# 電子政府推奨暗号の利用方法に関するガイドブック

平成 20 年 3 月

独立行政法人情報通信研究機構
独立行政法人情報処理推進機構

# 目　次

# 1. 本リストガイドの概要

## 1.1. リストガイドの目的

電子政府システムにおいて、安全な暗号技術を利用することを目的に、総務省および経済産業省が共同で開催する暗号技術検討会のもと、電子政府推奨暗号リストが 2003 年 2 月 20 日に公表された。また、2003 年 2 月 28 日に行政情報システム関係課長連絡会議において、各府省が情報システムの構築にあたり暗号を利用する場合には、可能な限り、電子政府推奨暗号リストに掲載された暗号の利用を推進する旨の「各府省の情報システム調達における暗号の利用方針」が了承された。

この電子政府推奨暗号リストは、安全性が確認された暗号アルゴリズムが列挙されている。一方、安全な電子政府システムを構築する際には、暗号アルゴリズムが組み合わされて使われているセキュリティの標準技術が調達の際の選択候補となる。そのため、構築する電子政府システムの安全性を確認するためには、暗号アルゴリズムの安全性とセキュリティの標準技術の関係を理解したうえで、推奨される標準技術を利用することが必要となる。

一方、2003 年 3 月に電子政府システムを調達する際に、調達プロセスの中で暗号技術の選定を行う方法について記述した「暗号調達のためのガイドブック」が発行されている。このガイドブックでは、電子政府システムを 5 つの形態に分けて、この形態の中で利用されている暗号技術の概要、および調達における仕様策定において、暗号技術をどのように考慮すべきかが述べられている。しかし、システム調達の際に必要となる製品やシステムの暗号に関する標準を考慮しつつ暗号を選択するようになっていない。

本ドキュメント、電子政府推奨暗号リストガイド（以下リストガイド）は、暗号調達のためのガイドブックで記述されていない上記の問題を解決することを目的としている。。このリストガイドは、現在広く利用されているセキュリティ技術について、その概要、その安全性と暗号アルゴリズムの安全性の関係を示した上で、標準技術の中で選択することが望ましい暗号アルゴリズムとそのセキュリティパラメータを示したものである。

本リストガイドは、電子政府の調達のために検討されているが、同様のセキュリティ基準を求める、金融分野などの他の情報システムにおいても参照することができる。

## 1.2. 想定する読者と利用方法

リストガイドの想定読者は、電子政府システムの調達者、およびシステム構築を行うベンダである。

システム調達者は、電子政府システムの調達を行う際に、その調達仕様の中にセキュリティに関する要件を盛り込む。調達に際し、システムベンダは提案書類の中に、暗号技術を用いたセキュリティ技術の利用を盛り込む。この提案されたセキュリティ技術が、本当

に安全であるかどうかを確認する際に、本リストガイドを参照する。
一方、システムベンダは、調達の際の安全性のガイドラインとして、本リストガイドの情報を参照して仕様の策定、および設計を行うことで、調達要件に沿った安全なシステム構築を容易に行うことができる。

## 1.3. リストガイドの構成

本リストガイドは、電子政府システムで利用することが想定される、標準的なセキュリティ技術について、

・技術の概要
・想定される脅威
・脅威に対する対策方針
・技術が備えるべき要件
・標準化動向
・技術の安全性と、暗号アルゴリズムの安全性の関係
・推奨される利用方法（暗号アルゴリズム、セキュリティパラメータ）

を記述している。記述対象のセキュリティ技術としては、

・認証技術
・PKI 関連技術
・通信路の暗号化技術
・蓄積データの暗号化技術
・改ざん検知、時刻認証技術
・鍵管理
・MAC、KDF

を対象としている。その中で、ISO、ITU-T、NIST などの標準化機関で定められた標準技術について記述を行っている。

## 1.4. 暗号技術に関する選択基準

本リストガイドにおいて､推奨される利用方法を選ぶ際の選択基準は以下の通りである。

・基本的な考え方として、使ってはいけない暗号技術を除外することを目標とする。
・標準規格の中に定められている暗号アルゴリズムの中に、電子政府推奨暗号リストに含まれる暗号アルゴリズムがある場合、当該アルゴリズムを推奨とする。
・標準規格の中で、特に暗号アルゴリズムが定められていない場合には、電子政府推奨暗号リストの暗号アルゴリズムを適用する。
・電子政府推奨暗号リストで注釈がついているアルゴリズムについては、標準規格の中で他に選択肢がない場合を除いては、リストガイド内では推奨しない。
・電子政府推奨暗号リストにない暗号技術（MAC など）については、標準で規定されている技術などで問題があるかどうかを確認する。
・セキュリティパラメータは、該当する利用方法に必要な保証期間を念頭に、過去の CRYPTREC レポートでの監視結果に基づいて選択する。
・公開鍵暗号の鍵長については 2048 ビット以上、ブロック暗号のブロック長は 128 ビット以上を基本を基本とし、規格の制約や実装上の制約がある場合には、必要な有効期間に応じて、注釈つきで別途短い鍵長について記載をする。また、使い方によっては短い鍵長やブロック長でも問題がないため、その旨の注釈をつける。
・DSA については、2048 ビットの仕様が策定されつつあるが、ドラフト版であるため、参考として掲載している。仕様と評価が固まり次第、推奨とする。
・パディングの方式が複数定義されている暗号技術の場合、過去の CRYPTREC REPORT での安全性評価に従い、最も安全な方式について推奨とする。

## 1.5. リストガイドの利用方法

1.2 にある通り、本リストガイドは、電子政府システムの調達者、および調達の際に仕様策定を行うベンダが参照することを想定している。具体的には、以下の方法で参照することが典型例として考えられる。

