈図4参照〉）**

（7）2-keyトリプルDESの鍵パーツの1つ[*TK*1*P*1, *TK*2*P*1]を、2つの64ビット鍵が等しくなる（つまり*TK*1*P*1= *TK*2*P*1）ように設定し内部で生成する。

（8）“Load_First_Key_Parts”コマンドを用いて、2-key トリプルDESのもう1つの鍵パーツ・ペア[*TK*1*P*2 , *TK*2*P*2]を［0, 0］に設定する。

（9）“Combine_Key_Parts”コマンドを用いて、2つの鍵パーツ・ペアの排他的論理和を計算して*TK*を生成する。*TK*= [*TK*1*P*1⊕0, *TK*1*P*1⊕0]= [*TK*1*P*1, *TK*1*P*1]

となる。*TK*は、2つの64ビット鍵が同一であるため、実質的な鍵長は64 bitとなっている。

（10）“Data_Key_Export”コマンドを用いて、上記（6）で得た*DK'*を*TK*で暗号化してコンソールに表示させる。

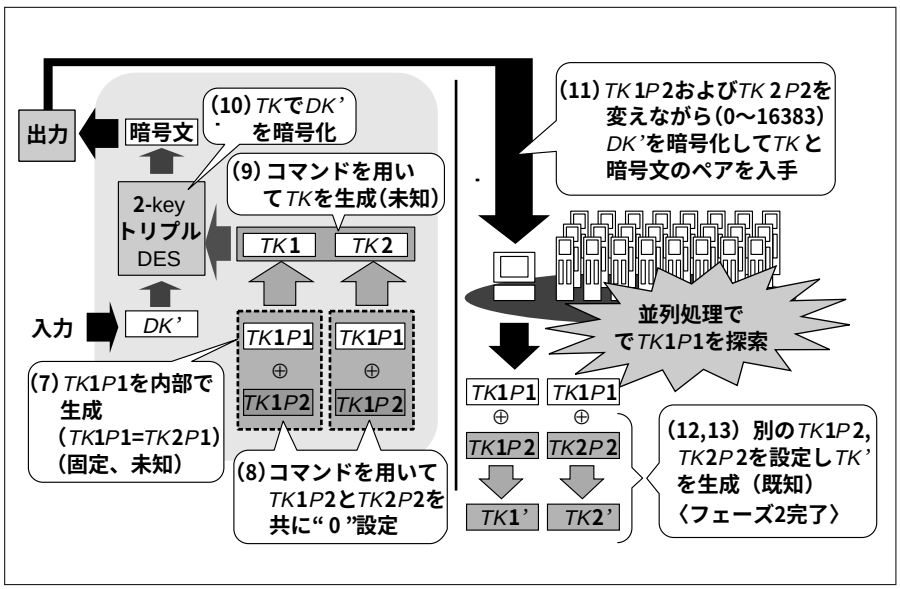
（11）上記（8）における*TK*1*P*2および*TK*2*P*2の値を“1”から“16383 (= $2^{14}-1$)”に変えながら上記（8）～（10）の処理を繰り返し、*DK'*を暗号化する（本処理は約10分で実行可能）。

次に、入手した複数の暗号文を用いて専用装置によって*TK*1*P*1を探索する（所要時間は約1日）。

（12）“Combine_Key_Parts”コマンドを用いて、鍵パーツ[*TK*1*P*2, *TK*2*P*2]（ただし*TK*1*P*2と*TK*2*P*2は互いに異なる）を新たに設定する。

（13）上記（11）で得た[*TK*1*P*1, *TK*1*P*1]と上記（12）で新たに設定した鍵パーツの排他的論理和を計算し、*TK'*(= [*TK*1*P*1⊕*TK*1*P*2, *TK*1*P*1⊕*TK*2*P*2]) を計算する。この結果、攻撃者は2-keyトリプルDESの鍵*TK'*を得る。

**図4　*TK*の探索・設定手順〈フェーズ2〉**




**（フェーズ3　*TK'*による鍵の読み出し）**

（14）“Data_Key_Export”コマンドを用いて、各種の鍵を*TK'*で暗号化してコンソールに表示させる。攻撃者は、暗号化された鍵を上記（13）で得た*TK'*で復号して入手する。

ボンドらは、上記攻撃に必要となる鍵探索用の専用装置を作製するためには約1,000ドルが必要となるほか、その専用装置を用いて上記(5)と(11)の処理を実行するのに約2日かかるとしている。また、IBM 4758に対するアクセス時間は20分程度と試算している。

このような攻撃が適用可能となるのは数種類のコマンドを実行する権限を有する操作者が攻撃者に協力した場合に限定される。しかし、鍵の設定等の処理を行う際に2つの鍵パーツを別々の操作者に配布して管理を行う仕組みを採用している以上、操作者の1人が攻撃者と結託する可能性は鍵管理方法の設計時に想定されていたはずであり、ボンドらの研究はIBM 4758の鍵管理方法に関するセキュリティ評価が十分ではなかったことを示唆している。FIPS 140-2において最上位のセキュリティ・レベルの認定を受けているIBM 4758において鍵管理方法に問題があったという事実を考慮すると、いかなる鍵管理方法においても潜在的なセキュリティ上の問題が存在し、秘密鍵が漏洩する可能性があることを改めて認識しておく必要があろう。

### ロ．漏洩形態2への対策と限界

破壊型・非破壊型解析による秘密鍵漏洩を未然に防ぐためには、こうした解析に対して十分な安全性を有するハードウエアを利用することが必要である。ハードウエアの破壊型・非破壊型解析に対する安全性を適切に評価するためには高度な専門技術と相応の設備が必要であり、ハードウエアの利用者が独自に評価を行うことは容易でない。現実的な対応策の1つとしては、専門技術と設備を有する評価機関によるハードウエアの安全性評価の結果を参考にしてハードウエアを選択することが考えられる。

暗号処理用ハードウエア・モジュールの第三者によるセキュリティ評価の代表的な枠組みとしては、米国の政府機関（NIST）とカナダの政府機関（CSE：Communications Security Establishment）が運営するCMVP（Cryptographic Module Validation Program）が挙げられる。CMVPでは、NISTやCSEから評価機関CMT（Cryptographic Module Testing laboratories）として認可を受けた第三者機関が、まず、FIPS 140-2に準拠して作製された暗号モジュールをテストし、FIPS 140-2のセキュリティ要件が満足されているか否かを評価する。その後、CMTの評価結果に基づき、NISTやCSEが「評価対象となった暗号モジュールがFIPS 140-2に準拠して作製されており、特定のレベルのセキュリティ要件を満足している」との認定を行うという仕組みとなっている。2002年3月28日の時点では、合計211の暗号モジュールがCMVPに基づく評価・認定を得ており、そのうち8つの暗号モジュールが最上位であるレベル4の認定を得ている[13]。

13 認定済みモジュールはhttp://csrc.nist.gov/cryptval/140-1/1401val.htmlで公開されている。

こうした評価・認定を受けた暗号モジュールを利用することによって、破壊型・非破壊型解析への対策に関して一定の効果を期待することができる。しかし、そうした効果にも限界があると考えられる。例えば、FIPS 140-2では、故障利用解析やサイド・チャネル解析に関して具体的なセキュリティ要件が規定されておらず、「これらの攻撃への対策が暗号モジュールに適用されている場合には、暗号モジュールのセキュリティ・ポリシーにその対策を記述すること」という要件が置かれているにすぎない。したがって、例えば、ある暗号モジュールが最新のサイド・チャネル解析に対して十分な安全性を確保しているかどうかについては、CMVPによる評価・認定だけからは判断することが困難なケースもある。

こうした状況を踏まえると、ハードウエアの厳密なセキュリティ評価を行うためには、第三者による評価・認定とは別に、暗号モジュールの仕様書等を参照して「その暗号モジュールがどのような攻撃を前提としているか、そして、どのようなセキュリティ対策を採用しているか」を自ら確認することが必要であり、また、破壊型・非破壊型解析に関する研究の動向を適切にフォローしていくことも必要である。しかし、ハードウエアの解析技術および対策技術のすべてが学会等のオープンな場で発表されるわけではなく、ハードウエア・メーカーやハードウエアの評価を行う研究所等のノウハウとして公開されていない情報も少なからず存在することを踏まえると、独自のセキュリティ評価にも限界があることは認識しておく必要がある。

**ハ．漏洩形態3への対策と限界**

デジタル署名方式やその利用方法の欠陥による秘密鍵の漏洩を未然に防ぐためには、デジタル署名方式を適切に選択することがまず必要である。デジタル署名を利用するアプリケーションのセキュリティ要件を明確にしたうえで、その要件に合致する安全性を有していると評価された署名方式の採用が求められる。さらに、デジタル署名方式の安全性が損なわれないように利用形態を選択する必要がある。具体的には、鍵長、ハッシュ値のサイズ、署名生成に用いる擬似乱数生成方法、鍵の更新頻度等を適切に選択することが挙げられる。

