IMES DISCUSSION PAPER SERIES

ICカードに利用される暗号アルゴリズムの安全性について

― EMV 仕様の実装上の問題点を中心に ―

鈴木雅貴（すずきまさたか）・神田雅透（かんだまさゆき）

Discussion Paper No. 2007-J-8

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 2007-J-8
2007 年 3 月

ICカードに利用される暗号アルゴリズムの安全性について
― EMV仕様の実装上の問題点を中心に ―

鈴木雅貴（すずきまさたか）*・神田雅透（かんだまさゆき）**

要　　旨

わが国の金融業界では、IC カード型のクレジット・カードやキャッシュ・カードが普及し始めている。IC カードは、その演算・判断・記憶の機能を活用することにより、従来の磁気ストライプ型カードに比べ、セキュリティ機能の向上が図られており、カードの偽造・不正使用に対する耐性が高まっている。

現在、こうした IC カードを用いたカード取引のための IC カードと端末に関するデファクト標準として、EMV 仕様が広く普及している。EMV 仕様では、IC カードのセキュリティ機能を高めるために様々な暗号技術が利用されている。そこで推奨されている暗号アルゴリズムは、定期的な見直しが行われているが、現在の暗号技術の水準からみると十分に安全とは言い切れないものも含まれている。EMV 仕様に基づいて IC カードを利用したシステムを構築する場合には、こうした暗号技術の安全性の観点を考慮しつつ、暗号アルゴリズムの適切な選択および適切な運用方法の採用による実効的な安全性向上を図っていくことが重要である。

本稿では、EMV 仕様を対象に、主として暗号アルゴリズムに着目しながら、情報セキュリティ面での安全性の考察を行う。

キーワード：IC カード、EMV 仕様、暗号アルゴリズム、RSA、2-key TDES、擬似 TDES-MAC
JEL classification: L86、L96、Z00

*　日本銀行金融研究所（E-mail：masataka.suzuki@boj.or.jp）
**　NTT 情報流通プラットフォーム研究所（E-mail：kanda.masayuki@lab.ntt.co.jp）

本稿は、2007 年 3 月 6 日に日本銀行で開催された「第 9 回情報セキュリティ・シンポジウム」への提出論文に加筆・修正を加えたものである。なお、本稿に示されている意見は、筆者たち個人に属し、日本銀行あるいは NTT 情報流通プラットフォーム研究所の公式見解を示すものではない。また、ありうべき誤りはすべて筆者たち個人に属する。

目　次

１．はじめに

わが国の金融業界では、IC カード型のクレジット・カードやキャッシュ・カードが普及し始めている。IC カードは、その演算・判断・記憶の機能を活用することにより、従来の磁気ストライプ型カードに比べ、セキュリティ機能の向上が図られている。例えば、IC カードでは、カード内部のファイルへのアクセス制御を行うことによって内部データを不正なアクセスから保護するなど、磁気ストライプ型カードに比べて格段に高いセキュリティが確保されている。

こうした IC カード・端末の技術的な要件や通信プロトコルを定めた標準としては、国際的なデファクト標準である EMV 仕様が広く用いられている。わが国では、銀行系カード会社で構成される日本クレジットカード協会（JCCA）が策定した「IC カード対応端末機能仕様書（JCCA 仕様）」や、全国銀行協会が策定した「全銀協 IC キャッシュカード標準仕様（全銀協仕様）」が EMV 仕様に基づいて策定されている。

EMV 仕様は、金融分野で用いられる IC カードが備えるべき共通事項を定めた標準として広く普及しているが、詳細な技術要件については、EMV 仕様の範囲内で、各利用主体が各々のニーズに基づいて独自に設定し、システムに実装することが可能である。このため、技術要件の内容やシステムへの実装の仕方によっては、セキュリティ機能の水準に差異が生じる可能性も否定できない。とりわけ暗号技術については、EMV 仕様で推奨されている暗号アルゴリズムの中に様々な特性や強度を持つものがあり、システムに実装する場合に、いかなる暗号アルゴリズムを選択するかによって、当該システムのセキュリティ上の安全性に違いが現れる可能性がある。

そこで、本稿では、EMV 仕様（EMVCo [2004]）について、セキュリティ機能を中心に紹介するとともに、これを実装する場合の安全性について、主として暗号アルゴリズムに着目しながら考察を行うこととする。

以下、2 節では、EMV 仕様の概要とそのセキュリティ機能を簡単に紹介する。そのうえで、3 節では、EMV 仕様で利用される暗号アルゴリズムの安全性について述べ、4 節で本稿を締め括る。

２．EMV 仕様について

（１）概要

EMV 仕様とは、金融分野における IC カードを用いたカード取引のための IC カードと端末に関する仕様を定めた国際的なデファクト標準である。EMV 仕様の策定の経緯をみると、国際的なクレジット・カード・ブランドである Europay International、MasterCard International、Visa International の 3 社が、IC カードと端末の互換性を確保するために必要な機能要件とカードの不正使用を防ぐためのセキュリティ要件に関する検討を 1993 年に開始し、その検討結果を踏まえて 1996 年に EMVv3.0 として公開され、その後、継続的に改訂が行われてきている[1]。

この EMV 仕様は、一般的な外部端子付 IC カードの物理的・機能的条件等を規定した国際規格 ISO/IEC 7816 シリーズ（Identification Cards ― Integrated circuit cards with contacts）に準拠しつつ、金融分野向けに必要な IC カードと端末の仕様を規定したものである。具体的には、①IC カードと端末の物理的・電気的特性およびハードウェア・インターフェース、②IC カード・端末間でやり取りするデータ要素およびコマンド、③IC カード・端末間の処理フロー等を定義している。

JCCA 仕様や全銀協仕様は、この EMV 仕様に基づいて策定されている。また、国際的なクレジット・カード・ブランドは、EMV 仕様に準拠した形でより詳細な技術要件を定めており、これをもとに、傘下のカード発行会社は各々の提供するサービスやリスク管理方針に応じて、システムへの実装を行っている（図 1 参照）。このように、EMV 仕様は、各利用主体がより詳細な技術要件を定め、システムに実装することを可能にする拡張性を備えている。

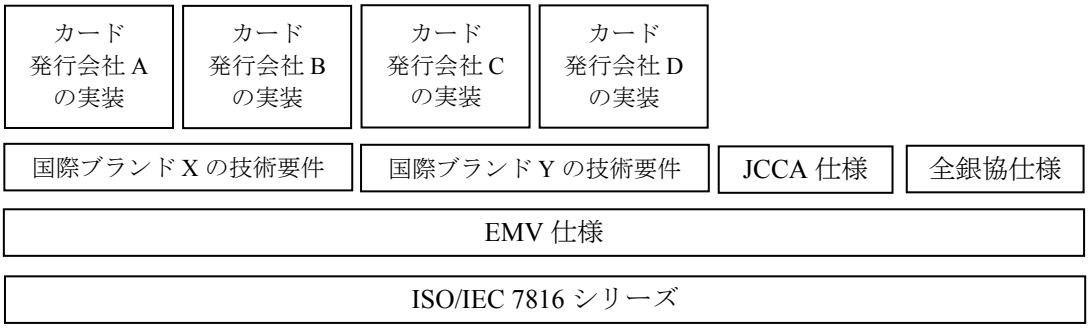

図 1．EMV 仕様に関連する国際標準や技術要件との関係

[1] EMV 仕様の管理・維持は、3 社の出資で 1998 年に設立された EMVCo, LLC が行っている。2002 年に MasterCard International が Europay International を吸収合併した後、2004 年には JCB が新たに EMVCo, LLC の出資者に加わった。なお、EMV 仕様の最新版は、2004 年に改訂された v.4.1 である。