具体的にどのデジタル署名方式を採用するかに当たっては、学会における評価や、ISOやIEEE（Institute of Electrical and Electronics Engineers）の国際標準、ANSI等の各国国内標準、PKCS（Public Key Cryptography Standards）等の業界標準を参考にすることが考えられる。また、わが国の電子政府に利用される暗号アルゴリズムの評価を行っている暗号技術検討会・暗号技術評価委員会（CRYPTREC）の報告書も有用な情報である。なお、現在構築が進められている金融関連の主なPKI・電子認証サービスをみると、各種標準に規定されているRSA署名を利用するケースが多い（宇根［2002］）。

ただし、このようなデジタル署名方式を利用したとしても、セキュリティ上の問題点が後日指摘されるケースは少なくない。秘密鍵の漏洩に直接結びつくものではないが、国際標準等に規定されていた署名方式に対してセキュリティ上の問題点が

存在することを示す研究成果が発表された例もみられる。例えば、メッセージ復元型デジタル署名方式の国際標準ISO 9796-2（ベースはRSA署名方式）に対する攻撃法の提案や、米国政府のデジタル署名方式の標準FIPS 186-2（DSA）における問題点の指摘等が挙げられる。

ISO 9796-2のデジタル署名方式に対する攻撃については、宇根［1999］、宇根・岡本［2000］、齊藤［2002］において既に紹介されているため、本稿ではDSAにセキュリティ上の問題が発見された事例について紹介する。

**〈DSAにおけるセキュリティ上の問題点について〉**

DSAは、離散対数問題（$y, g, p$ が与えられたときに$y = g^x \bmod p$を満たす$x$を求める問題）の困難性を利用した署名方式であり、鍵生成、署名生成・検証アルゴリズムは以下のとおりである。

**【鍵生成】**素数$p$（サイズは1,024ビット）と、$p-1$の素因数$q$（サイズは160ビット程度）を選択。次に、$p$を法とする乗法群の要素$x$と$g$を選択（$g$は位数が$q$となるように選択）。最後に、$y = g^x \bmod p$を生成。署名生成用の秘密鍵を$x$、署名検証用の公開鍵を$(y, g, p, q)$とする。

**【署名生成】**疑似乱数$k$を生成し、データ$M$に対する署名$(r, t)$を以下の数式によって生成。ただし、$h$はハッシュ関数（SHA-1）。

$$r = (g^k \bmod p) \bmod q$$
$$t = (h(M) + xr)/k \bmod q$$

**【署名検証】**以下の等式が成立した場合、署名検証が成功。

$$r = (g^{h(M)/t} y^{r/t} \bmod p) \bmod q$$

DSAの特徴の1つは、署名生成時に毎回異なる疑似乱数$k$が利用され、同一の秘密鍵によって同一のデータに対するデジタル署名を生成しても、毎回異なるデジタル署名が生成されるという点にある。このようなタイプのデジタル署名は確率署名（probabilistic signature）と呼ばれる。

ベル研究所の暗号研究者であるブライヘンバッハーは、2000年11月、FIPS 186-2に規定されている擬似乱数$k$の生成アルゴリズムに問題があり、DSAのぜい弱性につながっていることを指摘した[14]。具体的には、擬似乱数生成アルゴリズムによって生成される$k$の値に偏りが存在し、同一の鍵ペアによって生成された$2^{22}$個のデジタル署名を入手することができれば、離散対数問題を解くために必要な計算量よりも少ない計算量でデジタル署名の偽造を行うことができるとしている。本攻撃法の適用可能性について、ブライヘンバッハーは、

14 プレス・リリースについてはhttp://www.lucent.com/press/0201/010205.bla.htmlを参照。

「現時点では攻撃に必要な計算量が莫大となり、DSAの安全性に対して致命的な欠陥とはいえない」としている。

DSAを開発・標準化したNISTは、擬似乱数生成アルゴリズムの改良の必要性を認め、2001年10月、改良を加えたDSAのアルゴリズム（FIPS 186-2）を発表した（NIST［2000］）。FIPS 186-2には、「改良された擬似乱数生成アルゴリズムを利用する、もしくは、一定数（約200万個）のデジタル署名を生成するたびに鍵ペアを更新することによって、ブライヘンバッハーによって指摘された問題を回避することができる」と記述されている。

## 4．秘密鍵漏洩を前提とした署名偽造対策技術

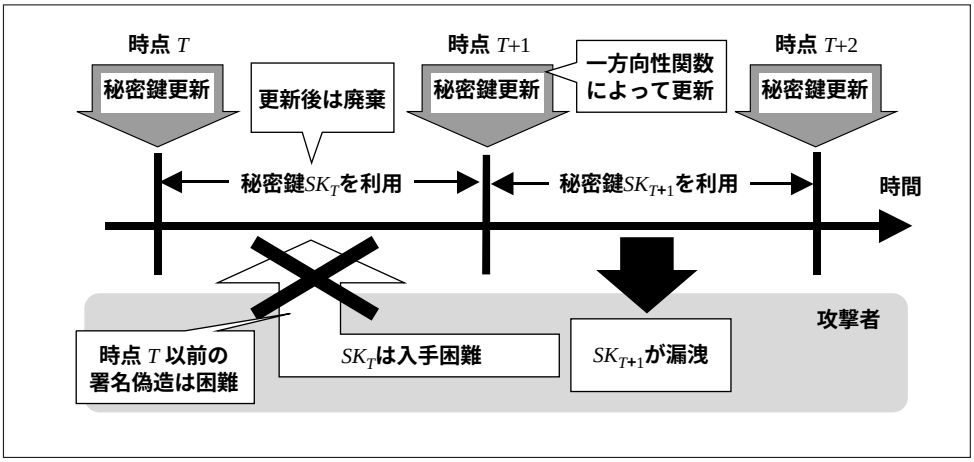
秘密鍵漏洩を前提とした署名偽造に対する対策の研究が近年盛んに行われ、いくつかの技術が提案されている。その中でも代表的なものとして、フォワード・セキュア署名、ヒステリシス署名、実行ハードウエア確認タグ付きデジタル署名、MAC付きデジタル署名が挙げられる。

### （1）フォワード・セキュア署名

フォワード・セキュア署名（forward-secure digital signature）は、秘密鍵を短い期間（例えば1日）で更新し、使用期間が満了した秘密鍵をその都度廃棄することによって、「現在使用している秘密鍵がハードウエア等から漏洩したとしても、その秘密鍵の使用が開始される前に生成されたとされる過去のデジタル署名の偽造を困難にする」という技術である（Anderson［2000］）。これまでにいくつかのフォワード・セキュア署名が提案されている（Bellare and Miner［1999］、Krawczyk［2000］、Abdalla and Reyzin［2000］）が、いずれも上記のアイデアに基づいている。

秘密鍵の更新は一方向性関数によって行われ、ある時点で使用されていた秘密鍵が漏洩してもそれよりも古い秘密鍵を導出困難にしている（図5参照）。さらに、本署名では、異なる複数の秘密鍵によって生成された署名を1つの公開鍵によって検証することができる。

**図5　フォワード・セキュア署名のアイデア**




以下では、これまでに提案されたフォワード・セキュア署名の中でも最も基本的なものとして、ベラーレとマイナーが提案した方式を紹介する（Bellare and Miner［1999］）。

**〈ベラーレ・マイナーのフォワード・セキュア署名〉**

**【準備】** 次の各値を設定する。

・$p, q : p = q = 3 \pmod 4$ を満たす素数
・$T$ ：秘密鍵を設定する期間の総数
・$SK_j$：期間 $j$ において利用される秘密鍵（$j = 1, 2, ..., T$）
・$PK$：公開鍵（すべての期間において利用される）
・$H$ ：出力データのサイズが$m$ ビットの衝突困難な一方向性ハッシュ関数
・$n$：$p$と$q$の積

**【鍵生成・更新アルゴリズム】**

（1）$n$より小さい正整数の集合から$m$個の要素$S_{i,0}$ ($i = 1, 2, ..., m$)をランダムに選択し、$U_i = S_{i,0}^{2^{(T+1)}} \bmod n$ ($i = 1, 2, ..., m$)を計算する。

（2）公開鍵$PK$と、最初の期間に利用される秘密鍵$SK_1$を次のように定め、$p$と$q$を廃棄する。ただし、$S_{i,1} = S_{i,0}^2 \bmod n$ ($i = 1, 2, ..., m$)である。