（２）セキュリティ機能

IC カードの特徴として、偽造や不正使用を防止するため、従来の磁気ストライプ・カードに比べて高度なセキュリティ機能を備えていることが挙げられる。EMV 仕様では、例えば、暗号技術を応用したセキュリティ機能として、データ認証、カード所持者認証、AC（Application Cryptogram）生成を用意している。

これらのセキュリティ機能を実際の IC カード用のシナリオに沿って説明すれば以下のとおりである。

まず利用者が端末にカードを提示すると、端末はカードやそのデータが真正であることを確認するために、「データ認証」を行う。次に、発行者のホストシステムまたは端末が、カードを提示した利用者がカードの真の所持者であることを「カード所持者認証」によって確認する。これらの確認を経たうえで、取引データ（取引金額、取引日時）その他の条件に基づいて、当該取引を承認するか否かを決定する。その際、承認の判断に利用される取引データが改ざんされていないことや、当該取引のためにカードが利用されていることを保証するために、取引毎に「AC 生成」を行うという仕組みになっている。

これらのセキュリティ機能の中には、複数の処理方法が存在するものもあり、発行者、端末を管理する店舗等がリスク管理方針に応じて選択することが可能である。セキュリティ機能の具体的な内容は以下のとおりである。

イ．データ認証

EMV 仕様では、データ認証方法として、①SDA（Static Data Authentication＜静的データ認証＞）、②DDA（Dynamic Data Authentication＜動的データ認証＞）、③CDA（Combined DDA / Application cryptogram generation）の 3 種類が用意されている。

SDA は、カード発行時に、発行者が口座番号（PAN：Primary Account Number）・有効期限・名前等の情報（静的認証データ）とこれに対応する発行者のデジタル署名をカードに登録する。その後、カード利用時に、カードに対応する公開鍵を用いて端末がデジタル署名を検証し、カードに登録された情報が改ざんされていないことを確認する。

SDA の場合には、デジタル署名の検証時に、カードから固定化されたデータが送信される。こうした固定化されたデータの代わりに、秘密の情報を持つ真正なカードだけが生成可能な取引毎に異なるデータを検証時に利用することで、より安全性の高い認証を行う方法が DDA である。具体的には、カード発行時に、発行者がカードに固有の秘密鍵を登録する。その後、カード利用時に、端末がカードに乱数等を送ると、カードが秘密鍵を用いてデジタル署名を施し、これ

を秘密鍵に対応する公開鍵とともに送り返す。端末は、このデジタル署名と公開鍵を検証することにより、カードが秘密の情報を持った真正なものであることを確認するという仕組みである。CDA は、DDA と後述する AC 生成をあわせて行うことにより、カード・端末間の通信回数を削減することを企図した認証方法である。

これらの 3 種類のデータ認証において利用される公開鍵の正当性を保証するために、公開鍵証明書が用いられる。

ロ．カード所持者認証

EMV 仕様では、現在、カード所持者認証の方法として、①オフライン PIN 認証、②オンライン PIN 認証、③手書き署名等が用意されている[2]。

オフライン PIN 認証では、まずカード発行時に、利用者（または発行者）が決めた暗証番号（PIN : Personal Identification Number）[3]をカード内に登録しておく[4]。その後、カード利用時に、利用者が PIN パッドから PIN を入力すると、端末がそれをカードに送る。これを受けて、カードが内部で PIN の照合を行い、その結果を端末に返すという流れで認証が行われる。

オンライン PIN 認証では、利用者が PIN パッドに入力した PIN を用いて、端末・発行者ホストシステム間で認証処理が行われるため、EMV 仕様とは別に、その詳細が各々のシステムやネットワークの仕様の中で決められている。また、手書き署名についても、EMV 仕様のほか、カードによる照合処理について別の仕様で規定されている。

ハ．AC 生成

AC は、取引データが改ざんされていないことと、当該取引のためにカードが利用されていることを検証する手段として、取引毎に生成される。まず、あらかじめカードに登録された「マスタ鍵」とカード自身が管理する取引カウンタ（ATC : Application Transaction Counter）のデータをもとに、取引毎に「セッション鍵」が生成される。この「セッション鍵」と端末から送信されてきた取引データ（取引金額、取引日時等）をもとに、AC が生成されるという仕組みである。

---

[2] 現時点では、認証方法として①～③が用意されているが、EMV 仕様では、今後、新たな認証方法を追加することも可能となっている。

[3] PIN には 4 桁の数字（0～9）が利用されるケースが多いが、EMV 仕様は ISO 9564 シリーズ（Banking – PIN management and security）に準拠しているため、4～12 桁の数字が利用可能である。

[4] カード発行後に、利用者が端末等で PIN の変更を可能とする運用もある。

なお、「マスタ鍵」は、発行者が管理する「システム鍵」からカード毎に生成される。AC 生成に必要となる鍵の体系は、図 2 に示すとおり、「システム鍵」、「マスタ鍵」、「セッション鍵」から成るツリー構造となっている。

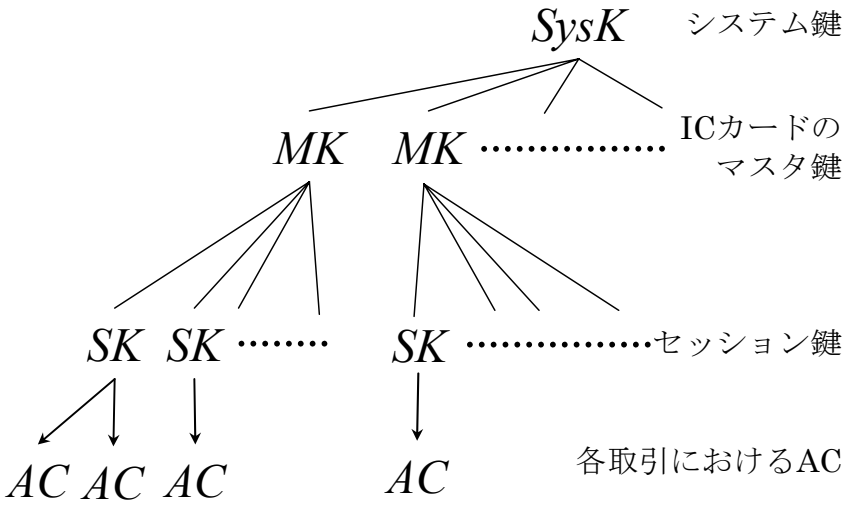

図 2. AC 生成に関連する鍵の体系

３．EMV 仕様で利用される暗号アルゴリズムの安全性

EMV 仕様では、各セキュリティ機能を提供するうえで推奨される暗号アルゴリズムを列挙している。これらの暗号アルゴリズムについては、定期的な見直しが行われているものの、最初の EMV 仕様が策定・公開されてから既に 10 年以上が経過していることから、現在の暗号技術の水準からみれば、必ずしも安全とは言い切れないものも含まれている。これは、EMV 仕様の問題というよりも、むしろ EMV 仕様のもとで各システムに応じた暗号アルゴリズムを選択する際のシステム設計上・運用上の問題であり、そうした問題を意識しておくことが EMV 仕様を利用していくうえで重要であると考えられる。