$PK = (n, T, U_1, U_2, ..., U_m)$〈サイズ：約$k$ $(m+1)$ビット〉

$SK_1 = (n, T, 1, S_{1,1}, S_{2,1}, ..., S_{m,1})$〈サイズ：約$k$ $(m+1)$ビット〉

（3）秘密鍵の更新（例えば$SK_1$から$SK_2$への更新）は次のように行われる。

①$S_{i,2} = S_{i,1}^2 \bmod n$ ($i = 1, 2, ..., m$)を計算し、($S_{1,2}, S_{2,2}, ..., S_{m,2}$)を生成する。

②上記①の結果を用いて$SK_2 = (n, T, 2, S_{1,2}, S_{2,2}, ..., S_{m,2})$と定める。

③$SK_1$を廃棄する。

— このように秘密鍵の更新方法として$n$を法とする平方剰余演算（上記①）が用いられている。大きな合成数$n$を法とする剰余群上で平方根を求めることは困難とみられており、$SK_j$から$SK_{j-1}$を求めることも困難とみられている。

**【署名生成アルゴリズム】**

期間$j$におけるデータ$M$に対する署名は以下の手順で生成される。

（1）$n$より小さく、かつ、$n$と互いに素な正整数の集合から1つの要素$R$をランダムに選択する。

（2）$Y = R^{2^{(T+1-j)}} \bmod n$を計算し、$H(j, Y, M) = c$を計算する。$c$の左から第$t$ビット目の値を$c_t$とする。

（3）$Z = R\prod_{i=1}^{m} S_{i,j}^{c_i} \bmod n$を計算し、$\langle j, (Y, Z)\rangle$を署名とする。

**【署名検証アルゴリズム】**

期間$j$における$M$に対する署名$\langle j, (Y, Z)\rangle$は以下の手順で検証する。

（1）$H(j, Y, M) = c$を計算する。

（2）$Z^{2^{(T+1-j)}} = Y\prod_{i=1}^{m} U_i^{c_i} \bmod n$が成立するかを確認する。成立した場合に限り、$\langle j, (Y, Z)\rangle$が正当な署名であると判定する。

上記方式の安全性に関しては、署名偽造に対する安全性が$n$の素因数分解の困難性に依拠しており、「署名偽造成功確率の上限値が$n$の素因数分解が成功する確率の関数として表される」（すなわち、素因数分解が成功する確率が非常に小さければ、署名の偽造が成功する確率も非常に小さい）ことが証明されている。ただし、公開鍵と秘密鍵のサイズが法$n$のサイズの約$(m+1)$倍（$m$はハッシュ関数$H$の出力データのサイズ）となり、署名生成・検証に必要な計算量がRSA署名やDSA等の一般的な署名方式と比べて増加するというデメリットがある。

フォワード・セキュア署名では、秘密鍵の漏洩形態として、ハードウエアから破壊・非破壊解析によって外部に漏洩する（漏洩形態2に対応）ことが想定されている。また、主なセキュリティ要件を整理すると以下のとおりである。

**〈セキュリティ要件〉**

**①秘密鍵の管理：**署名生成者は、使用期間が満了し、更新された古い秘密鍵を第三者に漏洩しないようにハードウエア内で適切に廃棄する。

**②秘密鍵漏洩の検知：**署名生成者は、漏洩した秘密鍵のうちで最も古いものがどれかを検知する。

上記②に関しては、フォワード・セキュア署名の提案論文には明示的に記述されていない。しかし、②の要件が充足されない場合、ある時点の秘密鍵を入手すると

それ以降の秘密鍵を容易に導出できるため、署名の検証者はどの時点より前の署名が安全かを認識不可能になる（高橋・洲崎・松本［2002］）。

## (2) ヒステリシス署名

ヒステリシス署名（hysteresis signature）は、通常のデジタル署名方式を構成要素の一部として利用するデジタル署名方式の一形態であり、「デジタル署名の生成履歴（署名生成履歴）を署名生成者自身でも偽造困難な形態で保管し、過去に生成されたとされる署名が秘密鍵漏洩等によって偽造された場合であっても、安全に保管されている署名生成履歴との整合性を確認することによって署名偽造を検知する」という技術である（松本ほか［2000］、洲崎ほか［2000］、宮崎ほか［2002］）。

本署名では、ある署名を生成する際にそれ以前の署名等を署名生成記録として署名に埋め込み（チェイニング署名）、すべての署名が相互に関連づけられる（図6、7参照）。この結果、過去の特定時点の署名を偽造するためには、それ以降に生成されたすべての署名を署名生成履歴と整合的に偽造する必要があると考えられるため、署名偽造が一層困難になるという効果が期待されている。

**図6　ヒステリシス署名のアイデア**

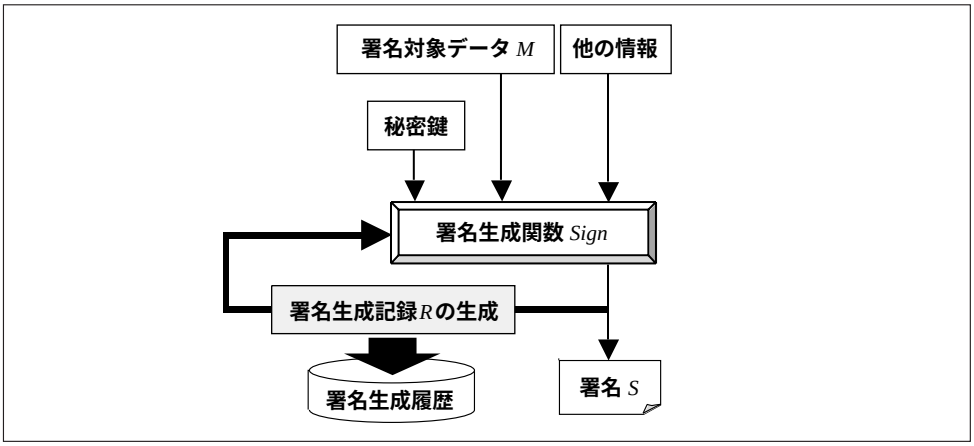
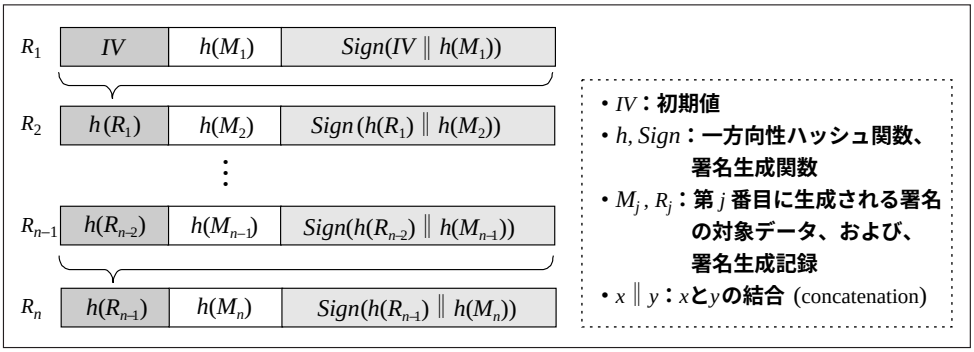
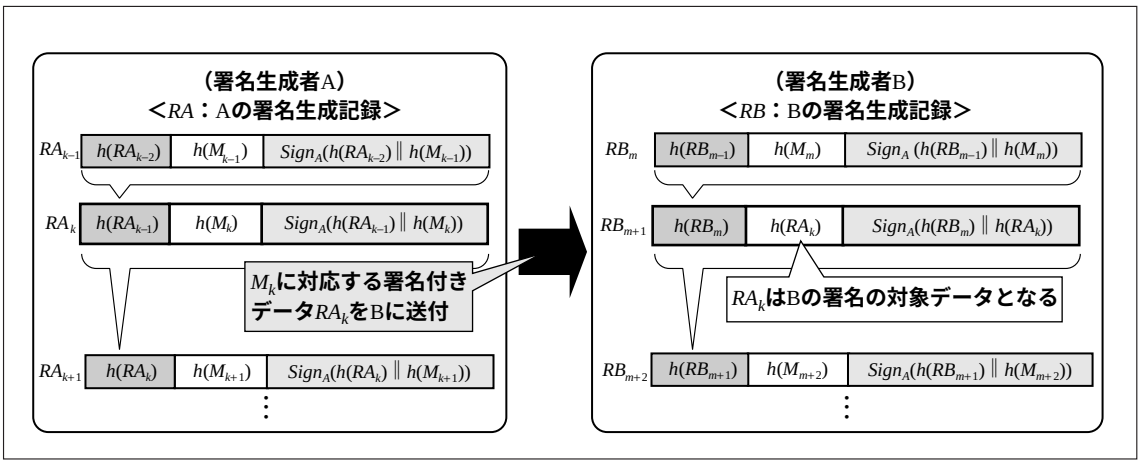