以下では、EMV 仕様で推奨されている暗号アルゴリズム等に着目して、その安全性について若干の検討を行う。

（１）公開鍵方式に対して想定される攻撃

イ．EMV 仕様で利用される公開鍵暗号

EMV 仕様では、データ認証において公開鍵方式の暗号アルゴリズムである RSA 署名が使用されている。すなわち、公開鍵の正当性を保証するための公開鍵証明書、SDA における静的認証データに対するデジタル署名、DDA における乱数等に対するデジタル署名に RSA 署名が用いられている。

仮に、RSA 署名を偽造可能な攻撃者が存在した場合には、公開鍵証明書や、静的認証データ、乱数等に対する署名の偽造が行えるため、データ認証において、真正であると誤認させることができる IC カードの作製が可能になると考えられる。

ロ．具体的な攻撃とそれへの対策

RSA 署名に対する署名の偽造方法は、①秘密鍵を求めることなく、署名を偽造する攻撃（署名偽造攻撃）と、②RSA 署名が安全性の根拠としている素因数分解問題[5]を解くことで、公開鍵から秘密鍵を求め、署名を偽造する攻撃（素因数分解攻撃）に分類できる。それぞれの攻撃は次のとおりである。

[5] 素因数分解問題：自然数 n が与えられたとき、$n = p\,q$ を満たす、相異なる素数 p、q を求めるという問題。

（イ）署名偽造攻撃

①ナイーブな RSA 署名方式に対する攻撃（積攻撃）

RSA 署名をその原論文（Rivest, Shamir, and Adleman [1978]）のとおりに何の工夫もなく利用することを「ナイーブな RSA 署名方式」と呼ぶ[6]。この署名方式には、「(mod n において）メッセージ m_1 とメッセージ m_2 の積で表されるメッセージ m_1 m_2 に対する署名と、m_1 に対する署名と m_2 に対する署名の積が等しい」という性質（乗法性）がある。そのため、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 $x\,m$ (mod n)に対する署名を入手することによって、x の署名を偽造できる。

②デスメット＝オドリズコ攻撃（Desmedt and Odlyzko [1986]）

積攻撃への耐性を持たせるために、メッセージそのものではなく、メッセージのハッシュ値を RSA 署名への入力とした署名方式が提案された。しかし、この署名方式に対しても、別の署名偽造攻撃が発見された。

攻撃者は、メッセージ m のハッシュ値が比較的小さな素数の積 $p_1 \times p_2 \times \cdots \times p_k$ となるようなメッセージを大量に生成し、署名生成者にこれらに対する署名 $s := {p_1}^d \times {p_2}^d \times \cdots \times {p_k}^d \pmod n$ を生成させる（ただし、p_i は素数、$1 \leqq i \leqq k$）。そして、入手した複数の署名を使って、各要素 p_i を d 乗した値（${p_i}^d \bmod n$）を得る。攻撃者は、入手した(${p_i}^d \bmod n$)の値を組合わせることで、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できる。

RSA 署名では、署名に入力するデータ（以下、署名変換データ）の生成に、ハッシュ値だけでなく、パディングを利用する方法や乱数を用いる方法などの様々な方式も提案されてはいるが、それらに対する署名偽造攻撃が発見されてきた[7]。

次に、EMV v4.0（EMVCo [2000]）が採用していた ISO/IEC9796-2:1997 署名方式に対する攻撃と、近年発見された新たな署名偽造攻撃（ブライヘンバッハーの攻撃、Bleichenbacher [2006]）についてみていく。

---

[6] ナイーブな RSA 署名における、メッセージ m に対する署名 s は、秘密鍵 d を用いて $s := m^d \bmod n$ より得られる。署名偽造攻撃②～④で紹介する各署名方式では、署名変換データ SR に対して、署名 $s := SR^d \bmod n$ を生成する。その場合、署名の検証は、公開鍵 e を用いて $SR' := SR^e \bmod n$ を計算したうえで、SR' と SR の対応関係を確認することで行われる。

[7] 様々な RSA 署名方式への攻撃法とそれらに対応した改良の経緯については、齊藤 [2002]に詳しい。

③ISO/IEC 9796-2:1997 署名方式に対する攻撃（CNS 攻撃）

ISO/IEC9796-2:1997 署名方式における署名変換データは、次のとおりである。

ISO/IEC 9796-2:1997 の署名変換データ

（公開鍵サイズを$|n|$=1,024、ハッシュ関数 H を SHA-1 とした場合）

署名変換データ $SR_{ISO} := 01 \| 1 \| 0\ 1010 \| m$ の左 848 bit $\| h \|$ 0xBC

ただし、各フィールドは、左から順に以下の通りとする。

・ヘッダ：01　（2 bit）

・フラグ：メッセージ全体を復元できる場合は 0、そうでない場合は 1。

・パディング：$|SR_{ISO}| = |n| = 1{,}024$ となるようにパディングを施す。

・復元されるメッセージ：メッセージ m の左から 848 bit 分。

・ハッシュ値：h　（160 bit）。

・トレイラ：0xBC　（8 bit）。

この署名方式に対する署名偽造攻撃（CNS 攻撃）が、Coron, Naccache, and Stern [1999]において示されている[8]。CNS 攻撃では、デスメット=オドリズコ攻撃同様、署名変換データが小さな素数の積のみから構成されるメッセージを見付け出し、それに対する署名を集めることにより、署名変換データが小さい素因数で構成されるメッセージに対する署名を偽造することが可能である。

④PKCS#1 v1.5 署名方式[9]に対する攻撃（ブライヘンバッハーの攻撃）

PKCS#1 v1.5 署名方式は、長年利用され続けても素因数分解問題を解くよりも効率的な攻撃が発見されていないという実績を持つ署名方式として、広く用いられてきた。しかし、この署名方式に対しては、最近、公開鍵 e が小さく（$e = 3$）、検証処理が不適切な場合に、署名を偽造できる可能性があることが示されている。攻撃対象とされた PKCS#1 v1.5 署名方式は、次のような構造の署名変換データを用いている。

[8] CNS 攻撃については、宇根 [1999]に詳しい。

[9] PKCS#1 v1.5（Public Key Cryptography Standards #1 version 1.5）：PKCS は、米 RSA 社が定める公開鍵暗号技術をベースとした規格群のことであり、PKCS#1 は、RSA を用いた暗号化方式とデジタル署名方式を定めている。特に、PKCS#1 v1.5 の署名方式は、S/MIME（守秘・認証等のセキュリティ機能を持つ電子メール用プロトコル、＜Secure Multipurpose Internet Mail Extension＞）に実装される等、業界標準として広く利用されている。

PKCS#1 v1.5 署名方式の署名変換データ

署名変換データ SR_{org} := 0x00 0x01 || *padding* || 0x00 || ID_H || h

ただし、*padding* は、|*padding*|≧64、かつ、$|SR_{org}| = |n|$を満たす 0xFF からなるビット列、n は公開鍵、ID_H はハッシュ関数 H の ID、h はメッセージ m に対するハッシュ値 H(m)とする。

通常、検証時には、署名変換データの長さをチェックし、ハッシュ値が最後にあることを確認する必要がある。この攻撃では、何らかの理由により不適切な実装が行われ、署名変換データの左端から各ビットを検査し、ハッシュ値を抽出したところで、ハッシュ値の検査に移り、ハッシュ値が署名変換データの最後にあるのか（つまり、ハッシュ値より右に余分なデータが存在しないか）を検査しない場合において、署名を偽造できるケースがあることを示している。具体的には、署名変換データの *padding* を短くし、その分をハッシュ値の右側に任意の値を取れる余分なデータ（*garbage*）として加えた署名変換データを用いても、*garbage* をチェックされることがないことから、*garbage* をうまく設定することで署名の偽造が行える可能性があることを示している。

ブライヘンバッハーの攻撃

公開鍵を $e = 3$、検証時にハッシュ値より右のビット列の検査をしないと仮定し、公開鍵サイズを$|n| = 3{,}072$ とする。

SR_{org} の *padding* を短くすることで、任意の値を取ることができるビット列 *garbage* を SR_{org} の末尾に付加し、次のデータを署名変換データ SR_{mod}（$|SR_{mod}| = |n|$）として生成する。

署名変換データ SR_{mod} := 0x00 0x01 || *padding* || 0x00 || ID_H || h || *garbage*

Step S1、S2、V2 では、ID_H、h、*garbage* を整数とみなして計算することとする。

＜署名＞

Step S1. 偽造対象メッセージを x とし、ハッシュ値 h := H(x)を計算する。

整数 $a := 2^{288} - (ID_H \times 2^{160} + h)$を計算し、$a$ が 3 の倍数でない場合は、メッセージ x を修正することで、a が 3 の倍数となるようにする。

Step S2. $s := 2^{1019} - a/3 \times 2^{34}$ を計算する。

＜検証＞

Step V1. ハッシュ値 h := H(x)を計算する。

Step V2. s^3 を計算する。

$s^3 = (2^{1019} - a/3 \times 2^{34})^3$

$= 2^{3057} - a \times 2^{2072} + a^2/3 \times 2^{1087} - a^3/27 \times 2^{102}$

$= (2^{985} - 2^{288} + ID_H \times 2^{160} + h) \times 2^{2072} + garbage$

ただし、$garbage := a^2/3 \times 2^{1087} - a^3/27 \times 2^{102}$ とする。

Step V3. 0x00 0x01 || *padding* || 0x00 || ID_H || h と一致するかを検証する

以上より、$2^{985} - 2^{288} + ID_H \times 2^{160} + h$ が、0x0001 || *padding* || 0x00 || ID_H || h の形式になっているため、見かけ上（*garbage* の部分以前のデータにおいて）s がメッセージ x に対する正しい署名と一致している。

この攻撃では、公開鍵 e として $e = 3$、ハッシュ値の右側にデータがあるかを検査しないことを仮定し、公開鍵サイズを$|n| = 3{,}072$ ビットとしている。$e = 3$ が有効であるのは、Step. S1 で偽造対象メッセージ x を求める際に、a が 3 の倍数になればよいため、平均 3 回程度の試行ですむからである。

この点、EMV v4.1 では、発行者の公開鍵に対する公開鍵証明書について[10]、ISO/IEC 9796-2:2002 における署名変換データを次のように利用している。

EMV v4.1 における発行者の公開鍵に対する公開鍵証明書

署名変換データ SR_{EMV} := 0x6A || 属性情報 || 発行者公開鍵(n, e)のサイズ || 発行者公開鍵 || $padding_{EMV}$ || ハッシュ値 || 0xBC

ただし、|発行者公開鍵| + | $padding_{EMV}$| = $|n|$ – 288 bit であり、パディングの値は 0xBB である。なお、n は CA の公開鍵である。

EMV v4.1 においても、公開鍵 e として $e = 3$（または、$2^{16} + 1$）を用いている。また、EMV v4.1 では、公開鍵証明書の検証手順において、$padding_{EMV}$ の値と長さを検査することは規定されていない。そのため、$padding_{EMV}$ を検査していない実装も少なくないと考えられる。この場合、攻撃者が故意に鍵長の短い公開鍵を発行者の公開鍵として設定することで、$padding_{EMV}$ のビット列を長く設定し、任意の値を $padding_{EMV}$ に設定できる可能性がある。

このように、EMV 仕様の公開鍵証明書における条件とブライヘンバッハーの攻撃の前提条件とは若干異なるが、ブライヘンバッハーの攻撃のアイデアを利用した類似の攻撃を受けることがないか確認しておく必要がある。仮に、こうした攻撃が考えられる場合、実装において検証時にパディング領域の検査を行う等の対策を講じていくことが一案であろう。

なお、1993 年に、暗号アルゴリズムが安全性の根拠としている数学的問題を

[10] 発行者の公開鍵証明書だけでなく、IC カードの公開鍵証明書についても同様の議論を行うことができる（EMVCo [2004] pp.61～63）。

解く難しさと、暗号アルゴリズムの安全性が等価であること、言い換えると、署名偽造攻撃が存在しないことを数学的に証明する概念（証明可能安全性）が提案された[11]。証明可能安全性を持つ RSA 署名方式としては、RSA-PSS 署名方式[12]等が挙げられる。

（ロ）素因数分解攻撃

素因数分解問題を解く困難さは、合成数の大きさ（公開鍵のサイズ）に依存しており、現在知られている素因数分解アルゴリズムを利用した素因数分解を実際に行うことで、どの程度の大きさの合成数であれば現実的な時間で素因数分解可能であるかによって評価されている。最新（2005 年 5 月）の実験結果によると、663 ビットの合成数（公開鍵）を実際に素因数分解できたことが報告されている。

公開鍵の素因数分解をいつのタイミングでどの程度の長さまで行うことができるかについては、様々な予測が行われている。図 3 の曲線(b)は、実際に素因数分解された最大の値を出発点とし、解読技術は現在のままで、コンピュータの性能向上がムーアの法則[13]に従うことを仮定した場合に素因数分解が可能と考えられる公開鍵サイズを示している（Brent [2000]）。図 3 の曲線(a)は、上記のムーアの法則に加え、過去の経験則に基づき、素因数分解アルゴリズムの効率が 1.5 年で 2 倍になることを仮定に加えたうえで、1982 年当時の DES と同程度の安全性をもつと考えられる公開鍵サイズを予測している（Lenstra and Verheul [2001]）。

EMVCo では、公開鍵認証局（CA）[14]の公開鍵サイズに関する見直しを毎年行っている（次ページ表 1 参照）。曲線(a)(b)と表 1 を比較すると、EMV 仕様が規定する公開鍵サイズは、曲線(b)よりむしろ曲線(a)に近いセキュリティ・マージンを確保したうえで設定しようとしていることがわかる。

---

[11] Bellare and Rogaway [1993].