**図7　署名生成記録の構造**



また、松本ほか［2000］では、過去の署名の偽造を一層困難にする手段として履歴交差と呼ばれる方法が提案されている。履歴交差は、ある署名生成者Aが自分で生成した署名付きデータを署名生成者Bに送り、Bは受け取った署名付きデータに対して署名を生成するという方法によって、異なる利用者の署名生成履歴を互いにつづり合わせるというものである（図8参照）。図8で説明すると、例えば、攻撃者がAの署名付きデータ$RA_k$を改ざんしようとする場合、Aが保管する$RA_k$以降の署名生成履歴全体に加えて、Bが保管する$RB_{m+1}$以降の署名生成履歴を改ざんすることが必要になる。

**図8　履歴交差（AからBへ署名を送信）の例**




ヒステリシス署名の生成・検証手順は以下のとおりである（宮崎ほか［2002］）。

**〈ヒステリシス署名の生成・検証アルゴリズム〉**

**【説明に使用する記号とその意味】**

・$Sign, h$：署名生成関数、衝突困難な一方向性ハッシュ関数
・$M_j$：第$j$番目に生成される署名の対象となるデータ
・$S_j, R_j$：第$j$番目に生成される署名付きデータ、署名生成記録
・$H_j$：第$j$番目の署名生成時における署名生成履歴

**【署名・署名生成履歴の生成】**

第$j$番目の署名付きデータ$S_j$を生成する場合を想定する。

<1> 署名対象データ$M_j$と第$j-1$番目の署名生成記録$R_{j-1}$のハッシュ値 $h(M_j)$、$h(R_{j-1})$をそれぞれ生成する。

<2> $S_j = [M_j \| h(R_{j-1}) \| Sign(h(M_j) \| h(R_{j-1}))]$を計算し、生成した$S_j$を$M_j$に対する署名付きデータとする。

<3> ハッシュ値$h(M_j)$を用いて第$j$番目の署名生成記録$R_j$と署名生成履歴$H_j$を次のように生成する。

$$R_j = [h(M_j) \| h(R_{j-1}) \| Sign(h(M_j) \| h(R_{j-1}))]、\ H_j = [H_{j-1} \| R_j]$$

**【署名・署名生成履歴の検証】**

第$j$番目の署名付きデータ$S_j$の検証は次のとおりである。

<1> $M_j$と$Sign(h(M_j) \| h(R_{j-1}))$を用いて、通常の署名検証を実行する。検証処理の内容は利用するデジタル署名方式に依存する。

<2>署名生成履歴の中に$R_j$が存在するか否かを確認する。

<3>署名生成履歴から$R_{j-1}$を取り出し、そのハッシュ値$h(R_{j-1})$が$R_j$に含まれているものと同一かを確認する。

<4>上記<3>の処理を$R_{j+1}$, $R_{j+2}$,...に関して順番に実行する。署名生成記録の一部のみが信頼できる場合には、その署名生成履歴を用いて実行する。

想定される秘密鍵の漏洩形態は、提案論文（松本ほか［2000］、洲崎ほか［2000］）で特に限定されていない。また、主なセキュリティ要件を整理すると次のとおりである。

**〈セキュリティ要件〉**

**①署名生成履歴の管理：**署名生成履歴は、一貫性が確保され、欠損がなく、どれが最新の署名生成履歴かを第三者が確認可能な状態で保管される。

**②署名生成の形態：**正当な署名生成者であっても、署名生成履歴に記録されない形態でヒステリシス署名を適正に生成することは不可能である。

これらのセキュリティ要件を満足させる方法として、洲崎ほか［2000］は、ヒステリシス署名のアルゴリズムを実行するモジュールと署名生成履歴を格納するメモリーを装備したICカードの活用を提案している。ただし、ICカードの故障等によって署名生成履歴が一部失われる場合も想定されるほか、署名生成履歴の管理方法の差異（ICカードに格納する場合や第三者機関に保管を依頼する場合等）によって署名生成履歴の信頼度に差が生じ、署名検証結果の信頼度にも差が生じることが考えられる[15]。こうした点に関して、宮崎ほか［2002］は、検証に利用される署名生成記録の数や管理方法によって署名検証結果の信頼度を算出する方法を提案してい

15 例えば、署名生成履歴がICカードに保管されるとともにその一部（署名生成記録）が定期的に新聞に公表されるというヒステリシス署名のシステムがあり、ICカードから得た署名生成記録よりも新聞に掲載される署名生成記録が一貫性において信頼されているとする。このシステムにおいて署名検証を行う際に、検証に用いられる署名生成記録の系列に新聞から得た署名生成記録が含まれる場合と含まれない場合を比較すると、検証結果に対する信頼度は、新聞から得た署名生成記録が含まれる場合の方が相対的に高いと考えられる。

る。攻撃者が署名生成履歴の一部を偽造することに成功したとしても、偽造署名と正当な署名を比較する場合に、署名検証結果の信頼度を手掛かりとして正当な署名を判定することも考えられる。

### (3) 実行ハードウエア確認タグ付きデジタル署名

実行ハードウエア確認タグ付きデジタル署名（以下、タグ付き署名という）は、通常のデジタル署名方式を構成要素の1つとして利用する署名方式の一実現形態であり、「検証対象の署名が特定のハードウエアにおいて生成されたか否かを確認することで署名偽造を検知する」という技術である（松本・田中［2000］、宇根・松本［2002］）。タグ付き署名では、耐クローン・モジュール（以下、単にモジュールという）[16]が利用可能であることが想定されている。このモジュールはハードウエアに格納され、モジュールの出力等からハードウエアを特定する「タグ」が生成される。署名検証者はタグと署名を用いてハードウエアの確認を行う。こうしたハードウエアを確認するというアイデアは松本・田中［2000］によって提案され、タグ付き署名はこのアイデアをデジタル署名に適用したものである（宇根・松本［2002］）。

タグ付き署名のプロトコルは、管理者A（識別子$ID_A$）、ハードウエア所持者H（署名生成者、識別子$ID_H$）、検証者Vから構成される（図9参照）。Aは、ハードウエアを準備してHに提供し、十分に大きな記憶容量を有するデータベースDB1、DB2を管理するほか、Vの依頼でタグ付き署名の検証を行う。Aは、モジュールの入力集合の一部を実際にモジュールに入力し、その出力を入手する（図10・準備1）。次に、モジュールの入出力テーブル$T_H$を作成して識別子$ID_H$とともにDB1に登録し（図10・準備2）、ハードウエアに組み込んでHに渡す（図10・準備3）。

宇根・松本が示したタグ付き署名のプロトコル例は次のとおりである。

16 耐クローン・モジュールは、「出力値がランダムであり、モジュールの入出力関係を表すテーブルを作成するにはモジュールに実際に入力を与えて出力を観察するという方法しか存在せず、そのためには莫大な時間と記憶領域を必要とする」という性質をもつ（松本・田中［2000］）。こうしたモジュールに関する概念、実現方法、安全性評価方法等に関する研究も近年盛んに進められている（Matsumoto and Matsumoto［2000］）が、現時点では利用可能なものは開発されていない。