[12] RSA-PSS 署名方式（RSA-Probabilistic Signature Scheme）：ランダムオラクルモデルの下、RSA 暗号関数が一方向性を有していると仮定した場合に、「能動的攻撃が可能な環境であったとしても、いかなるメッセージに対する署名も偽造することができないことが証明されている（Bellare and Rogaway [1996]）。

[13] ムーアの法則：半導体の集積密度が 1.5 年～2 年で倍増するという予想に基づく法則。これに基づき、一定の金額で購入できるコンピュータの処理速度が 1.5 年～2 年で 2 倍になることを想定している。

[14] 公開鍵認証局（CA＜Certificate Authority＞）：発行者と IC カードの公開鍵の正当性を保証するための大元（ルート）となる機関。

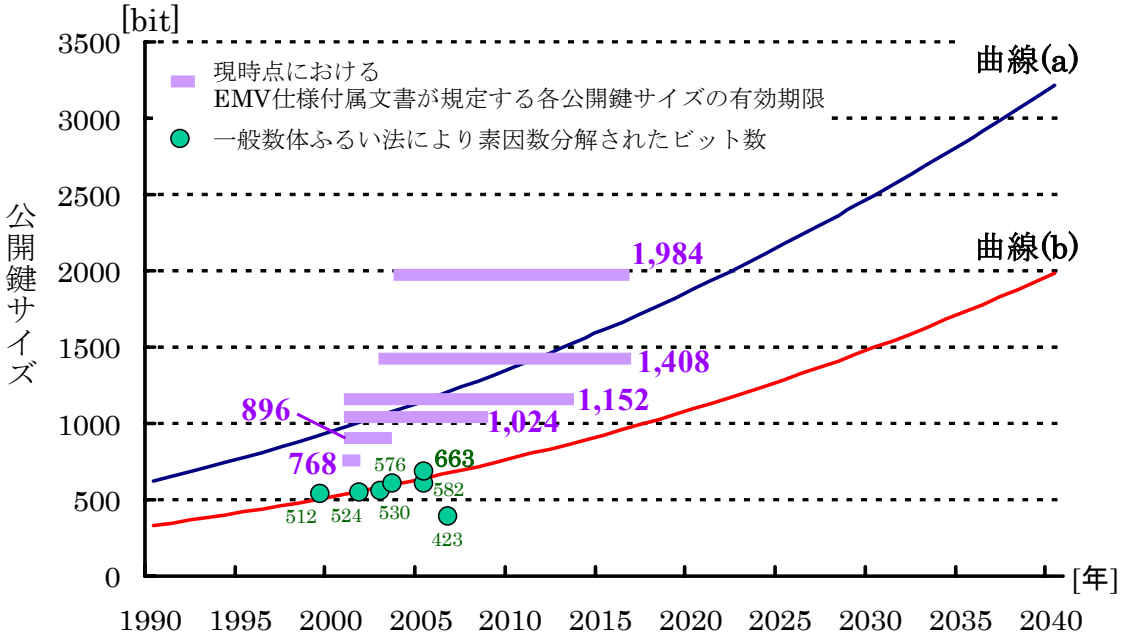

図 3. RSA の公開鍵サイズに関する予測

公開鍵サイズ [bit]	No.2 2001/11	No.3 2002/11	No.4 2003/9	No.5 2003/12	No.6 2004/2	No.8 2004/12	No.9 2005/10	No.10 2006/10
786	2002	2002	—	—	—	—	—	—
896	2004	2004	2004	2004	2004	2004	—	—
1,024	2008	2008	2009	2009	2009	2009	2009	2009
1,152	2010	2010	2010	2012	2012	2012	2013	2014
1,408	—	—	—	2014	2014	2014	2016	2017
1,984	—	—	—	—	2016	2016	2016	2017

表 1. EMV 仕様の各付属文書（No.2～10）が規定する CA の公開鍵サイズ

備考：EMVCo [2007]より、該当箇所を抜粋。

各セルは、該当する公開鍵サイズの有効期限を示す。

例えば、“2009”は、2009 年 12 月 31 日まで有効であることを示す。

ハ. 小括

EMV 仕様に基づく IC カードが使用している RSA 署名では、2010 年問題の影響もあり、鍵長に対する関心が集まっている。上に示したとおり、素因数分解攻撃による RSA 署名の解読リスクという観点から考えると、RSA 署名の鍵長は現行の 1,024 ビットから、今後、より長い鍵長に移っていくことが望ましい。

また、EMV 仕様に基づいて開発された IC カードが署名生成においてどのよ

うな署名方式を採用しているのかについての検討も重要である。署名変換データを対象にした攻撃に対しては、証明可能安全性を持つ署名方式を採用することが最も有効であるが、そうでない場合でも、署名変換データのパディング方法やハッシュ関数の使い方等、鍵長以外の署名方式の特徴についても、安全性の観点から検討しておくことが重要である。

（２）共通鍵方式に対して想定される攻撃

イ．EMV 仕様で利用される共通鍵暗号

EMV 仕様では、発行者のシステム鍵から各 IC カードのマスタ鍵を生成する段階（マスタ鍵生成）と、各 IC カードにおいて、マスタ鍵から取引毎にセッション鍵を生成する段階（セッション鍵生成）において、2-key トリプル DES（以下、2-key TDES）が使用されている。また、取引時に生成する AC（Application Cryptogram、２節（２）ハ．参照）には、2-key TDES をベースにしたメッセージ認証子生成方式と、DES をベースにしたメッセージ認証子生成方式が推奨されている。

２節（２）ハ．で述べたとおり、AC 生成に至るプロセスでは、システム鍵、マスタ鍵、セッション鍵という、上位鍵から下位鍵の生成の連鎖を経ている。このため、AC あるいは下位鍵から上位鍵を特定できるような場合は、こうした生成の連鎖を経て、最終的に別の AC が生成される可能性がある。これにより、端末から送られてきた取引データに対する AC を偽造することや、悪意のある端末管理者が取引データを改ざん（例えば、取引金額を増やす等）し、それに対する AC を偽造すること等が考えられる。

ロ．具体的な攻撃とそれへの対策

以下では、EMV 仕様が例示するマスタ鍵とセッション鍵それぞれについて、鍵生成に使用した上位鍵を特定する攻撃と、EMV 仕様が推奨するメッセージ認証子生成方式に対してセッション鍵を求める攻撃について考察する。

（イ）マスタ鍵生成方式に対する攻撃

最新の EMV 仕様では、発行者がシステム鍵を用いて、各 IC カードのマスタ鍵を生成する方法を例示している（次ページ図 4 参照、EMVCo [2004] p.134）。

IC カードのマスタ鍵の生成

2-key TDES の暗号化関数を E_{TDES}、発行者が持つシステム鍵を *SysK*（$|SysK| = 128$）、IC カードのマスタ鍵を *MK*（$|MK| = 128$）、IC カードの口座番号を *PAN*、PAN シーケンスを *Seq* とする。なお、平文 *x*、鍵 *z* に対し暗号化関数 E を利用して得られた暗号文を $y := E(z; x)$ と表記する。同様に、復号関数 D に対しても、$x := D(z; y)$ と表記し、以下同様とする。