**図9　タグ付き署名方式の枠組み**

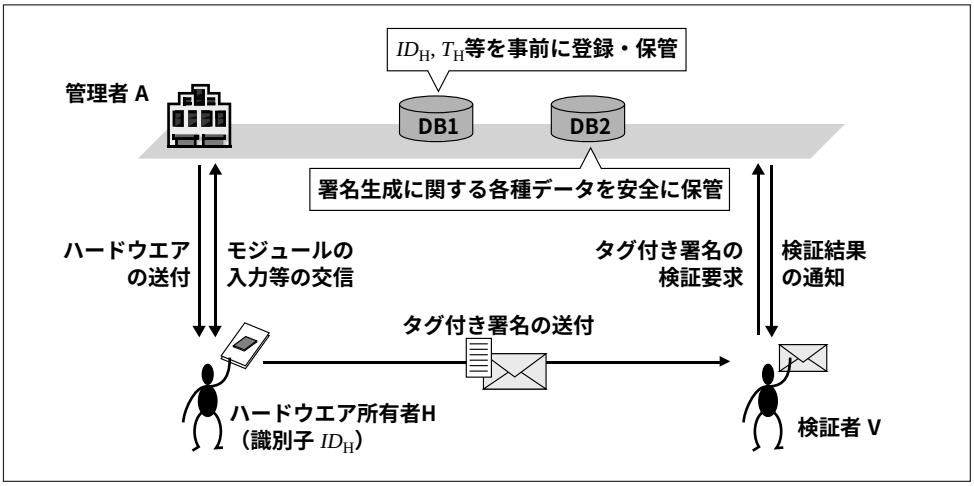
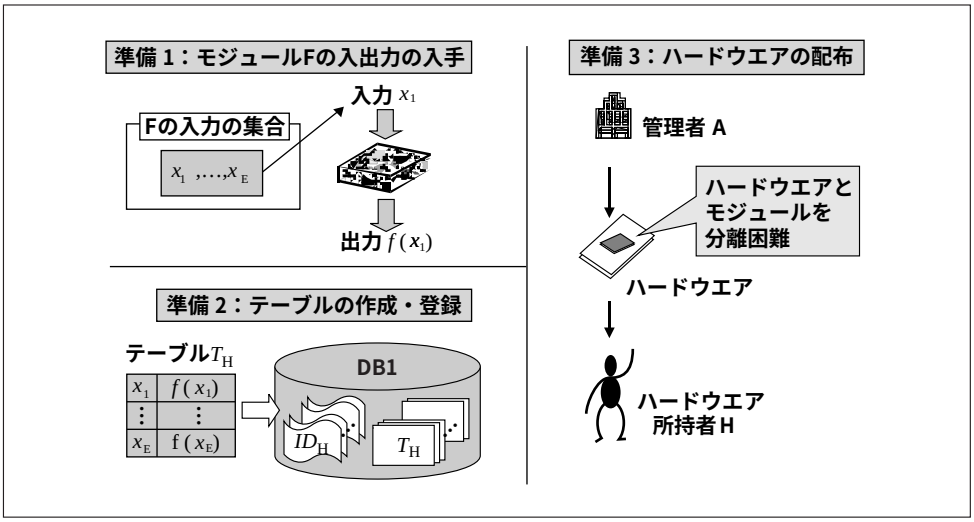



**図10　管理者によるハードウェアの準備**




**〈タグ付き署名の生成・検証プロトコル例〉**

**【記号】**

- $f$,$h$ : Fの入出力関係を表す関数、衝突困難な一方向性ハッシュ関数
- $SD(M)$, $Tag\ (M)$：データ$M$に対する署名付きデータ、および、タグ
- $SKS_A$, $PKE_A$ : $A$の署名生成用秘密鍵と守秘用公開鍵
- $S_A(M)$, $Cert\ (SKS_A)$ : $M$に対する$A$の署名、$SKS_A$に対する公開鍵証明書

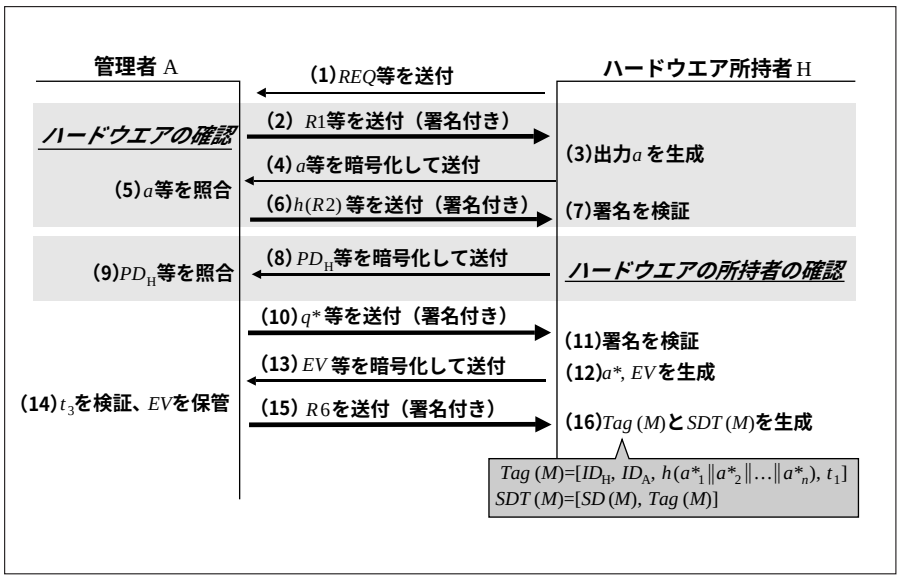
— ハードウエア所持者は自分のハードウエアとパスワードを厳重に管理するほか、ハードウエアの盗難やパスワード漏洩に気づいた場合には直ちに管理者に利用差し止めの請求を行う。

**【準備】図10の準備1〜3に加えて、以下の準備4、5を実行する。**

・準備4：Hはハードウエア内部で署名生成用の鍵ペアを生成する。

・準備5：Hは本人確認用パスワード$PD_H$を生成してAに秘密に通知する。Aは$PD_H$を（$ID_H$, $T_H$）とともにDB1に登録・保管する。

**図11　タグ付き署名方式のプロトコル例：生成手順の概要**




**【タグ付き署名の生成手順〈図11参照〉】**

**（ハードウエアの確認）**

（1）Hはモジュールfへの入力$q$を要求する$REQ$をAに送る。
$A$は、$REQ$の受信日時$t_1$をDB2に記録し、$T_H$から$q = (q_1, q_2,..., q_n)$を選択して$R1 = (ID_H, t_1, q, PKE_A)$、$S_A(R1)$、$Cert\ (SKS_A)$を生成、Hに送る。Hは$S_A(R1)$を検証し、$a_i = f(q_i)$ $(i = 1, 2,..., n)$を生成する。

（2）Hは$R2 = [ID_H, h(R1), a]$ を$PKE_A$ で暗号化してAに送る。ただし、$a = (a_1, a_2,..., a_n)$とする。

（3）Aは$R2$を復号し、$a$が$q$に対応していることを確認する。

（4）Aは、ハードウエアの確認が成功した証として$h(R2)$と$S_A(h(R2))$を送る。Hは$S_A(h(R2))$を検証する。

**（ハードウエア所持者の確認）**

（8）Hは$R3 = [ID_H, h(R2), PD_H]$ を$PKE_A$で暗号化してAに送る。

（9）Aは$R3$を復号し、<1>$PD_H$がDB1に記録されているものと一致するか、<2>$R3$の受信日時$t_2$と$t_1$の時間差が一定値以内か、<3>$R2$のハッシュ値と$R3$に含まれる$h(R2)$が一致するかを確認する。いずれかが成功しない場合、AはHに対して$R3$を再送するように通知する。

**（タグ付き署名およびタグ検証用データ*EV*の生成）**

（10）Aは、$T_H$から選択した$q_i^*$ ($i$ = 1, 2,..., $n$)と$t_2$をDB2に記録し、$R4 = (q^*, t_2, ID_H)$と$S_A(R4)$をHに送る。ただし、$q^* = (q_1^*, q_2^*, ..., q_n^*)$である。

（11）Hは$S_A(R4)$を検証した後、$SD(M)$を生成する。

（12）Hは$a_i^* = f(q_i^*)$を求め、$EV = h(a_1^* \| a_2^* \| ... \| a_n^* \| SD(M) \| t_1)$ を生成する。

（13）Hは$R5 = [ID_H, h(R4), EV]$ を$PKE_A$で暗号化してAに送る。

（14）Aは$R5$を復号し、$R5$の受信日時$t_3$と$t_1$の時間差が一定値以内か、および、$R4$のハッシュ値と$R5$に含まれる$h(R4)$が一致するかを確認する。いずれかが成功しない場合、Aは$R5$の再送を要求する。一定回数以上連続して失敗すると、ハードウエアの使用を差し止める。

（15）Aは$EV$と$t_3$をDB2に記録し、$R6 = [t_3, h(R4)]$と$S_A(R6)$をHに送る。

（16）Hは$Tag(M)$とタグ付き署名データ$SDT(M)$を以下のとおり生成する。

$$Tag(M) = [ID_H, ID_A, h(a_1^* \| a_2^* \| ... \| a_n^*), t_1], \quad SDT(M) = [SD(M), Tag(M)]$$

Aは$DB2に(ID_H, t_1, q^*, EV)$を保管する。

**【タグの検証】**

（1）Aは、$Tag(M)$ に含まれる$(ID_H, t_1)$に対応する$q^*$をDB2において探索する。

（2）Aは$a_i^{*\prime} = f(q_i^*)$、$h(a_1^{*\prime} \| a_2^{*\prime} \| ... \| a_n^{*\prime})$を計算し、$Tag(M)$内の$h(a_1^* \| a_2^* \| ... \| a_n^*)$と照合する。

（3）Aは、$SDT(M)$内の$SD(M)$を用いて$EV' = h(a_1^{*\prime} \| a_2^{*\prime} \| ... \| a_n^{*\prime} \| SD(M) \| t_1)$を生成し、DB2に記録されている$EV$と照合する。

上記のプロトコル例では、Aが、ハードウエアの正当性をモジュールの入出力の確認によって検証するほか、ハードウエアを操作している者が正当なハードウエア所持者であることをパスワード照合によって検証する。これらの処理が完了した後、タグ付き署名データが生成される。このとき、モジュールの出力と署名を結び付けるデータとして$EV$が生成され（生成処理の（12）に対応）、管理者に送付されたDB2に登録・保管される。後日タグ付き署名を検証する際に、DB2に保管されている$EV$の照合によって署名偽造を検知可能となる。

想定される秘密鍵の漏洩に関しては特に限定されていない（松本・田中［2000］、宇根・松本[2002]）。タグ付き署名の主なセキュリティ要件を整理すると次のとおりである。

**〈セキュリティ要件〉**

**①モジュールとハードウエア：**管理者は、モジュールの機能を損なうことなくモジュールをハードウエアから分離困難な形態でハードウエアを準備する。

**②管理者：**管理者は、攻撃者との結託等の不正な行為を行わないと同時に、2つのデータベースDB1、DB2を第三者に情報が漏洩しないように、また、第三者によって不正に改ざんされないように管理する。

### （4）MAC付きデジタル署名

MAC付きデジタル署名（以下、MAC付き署名という）は、通常のデジタル署名方式を構成要素の1つとするデジタル署名方式の一実現形態であり、「あるデータに対する署名を生成する際にMAC（message authentication code）をあわせて生成し、署名が偽造されたと疑われる場合にはMACの検証を行って署名生成を実行したハードウエアを確認し、偽造の有無を判定する」という技術である（小森・松浦・須藤［2001a, b, 2002］）。

MAC付き署名において使用される署名生成用ハードウエアは、「ハードウエア製造時にはデータの書込みが可能であるが、その後は書込みも不正な読出しも不可能」なメモリー領域を有し、その領域にMAC生成用の秘密鍵を格納する。一方、署名生成用の秘密鍵は、外部で生成された後に別のメモリー領域に格納され、定期的に更新されることが前提となっている。このように、MAC付き署名では、例えばハードウエア内部で署名生成用秘密鍵等を生成し、外部に一切漏れないようにするといった鍵管理方法を採用しているアプリケーションを対象としていない。

MAC生成用の秘密鍵については、署名生成用の秘密鍵とは異なり、ハードウエア製造後は書込みおよび不正な読出しが不可能なメモリー領域に格納され、いったん格納されると更新されることはない。MAC生成用の秘密鍵のサイズに関しては、そのハードウエアが利用される期間中において十分な安全性を確保可能と考えられる値に設定される。

MAC付き署名を構成するエンティティは、署名生成者、署名検証者、調停者の三者である。調停者は、署名生成者や署名検証者から信頼される第三者機関であり、署名偽造の疑いが発生した場合等に署名検証者からの依頼によってその署名に対応するMACの検証を実行する。

署名生成および検証の手続は以下のとおりである。

**〈MACを用いた署名偽造の検知方法〉**

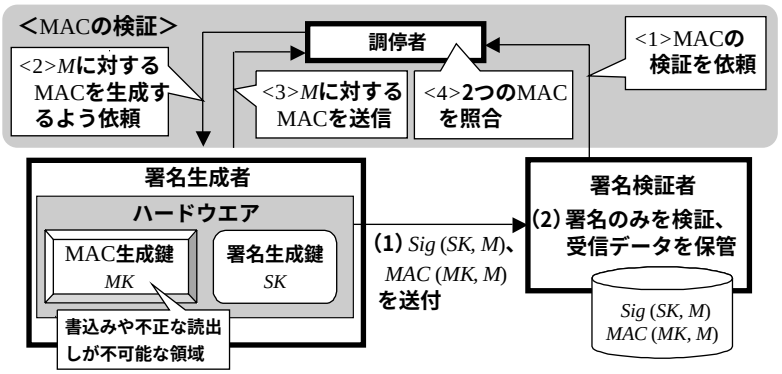
**【記号】**

・*SK, MK* ：署名生成用の秘密鍵、MAC生成用の秘密鍵
・*MAC*(*MK, M*) ：データ*M*に対する（鍵*MK*によって生成された）MAC
・*Sig*(*SK, M*)：*M*に対する署名と*M*を結合したデータ

**【署名・MAC生成手続〈図12参照〉】**

（1）署名生成者はハードウエアにPINを入力し、本人確認を行う。
（2）署名生成者は、*Sig*(*SK, M*)と*MAC*(*MK, M*)を生成して署名検証者に送る。
（3）署名検証者は、*Sig*(*SK, M*)の検証を行ったうえで、*Sig*(*SK, M*)と*MAC*(*MK, M*)をデータベースに保管する。

**図12　MAC付き署名の生成と検証**




**【署名偽造の疑いが生じた際のMAC検証手続〈図12参照〉】**

(1) 署名検証者は、署名が偽造された疑いがある署名対象データ*M*と*MAC* (*MK, M*)を調停者に提出し、MACの検証を依頼する。

(2) 調停者は、*M*を署名生成者に送付し、ハードウエアを用いて*M*に対するMACを生成するように要請する。

(3) 署名生成者は*M*に対するMACを生成し、調停者に送付する。

(4) 調停者は、署名検証者から入手したMACと署名生成者から入手したMACを照合する。一致する場合には*M*に対する署名が正当なものであると判定し、一致しない場合には、署名が偽造されたものであると判定する。

想定される秘密鍵の漏洩に関しては、提案論文（小森・松浦・須藤［2001a］）において特に限定されていない。主なセキュリティ要件を整理すると以下のとおりである。

**〈セキュリティ要件〉**

① **ハードウエア：**署名生成用のハードウエアは、MAC生成用の秘密鍵を格納するために、製造時以外は書込みが不可能であり、不正な読出しが不可能なメモリー領域を有する。

② **MAC生成用アルゴリズム：**署名生成者は、MAC生成用アルゴリズムやその実装方法からMAC生成用の秘密鍵が漏洩しないように、MAC生成用アルゴリズムやその実装方法を適切に選択する。

③ **署名検証者のデータベース：**署名検証者は、署名生成者から受け取った署名やMACをトラブル時に備えて安全に保管する。

上記③のセキュリティ要件を充足する方法の1つとして、署名検証者がデータベースに保管するデータのログを生成し、仮にデータベースのデータが一部改ざんされたとしても事後的に検知可能にする方法が提案されている（小森・松浦・須藤［2002］）。

### （5）技術の分類・比較

以上説明した4つの技術について、それらの効果、想定環境、セキュリティ要件等を整理すると次頁の表1のとおりである。以下では表1を基に技術の分類・比較を行う。

#### イ．効果と実現方法による技術の分類

秘密鍵が漏洩した場合における各技術の効果をみると、まず、漏洩した秘密鍵によって偽造が可能となるデジタル署名の範囲を限定する技術と、署名偽造を事後的に検知可能にする技術に分けられる（図13参照）。