このとき、IC カードのマスタ鍵 *MK* は、次式で与えられる。

$$MK := E_{TDES}(SysK; (PAN, Seq))$$

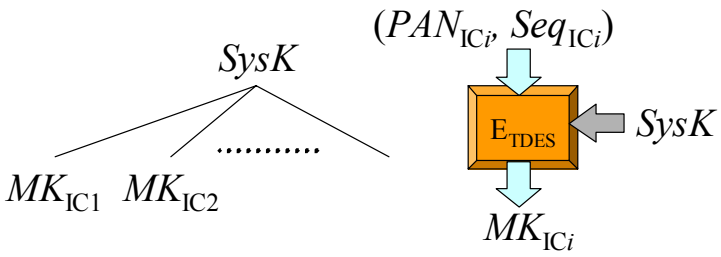

図 4．IC カードのマスタ鍵の生成

2-key TDES における平文と暗号文は、マスタ鍵生成における PAN と PAN シーケンスと、マスタ鍵にそれぞれ対応している。PAN と PAN シーケンスと、マスタ鍵の組が何らかの方法で入手できたとすると、2-key TDES に対する既知平文攻撃が可能となる。この既知平文攻撃については、攻撃者が N 組の既知平文を入手した場合、2 の$(120 - log_2 N)$乗の計算量で鍵を特定できる方法が知られている（Oorshot and Wiener [1990]）。この方法は、例えば 2^{80} 程度の計算能力を持つ攻撃者を仮定した場合、2^{40} 程度の平文・暗号文ペアが入手できれば、システム全体で利用する秘密鍵（*SysK*）が解読できてしまうことを意味している。実際には、2^{40} 程度の平文・暗号文ペアを入手することは困難と考えられるが、より効率的な 2-key TDES の解読法が発見されたり、PAN と PAN シーケンスが大量に入手できる条件が成立した場合には、システム全体が危機を迎えることになる。こうしたリスクが発生しないよう、十分な注意が必要である。

なお、2-key TDES を利用するのではなく、AES や Camellia 等のより安全な暗号アルゴリズムを採用した場合は、これを計算量的に防ぐことができる。システムの性能要件や運用要件等の許容範囲内であるならば、より安全な方式に移行することが本来は望ましいといえるであろう。

（ロ）セッション鍵生成方式に対する攻撃

最新の EMV 仕様では、セッション鍵からマスタ鍵の特定が困難となる鍵生成方式についても例示している（次ページ図 5 参照、EMVCo [2004] pp.130～131）。

2-key TDES を用いたセッション鍵の生成

2-key TDES の暗号化関数を E_{TDES}、IC カードのマスタ鍵を MK ($|MK| = 128$)、取引カウンタ ATC の値を c ($0 \leqq c \leqq 2^{16} - 1$)とする。関数 Φ を次のように定義する。

$$\Phi(x, y, j) := (E_{TDES}(x; y_l \oplus (j \bmod 4)) \,||\, E_{TDES}(x; y_r \oplus (j \bmod 4) \oplus (0xF0)))$$

ただし、$|x| = 128$、$|y_l| = |y_r| = 64$、$y = y_l \,||\, y_r$ とする。

中間鍵 IK を次式により求める。

$IK_{1,j} := \Phi(MK, 0^{128}, j), j = 0, 1, 2, 3.$

$IK_{i,j} := \Phi(IK_{i-1,j/4}, IK_{i-2,j/16}, j), 0 \leqq j \leqq 4^i - 1, 2 \leqq i \leqq 8.$

ここで、$j := c$ と置き、次式によって、ATC の各値 c に対応したセッション鍵 SK_c を得る。

$SK_c := IK_{8,c} \oplus IK_{6,c/16}$

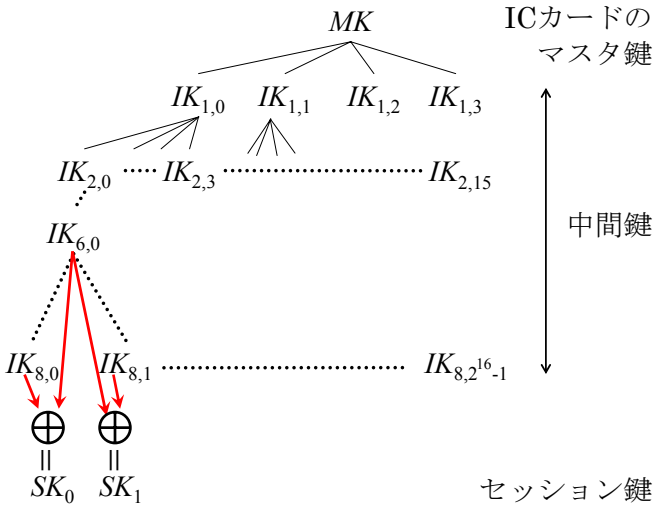

図 5. EMV 仕様で示されているセッション鍵生成方法

セッション鍵からマスタ鍵を特定することが可能か否かについてみると、本方式では、攻撃者があるセッション鍵 SK_c を入手できたとしても、$IK_{8,c}$ と $IK_{6,c/16}$ の 2 元 1 次方程式を解くことになり、中間鍵 IK を求められないため、マスタ鍵 MK を特定することが困難である。複数のセッション鍵が入手できたとしても、変数となる中間鍵の種類がその分増えるため、同様に特定できない。

しかし、EMV 仕様では、発行者が独自の鍵生成方式を採用することを認めているため、発行者が、「セッション鍵からマスタ鍵の特定が困難であること」を満たさないセッション鍵生成方式を採用することで、マスタ鍵が特定できる可能性がある。こうした不適切な鍵生成方式の例を挙げると以下のとおりである（次ページ図 6 参照）。

不適切なセッション鍵生成

IC カードのマスタ鍵を MK (= $mk_1 \| mk_2$)、セッション鍵を SK、取引カウンタ ATC の値を c ($|c| = 16$)とする。ただし、$|mk_1| = |mk_2| = |sk_1| = |sk_2| = 64$。このとき、$SK$ を次式より得る。

$sk_1 := mk_1 \oplus (0^{48} \| c)$

$sk_2 := mk_2 \oplus (0^{48} \| (c \oplus 1^{16}))$

$SK := sk_1 \| sk_2$

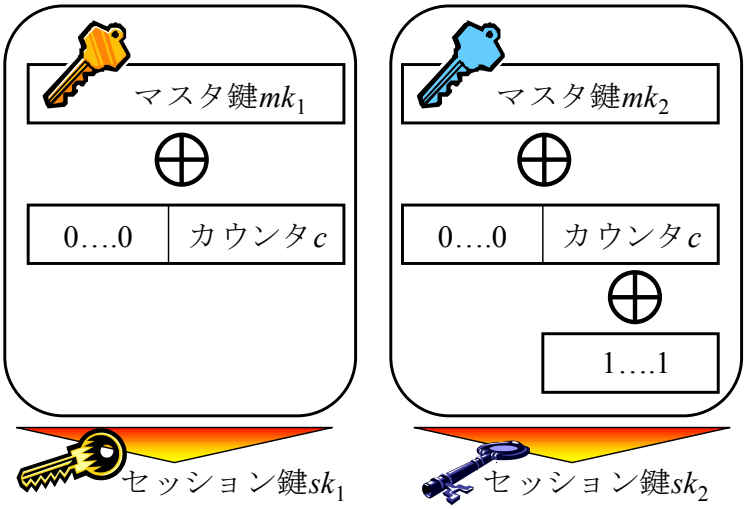

図 6. 不適切なセッション鍵生成

このような不適切なセッション鍵生成を利用している場合は、仮に、IC カードのセッション鍵 *SK* が露呈したとすると、攻撃者は IC カード・端末間を流れる取引カウンタの値 c を盗聴し、$mk_1' := sk_1 \oplus (0^{48} \| c)$、$mk_2' := sk_2 \oplus (0^{48} \| (c \oplus 1^{16}))$ を計算することで、マスタ鍵を特定できる。マスタ鍵を特定した攻撃者は、端末から送られてくる取引データに対して、マスタ鍵からセッション鍵を生成することで、AC を偽造できる可能性がある。