**図13　効果・実現方法による技術の分類**

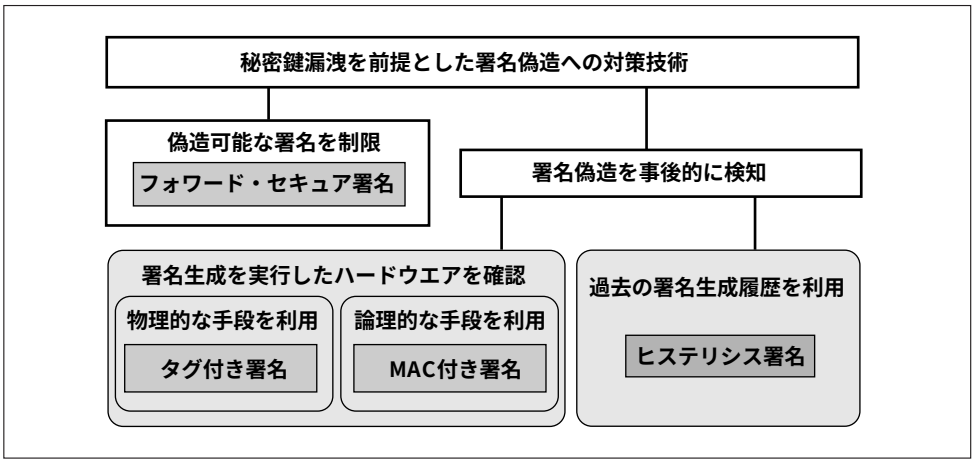


**表1　各技術の効果・実現方法・想定環境・セキュリティ要件**

| | 効果 | 実現方法 | 想定環境 | | セキュリティ要件 | | | | 署名生成・検証時のデータ通信 |
|---|---|---|---|---|---|---|---|---|---|
| | | | 秘密鍵の漏洩形態 | その他 | 鍵管理 | 信頼できる第三者 | 署名生成用ハードウエア | その他 | |
| フォワード・セキュア署名 | **偽造可能な署名を限定する。**漏洩した秘密鍵が使われる前の署名を偽造困難にする。 | ・秘密鍵を短い期間で更新し、古い秘密鍵を更新の都度廃棄する。<br>・一方向性関数を用いて、更新後の秘密鍵から更新前の秘密鍵を導出困難にする。 | 漏洩形態2 | —— | 署名生成者は漏洩した秘密鍵のうちで最も古いものを検知する。 | —— | 古い秘密鍵を内部で安全に廃棄する機能を有する。 | —— | —— |
| ヒステリシス署名 | **署名の偽造を事後的に検知する。**署名生成履歴を保管し、検証対象の署名との整合性を確認する。 | ・通常の署名方式を構成要素とする。<br>・過去に生成した署名を署名生成履歴としてすべて保管しておき、検証対象の署名と署名生成履歴との整合性を確認する。<br>・ある署名を生成する際に、それ以前に生成された署名等を埋め込み、署名の連鎖を形成する。 | 漏洩形態1, 2, 3 | | ——<br>（秘密鍵漏洩を直ちに検知可能か否かは各技術の効果に影響を与えない） | 署名生成履歴を保管する第三者を用いる場合では、その第三者が安全に管理する。 | 署名生成履歴にすべての署名生成記録を正しくつづり込み、安全に保管する機能を有する。 | | 履歴交差を利用する場合には署名生成者間で通信が行われる。 |
| タグ付き署名 | **署名の偽造を事後的に検知する。**署名が特定のハードウエアにおいて生成されたことを確認する。 | ・通常の署名方式を構成要素とする。<br>・ハードウエアに耐クローン・モジュールを組み込む。<br>・ハードウエアを特定するデータ（タグ）を署名とともに保管する。<br>・署名が偽造された可能性がある場合、信頼できる第三者機関（管理者）にタグの検証を依頼する。 | | 耐クローン・モジュールを利用できる。 | | 管理者が2種類のデータベースを安全に管理する。 | 耐クローン・モジュールが署名生成用ハードウエアに分離困難な形態で組み込まれる。 | | 署名生成時に署名生成者と管理者間で、署名検証時には署名検証者と管理者間で通信が行われる。 |
| MAC付き署名 | **署名の偽造を事後的に検知する。**署名対象データに対するMACを検証して署名が生成されたハードウエアを確認する。 | ・通常の署名方式を構成要素とする。<br>・ハードウエアにはMAC生成用と署名生成用の秘密鍵が格納される。<br>・署名対象データのMACを署名とともに保管する。<br>・署名偽造判定では、検証者は署名とMACを調停者に提出し、調停者は署名生成者が再度生成したMACと検証対象のMACを比較する。 | | —— | | 調停者が署名生成者の協力を得て署名偽造の有無を判定する。 | MAC生成用秘密鍵を格納する記憶領域（書込みや不正な読出しが不可能）を有する。 | MAC生成アルゴリズムや実装方法は安全である。 | 署名偽造を判定する際に調停者と検証者、および、調停者と署名生成者との間で通信が行われる。 |

偽造可能な署名を限定する技術にはフォワード・セキュア署名が該当する。フォワード・セキュア署名では、偽造可能な署名を限定する方法として、秘密鍵を短期間で更新し、更新の都度古い秘密鍵を廃棄するという方法が採用されている。これは、「古い秘密鍵を廃棄することによって漏洩を未然に防ぐ」というアイデアである。

一方、署名偽造を事後的に検知可能にする技術には、ヒステリシス署名、タグ付き署名、MAC付き署名が該当する。これらの技術はいずれも通常のデジタル署名方式に追加的な工夫を施したものであるが、その工夫の内容が異なっている。ヒステリシス署名では、過去に正しく生成された署名の記録との整合性を確認するという工夫が施されている。これに対して、タグ付き署名とMAC付き署名では、前者は物理的な手段、後者は論理的な手段によって署名生成が実行されたハードウエアを確認するという工夫が施されている。MAC付き署名に関しては、MAC生成用の秘密鍵がハードウエアと一体化されて利用されることから、調停者によるMACの検証によってハードウエアと署名対象データとの関連性が確認されることとなる[17]。

**ロ．想定環境**

まず、想定される秘密鍵の漏洩形態に関しては、ヒステリシス署名、タグ付き署名、MAC付き署名では特に限定されているわけではないが、フォワード・セキュア署名ではハードウエアに対する破壊型・非破壊型解析による秘密鍵の漏洩（漏洩形態2）に限定されている。このことから、ヒステリシス署名、タグ付き署名、MAC付き署名で想定される秘密鍵の漏洩形態は、フォワード・セキュア署名で想定されている漏洩形態よりも広いといえる。

その他、タグ付き署名では、耐クローン・モジュールが利用可能であることが想定されている。耐クローン・モジュールは現時点で利用可能なものは存在しておらず、既存技術の組合せによって実現可能とみられる他の3つの技術に比べて実現可能性という点で劣っている。

**ハ．セキュリティ要件**

主なセキュリティ要件として、秘密鍵の管理、信頼できる第三者、署名生成用のハードウエアに関する要件等を取り上げ、比較する。

まず、秘密鍵の管理に関するセキュリティ要件については、いずれの技術においても、提案論文には明示されていないが秘密鍵の適切な生成・配布・廃棄等の一般的な要件を充足することが前提となっていると考えられる。ただし、フォワード・セキュア署名においては、「秘密鍵が漏洩した場合、署名生成者は漏洩した秘密鍵のうちで最も古いものを検知する」という要件が追加されている。

17 MAC付き署名では、MACによって「ハードウエアと署名対象データ」の関連性が確認可能となるほか、ハードウエアの本人確認手段（別途準備される）によって「署名生成者とハードウエア」の関連性が確認可能となる。この結果、署名生成者と署名対象データの関連を第三者が確認することができる。

信頼できる第三者に関するセキュリティ要件については、フォワード・セキュア署名では、署名生成・検証をそれぞれ署名生成者と署名検証者が単独で実行する形態となっており、信頼できる第三者の利用が特に求められているわけではない。ヒステリシス署名では、基本的には信頼できる第三者の利用が想定されていないが、ハードウエアの外部で署名生成履歴を安全に保管するアプリケーションの場合、信頼できる第三者に関するセキュリティ要件が設定されることも考えられる。タグ付き署名とMAC付き署名は、それぞれ管理者、調停者という信頼できる第三者を利用することが要件となっている。

署名生成用のハードウエアに関するセキュリティ要件については、フォワード・セキュア署名では、秘密鍵を安全に廃棄するなどの一般的な署名生成用ハードウエアに求められる要件のほかには特別な要件は設定されていない。一方、ヒステリシス署名では、署名生成が実行されるたびにその署名生成記録を署名生成履歴につづり込み、安全に保管するという要件が置かれている。また、タグ付き署名では、耐クローン・モジュールの機能を損なうことなくそれをハードウエアから分離することが困難であるという要件が置かれている。MAC付き署名では、MAC生成用の秘密鍵を格納するために、書込みや不正な読出しが不可能な記憶領域を有するという要件が置かれている。

### ニ．分類・比較のまとめ

フォワード・セキュア署名は、信頼できる第三者やハードウエアに頼らないという特徴を有し、実現するには大きなコストを要すると考えられる「信頼できる第三者」を準備する必要がないという利点を有しているといえる。しかし、秘密鍵の管理に関して追加的なセキュリティ要件が設定されており、秘密鍵の管理による署名生成者の負担が相対的に大きいと考えられる。さらに、フォワード・セキュア署名は、現在広く採用されているRSA署名やDSA等のデジタル署名方式とは異なるタイプの独自のアルゴリズムを採用しており、デジタル署名方式の国際標準等に規定されているわけではなく汎用性が低いと考えられるほか、CRYPTREC等の第三者による安全性評価が行われていないという問題がある。

ヒステリシス署名、タグ付き署名、MAC付き署名は、秘密鍵の管理に関しては追加的なセキュリティ要件が設定されておらず、秘密鍵管理における署名生成者の負担が相対的に小さいというメリットがある。また、これらの技術は、RSA署名やDSAなどの署名方式を構成要素の1つとして利用可能であり、汎用性が高いと考えられる。しかし、フォワード・セキュア署名と同様に、現時点では安全性の評価が十分に行われていないという問題があるほか、信頼できる第三者やハードウエアを利用することにともなって追加的なコストが発生するというデメリットがある。