発行者が EMV 仕様に例示されたセッション鍵生成方式に変更を加えた場合、それが僅かであってもセッション鍵からマスタ鍵が特定されるといった深刻な脅威に繋がる可能性がある。万一、IC カード実装上の制約があったとしても、暗号鍵の管理については、発行者は十分な注意が必要である。

（ハ）メッセージ認証子生成方式に対する攻撃

最新の EMV 仕様である EMV v4.1 が検討された当時は、既に DES の安全性の低下が広く認識されていた[15]。このため、EMVCo では、DES 以上の安全性を持つ方式を採用することが基本的な方針として示されている。

EMV v4.1 では、AC 生成に利用するメッセージ認証子生成方式として、2 種類のアルゴリズムを推奨している。第 1 は、DES に代わる、より安全性の高い 2-key TDES を用いた CBC-MAC（以下、TDES-MAC、次ページ図 7-(a)参照）である。第 2 は、DES を用いた CBC-MAC の最後に DES の処理を 2 回加えることにより、最終処理が 2-key TDES の形式を備えているアルゴリズムである

[15] 1998 年に、解読専用ハードウエア（いわゆる、DES Cracker）を用いることで、鍵全数探索攻撃によって 56 時間で DES が解読されている。

（本稿では、これを擬似 TDES-MAC と呼ぶ、図 7-(b)参照）[16]。この方式は、IC カードの計算能力が低い場合に配慮して用意された方式と考えられる。具体的な擬似 TDES-MAC の処理は以下のとおりである。

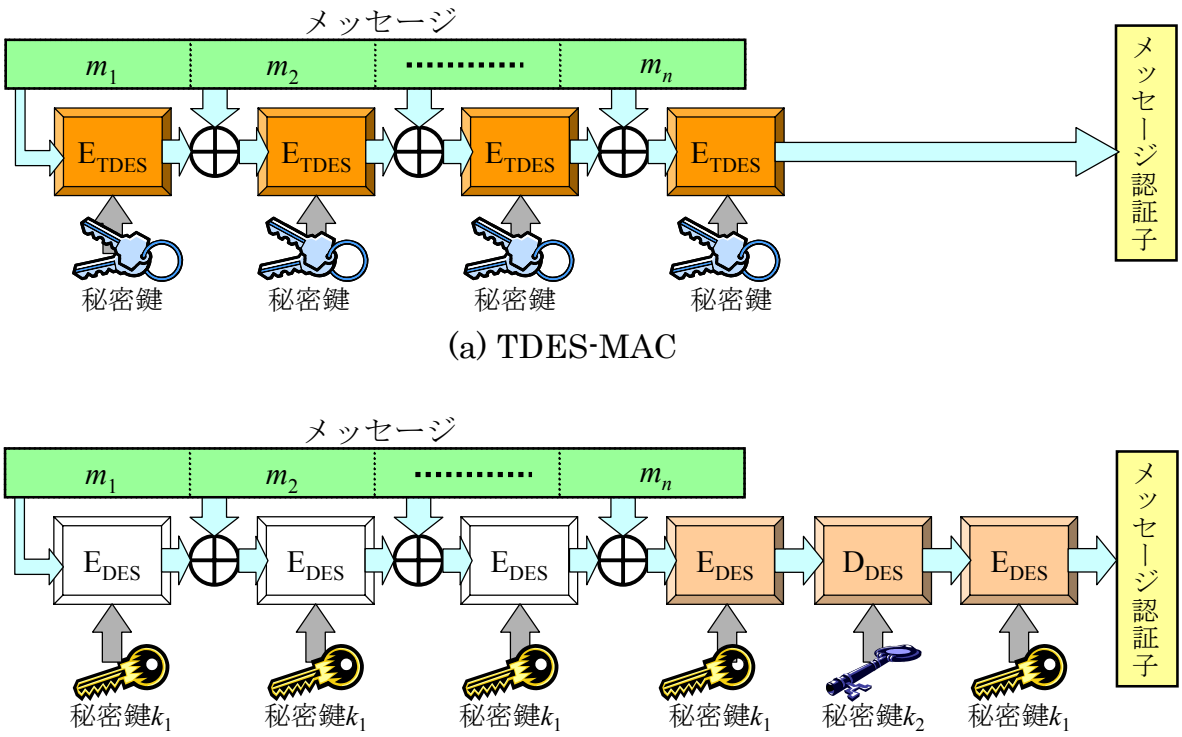

図 7. TDES-MAC と擬似 TDES-MAC

擬似 TDES-MAC

DES の暗号化関数を E_{DES}、DES の復号関数を D_{DES}、メッセージを $m = (m_1, m_2, \ldots, m_n)$、鍵を $k\ (= k_1 \| k_2、|k_1| = |k_2| = 64)$、とすると、メッセージ認証子 d は、次のように計算される。

$d_i := E_{DES}(k_1; m_i \oplus d_{i-1}),\ i = 1, 2, \ldots, n,\ d_0 = 0^{64}$

$d := E_{DES}(k_1; D_{DES}(k_2; d_n))$

この擬似 TDES-MAC に対しては、ある条件の下で、DES と同程度の計算量で鍵の特定が可能となる攻撃が考えられる。

[16] 擬似 TDES-MAC は、ISO/IEC 9797-1 で規定されている「アルゴリズム 3」をベースとし、ブロック暗号として DES を適用した方式である。

擬似 TDES-MAC への攻撃

Step 1. ランダムにメッセージを生成し、擬似 TDES-MAC によりメッセージ認証子を求める。メッセージ認証子の長さが 64 bit であることから、ランダムに生成した 2^{32} 個のメッセージのメッセージ認証子を集めれば、バースディ・パラドックスの原理[17]により、同じ値のメッセージ認証子となるメッセージのペア(m, m')が 50%以上の確率で見つけられる。

このとき、$d := E_{DES}(k_1; D_{DES}(k_2; d_n)) = E_{DES}(k_1; D_{DES}(k_2; d'_n))$となる。ただし、$d_n$ は、$d_i := E_{DES}(k_1; m_i \oplus d_{i-1})$, $i = 1, 2, \ldots, n$, $d_0 = 0^{64}$ より得られる。d'_n についても同様である。

Step 2. メッセージのペア(m, m')に対して、$d_n = d'_n$ を満たす鍵 k_1 を鍵全数探索で求める。DES の鍵長が 56 bit であることから、2^{56} の探索で k_1 を特定できる。

Step 3. 求めた鍵 k_1 を使って、$D_{DES}(k_1; d)$より $D_{DES}(k_2; d_n)$が求まる。$D_{DES}(k_2; d_n)$ と d_n から、鍵全数探索を行い 2^{56} の探索で鍵 k_2 を特定できる。

この攻撃により、擬似 TDES-MAC は、112 ビットの鍵を利用しているにもかかわらず、2^{32} 個のメッセージとメッセージ認証子のペアを集められるならば、DES と同程度の計算量、すなわち、2^{57}（$= 2^{32} + 2^{56} + 2^{56}$）で 112 ビットの鍵を特定することができる。このことから、EMV 仕様が推奨している擬似 TDES-MAC は、一定条件の下では、DES と同程度の安全性しか持たないことがわかる。