## ５．おわりに

デジタル署名は安全な電子商取引等を実現するうえで重要な技術の1つとなっている。しかし、仮に署名生成用の秘密鍵が第三者（攻撃者）に漏洩した場合、署名の偽造が可能となり、現在のPKIの枠組みだけでは十分な対応が困難である。安全性の観点でより安心できる電子商取引を実現するためには、秘密鍵の漏洩を前提とした署名偽造の影響について考慮しておくことが求められる。

本稿では、そうした署名偽造への対策として提案されている代表的な技術について紹介した。これらの技術では、それぞれ異なったアプローチを採用し、秘密鍵漏洩にともなう署名偽造に対して一定の効果を有している。しかし、いずれも現時点で完成した技術というわけではなく、さまざまな課題が残されており、実際に実装可能なものにするためには更なる研究・開発が必要とされている。今後は、これらの技術の研究動向についてもフォローし、署名生成者や署名検証者がより安心して利用することができるデジタル署名方式について検討していくことが必要であろう。

## 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、日本銀行金融研究所、1999年9月、51～84頁

―――、「金融分野におけるPKI：技術的課題と研究・標準化動向」、『金融研究』第21巻別冊第1号、日本銀行金融研究所、2002年6月、227～283頁

―――・岡本龍明、「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、日本銀行金融研究所、2000年4月、55～104頁

―――・松浦幹太・田倉昭、「デジタルタイムスタンプ技術の現状と課題」、『金融研究』第19巻別冊第1号、日本銀行金融研究所、2000年4月、105～153頁

―――・松本 勉、「実行ハードウエア確認タグ付きディジタル署名方式」、『情報処理学会研究報告』2002-CSEC-18、情報処理学会、2002年7月、245～252頁

桶屋勝幸・櫻井幸一、「サイド・チャネル攻撃およびフォールト攻撃を凌ぐハイブリッドハードウエア攻撃」、『電子情報通信学会技術研究報告』Vol. 102, No. 210、電子情報通信学会、2002年3月、133～138頁

小森 旭・松浦幹太・須藤 修、「PKIに基づくC/S型アプリケーションの安全性分析と証拠性評価」、『コンピュータセキュリティシンポジウム2001論文集』、情報処理学会、2001年10月a、319～324頁

―――・―――・―――、「契約時に添える付加的なMACに関する総合的分析」、『情報処理学会研究報告』、2001-CSEC-15、情報処理学会、2001年12月b、31～36頁

―――・―――・―――、「電子商取引における紛争解決のための電子証拠物に関する分析」、『2002年暗号と情報セキュリティシンポジウム予稿集』、電子情報通信学会、2002年2月、627～632頁

齊藤真弓、「RSA署名方式の安全性を巡る研究動向について」、『金融研究』第21巻別冊第1号、日本銀行金融研究所、2002年6月、285～323頁

情報処理振興事業協会、『平成11年度 スマートカードの安全性に関する調査 調査報告書』、2000年2月

洲崎誠一・松本 勉、「電子署名の偽造に関する一考察」、『コンピュータセキュリティシンポジウム2001論文集』、情報処理学会、2001年10月、211～216頁

―――・宮崎邦彦・宝木和夫・松本勉、「暗号ブレイク対応電子署名アリバイ実現機構（その2）―詳細方式―」、『情報処理学会研究報告』2000-CSEC-8、情報処理学会、2000年3月、18～23頁

全国銀行協会、「ICキャッシュカードに関する認定制度の創設と全銀協認証局の設置について」、『金融』2001年11月号、全国銀行協会、2001年11月、16～25頁

総務省、『政府認証基盤（GPKI）府省認証局CP／CPSガイドライン』、2001年4月（http://www.soumu.go.jp/gyoukan/kanri/010514_5.pdf）

高橋知史・洲崎誠一・松本 勉、「Forward-Secure Digital Signatureは役に立つか」、『2002年暗号と情報セキュリティシンポジウム予稿集』、電子情報通信学会、2002年1月、837～842頁

田中圭介・岡本龍明、「量子公開鍵暗号」、『電子情報通信学会誌』2002年8月号、電子情報通信学会、2002年8月、613～617頁
谷口文一、「金融業界におけるPKI・電子認証について ―」、『金融研究』第19巻別冊第1号、日本銀行金融研究所、2000年4月、15～54頁
電子商取引実証推進協議会認証・公証ワーキンググループ、『認証局の責任に関する提言』、H11－認証・公証－3、2000年3月
中山靖司・小松尚久、「バイオメトリックスによる個人認証技術の現状と課題 －金融サービスへの適用の可能性－」、『金融研究』第19巻別冊第1号、日本銀行金融研究所、2000年4月、155～192頁
松本 勉・岩村 充・佐々木良一・松木 武、「暗号ブレイク対応電子署名アリバイ実現機構（その1）―コンセプトと概要―」、『情報処理学会研究報告』、2000-CSEC-8、情報処理学会、2000年3月、13～17頁
――――・田中直樹、「計算の実行ハードウエアを確認する方法」、『コンピュータセキュリティシンポジウム2000論文集』、情報処理学会、2000年10月、199～204頁
宮崎邦彦・吉浦 裕・岩村 充・松本 勉・佐々木良一、「連鎖構造を用いた電子署名技術における信頼性評価手法の提案」、『電子情報通信学会技術研究報告』Vol. 102, No. 212、電子情報通信学会、2002年7月、109～115頁
Abdalla, Michel, and Leonid Reyzin, “A New Forward-Secure Digital Signature Scheme,” *Proceedings of ASIACRYPT 2000*, LNCS 1976, Springer-Verlag, 2000, pp. 116-129.
Anderson, Ross, “Two Remarks on Public-Key Cryptology,” Manuscript, 2000.
――――, *Security Engineering – A Guide to Building Dependable Distributed Systems*, Wiley Computer Publishing, John Wiley & Sons, Inc., 2001.
Bellare, Mihir, and Sara K. Miner, “A Forward-Secure Digital Signature Scheme,” *Proceedings of CRYPTO ’99*, LNCS 1666, Springer-Verlag, August, 1999, pp. 431-448.
Bond, Mike, “Attacks on Cryptoprocessor Transaction Sets,” *Proceedings of CHES 2001*, LNCS 2162, Springer-Verlag, 2001, pp. 220-234.
Boneh, Dan, Richard A. DeMillo, and Richard J. Lipton, “On the Importance of Checking Cryptographic Protocols for Faults,” *Proceedings of EUROCRYPT ’97*, LNCS 1233, Springer-Verlag, 1997, pp. 37-51.
Chokhani, Santosh, Warwick Ford, Randy Sabett, Charles Merrill, and Stephen Wu, *RFC 2527 Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework*, July, 2001(http://www.ietf.org/rfc /rfc2527.txt).
Clayton, Richard, and Mike Bond, “Experience Using a Low-Cost FPGA Design to Crack DES keys,” presented at *CHES* 2002, August 1, 2002.
IBM Corporation, *IBM PCI Cryptographic Coprocessor, CCA Basic Service Reference and Guide for IBM 4758 Models 002 and 023 with Release 2.40*, September, 2001.
――――, *IBM PCI Cryptographic Coprocessor, CCA Basic Service Reference and Guide Release 2.41, Revised for IBM 4758 Models 002 and 023*, May, 2002a.

――――, *IBM Cryptographic Products, IBM PCI Cryptographic Coprocessor General Information Manual*, May, 2002b.

Kocher, Paul C., "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS and Other Systems," *Proceedings of CRYPTO '96*, LNCS 1109, Springer-Verlag, 1996, pp. 104-113.

――――, Joshua Jaffe, and Benjamin Jun, "Differential Power Analysis," *Proceedings of CRYPTO '99*, LNCS 1666, Springer-Verlag, 1999, pp. 388-397.

Krawczyk, Hugo, "Simple Forward-Secure Signatures From Any Signature Scheme," *7th ACM Conference on Computer and Communications Security*, 2000.

Matsumoto, Hiroyuki, and Tsutomu Matsumoto, "Artifact-metric Systems," *Technical Report of IEICE*, 100 (323), Institute for Electronics, Information and Communication Engineers, 2000, pp. 7-14.

Menezes, Alfred J., Paul C. van Oorschot, and Scott A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.

National Institute of Standards and Technology, *FIPS PUB 140-1: Security Requirements for Cryptographic Modules*, January 11, 1994 (http://csrc.nist.gov /publications/fips/fips1401.pdf).

――――, *FIPS PUB 186-2 : Digital Signature Standard* (*DSS*), January, 2000 (http://csrc.nist.gov/publications/fips/fips186-2/fips186-2-change1.pdf).

――――, *FIPS PUB 140-2: Security Requirements For Cryptographic Modules*, May 25, 2001 (http://csrc.nist.gov/publications/fips/fips1402.pdf).