擬似 TDES-MAC への攻撃に必要となるメッセージとメッセージ認証子のペアは、AC 生成における取引データと AC のペアに対応する。仮に、セッション鍵が常に固定であり、かつ、IC カードに何度も取引データに対する AC を生成させられる場合には、攻撃に必要な数のペアを攻撃者が集められる可能性があるため、実装の際には注意が必要である。ただし、TDES-MAC や、AES や Camellia 等の 128 ビット・ブロック暗号を用いた CBC-MAC 等を利用すれば、この攻撃自体を計算量的に防ぐことができるため、本質的には擬似 TDES-MAC を利用しないことが望ましいといえるであろう[18]。

---

[17] バースディ・パラドックスの原理：A 個の値をとり得るデータから重複を許してランダムに選択した B 個の中に、同じデータが少なくとも 2 個以上存在する確率 P は、$P = 1 - e^{\wedge}(-A^2/2B)$ となる。この事実は、ランダムに 23 人集めると、1/2 以上の確率で同じ誕生日の人が 2 人以上いることから、バースディ・パラドックスと呼ばれる。

[18] TDES-MAC を用いたケースでは、攻撃には 2^{100} 以上の計算量が必要となる。また、128 ビット・ブロック暗号を用いた CBC-MAC のケースでは、そもそも攻撃以前に、攻撃に必要な取引データと AC のペアを 2^{64} 個集めなければならない。どちらのケースにおいても、現実的な時間やメモリでは、鍵を求めることが困難であると言える。

ハ．小括

暗号アルゴリズムの 2010 年問題（宇根・神田 [2006]）においても指摘されているように、2-key TDES の安全性が低下しつつある。EMV 仕様は、共通鍵暗号方式として、この 2-key TDES を採用していることから、上記のようなセキュリティ上の問題点を持つこととなった。IC カードや端末の性能要件や相互運用性の観点からみて可能である場合には、AES や Camellia のようなより安全性の高い暗号アルゴリズムを採用することが望ましい。

また、AC 生成方式として推奨している擬似 TDES-MAC は、DES を利用しつつ 2-key TDES と同水準の安全性を確保しようとするものである。しかし、ある条件の下では DES と同程度の安全性しか得られていないことが判明している。したがって、本来はより安全な方式を利用することが望ましいが、現在のものを使い続ける必要があるような場合には、こうした暗号を利用していることを考慮し、慎重に運用する必要があろう。

さらに、EMV 仕様が例示する鍵生成等に関する方式は、僅かな変更を加えただけで、システムの安全性に重大な影響も与える可能性がある点についても、十分に留意する必要があろう。

４．おわりに

本稿では、EMV 仕様を実装する場合の安全性について、主として暗号アルゴリズムの観点から考察してきた。暗号アルゴリズムの安全性については、いわゆる 2010 年問題のような鍵長の観点に加え、署名変換データやメッセージ認証子の生成方法等についても考慮する必要があることが明らかとなった。

暗号アルゴリズムの安全性低下等の理由から、システムに一旦実装した暗号アルゴリズムを別のものに移行する場合には、そのシステム規模が大きければ、多大なコスト負担を要することとなる。こうしたコスト負担を可能な限り回避するという意味において、システムに実装する暗号アルゴリズムの選択はシステム投資判断の重要な要素である。暗号アルゴリズムの選択に当たっては、当該システムの使用期間を考慮しつつ、使用期間満了までの安全性が確認されている暗号アルゴリズムの中から、十分な安全性を有するものを選択することが望ましいといえる。

もっとも、システム構築段階で適切な暗号アルゴリズムを選択したとしても、新たな攻撃技術の発見やコンピュータの性能向上により、システムで使用している暗号アルゴリズムの安全性が低下することもありうる。このような場合、安全な暗号アルゴリズムに移行することが本来望ましいが、既存システムとの相互運用性の確保や実装技術上の問題から直ちに移行することが難しい状況も考えられる。そうした場合には、安全性の低下した暗号アルゴリズムへの攻撃の前提条件を充足しないように、システムの運用に制約を設けることが次善の策として考えられよう。例えば、攻撃に必要な暗号文と平文のペアが集まらないよう鍵を定期的に更新する、特定のパラメータの値を避けて利用するといった運用上の方策が考えられる。これらの方策を講じる場合には、具体的な攻撃を想定した適切な対応が必要となるため、暗号技術の専門的知見を十分に活用することが重要となろう。

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』、金融研究所、1999 年

――――・神田雅透、「暗号アルゴリズムにおける 2010 年問題について」、『金融研究第 25 巻別冊第 1 号』、金融研究所、2006 年

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』、金融研究所、2002 年

田村裕子・廣川勝久、「リテール・バンキング・システムの IC カード対応に関する現状とその課題」、IMES Discussion Paper Series No.2007-J-10、2007 年

Bellare, M. and P. Rogaway, "Random Oracles are Practical: A Paradigm for Designing Efficient Protocols," Proc. of 1st ACM Conf. on Computer and Communications Security, ACM Press, 1993, pp.62-73.

――――, and ――――, "The Exact Security of Digital Signatures – How to Sign with RSA and Rabin," Advances in Cryptology – Proceedings of EUROCRYPT '96, LNCS 1070, Springer-Verlag, 1996, pp.399-416.

Bleichenbacher, D., "Forging some RSA signatures with pencil and paper," CRYPTO 2006 Rump Schedule, 2006.

Brent, R., "Recent progress and prospects for integer factorization algorithms," Proc. of COCOON 2000, LNCS 1858, Springer-Verlag, 2000, pp.3-20.

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

Desmedt, Yvo, and A. M. Odlyzko, "A chosen text attack on the RSA cryptosystem and some discrete logarithm schemes," Proc. of CRYPTO'85, LNCS 218, Springer-Verlag, 1986, pp.516-522.

EMVCo, "EMV2000 Integrated Circuite Card Specification for Payment Systems: Book 2 – Security and Key Management," EMVCo, 2000.

――――, "EMV Integrated Circuite Card Specification for Payment Systems: Book 2 - Security and Key Management, EMVCo, 2004.

――――, "EMV Integrated Circuite Card Specification for Payment Systems: Common Payment Application Specification," EMVCo, 2005.

――――, *Bulletines - Notices*, 2007.
(http://www.emvco.com/bulletins.asp?show=14, access date: February 28, 2007)

International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC), "ISO/IEC 9797-1:1999 Information technology – Security techniques – Message Authentication Codes (MACs) – Part 1: Mechanisms using a

block cipher", ISO/IEC, 1999.

———— and ————, "ISO/IEC 9796-2:2002 Information technology – Security techniques – Digital signature schemes giving message recovery – Part 2: Integer factorization based mechanisms," ISO/IEC, 2002.

Lenstra, A. K. and E. R. Verheul, "Selecting Cryptographic Key Sizes," Journal of Cryptology, 14(4), 2001. pp.255-293.

Oorshot, P. C. van, and M. J. Wiener, "A known plaintext attack on two-key triple encryption," Advances in Cryptology - Proceedings of EUROCRYPT'90, LNCS, vol.473, Springer-Verlag, 1990, pp.318-325.

Rivest, R. L., A. Shamir and L. Adleman, "A method for obtaining digital signatures and public key cryptosystems," Communications of the ACM, vol. 21, ACM Press, 1978, pp.120-126.