### (1) システム形態に対応した利用方法

本リストガイドでは、セキュリティを検討する観点で、電子政府システムを以下の 3 つのシステムに分類した上で推奨を検討した。この分類は、「内閣官房情報セキュリティセンタ（NISC）政府機関統一基準適用個別マニュアル DM6-07 情報システムの構築等におけるセキュリティ要件およびセキュリティ機能による分類」に基づいたものである。

・ Web システム

Web サーバを用いた情報提供を行うシステムである。また、このシステムでは、提供する情報を外部委託された保守業者が行うことを想定しており、保守業務に関するセキュリティも考慮される。このタイプのシステムでは、Web を介した情報提供における相手認証、

相手認証のベースとなる PKI、情報自体の改ざん防止、および Web サーバの認証に用いる鍵の管理が求められる。

このため、このようなシステムの調達においては、2 章.通信相手の認証、3 章 PKI、6 章.改ざん検知・時刻保証、7 章.鍵管理を参照する必要があり、その他の要件に応じて他の章も参照する。

・ インターネットサービスシステム

クライアント-サーバモデルのシステムを用いて、インターネット経由で、コンテンツ閲覧、電子申請、届け出、メールマガジンの登録や発行などのサービスを行うシステムである。WWW の技術が用いられることがあるが、WWW 以外の技術を用いて構成することも可能で、広くインターネットを利用したサービスが対象となる。このタイプのシステムでは、サーバの認証、クライアントの認証、相手認証のベースとなる PKI、通信路や蓄積データの暗号化、情報自体の改ざん防止、および鍵の管理が求められる。

このため、このようなシステムの調達においては、2 章.通信相手の認証、3 章.通信路の暗号化、4 章.蓄積データの暗号化、5 章.改ざん検知・時刻保証、6 章 PKI、7 章.鍵管理を参照する必要があり、その他の要件に応じて他の章も参照する。

・ 複合機システム

イントラネットの中で、印刷されるデータ、スキャンするデータなど、様々なデータ、ファイルを蓄積するシステムである。このタイプのシステムでは、ユーザ認証、蓄積データの暗号化、および鍵管理を行う必要がある。

このため、このようなシステムの調達においては、2 章.通信相手の認証（特に IC カード)、5 章.蓄積データの暗号化、7 章.鍵管理を参照する必要があり、その他の要件に応じて他の章も参照する。

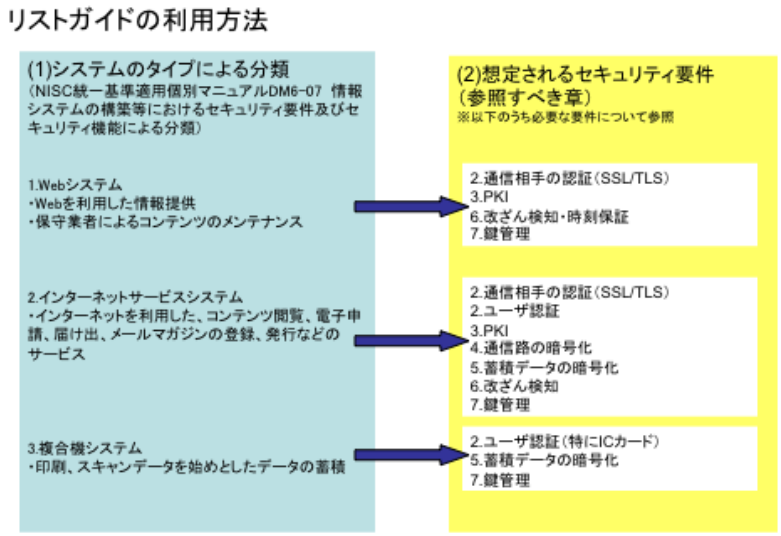



図 1.1 システム形態に対応した利用方法

以上の利用方法を図 1.1 に示す。

本リストガイド自体は、上記以外のシステムにおいても適用可能であり、標準的なセキュリティ技術を用いたシステムにおいて参照可能である。

### (2) 調達作業に対応した利用方法

本リストガイドは、各対象技術について、技術の概要、想定される脅威と対策方針、技術のセキュリティ要件、標準化動向、技術の安全性の根拠、および推奨される利用方法を記述している。これらの記述は、全てを参照する必要はなく、システム調達のタイミングに応じて読み分けることができる。典型的な利用方法は以下の通りである。

・　　調達システムの基本検討

基本検討においては、調達するシステムに必要なセキュリティ要件と対策技術を導出するために、リスク分析を実施する。リスク分析を実施するにあたり、システム構成とシステムで想定する脅威に応じたセキュリティ技術を選ぶ必要がある。この作業を行う際には、各技術に関する記述の中で、「技術の概要」、「想定される脅威」、「脅威に対する対策方針」を参照する。

・　　調達仕様の検討、仕様書の策定

この作業では、セキュリティ要件の具体化と、仕様書の策定を行う。仕様書の策定においては、標準に沿った技術の選定と、正しい暗号の利用を仕様として指定する必要がある。

この作業を行う際には、各技術に関する記述の中で「標準化動向」、「推奨される利用方法」を参照する。

・　　調達先の決定

この作業では、調達システムに適合する提案を行った業者を選定する。この作業を行うにあたり、「標準化動向」、「推奨される利用方法」を参照し、判断を行う。また、これらの推奨に沿わない提案が行われた場合、「技術の安全性と暗号アルゴリズムの安全性の関係」を参照し、ベンダに対して提案内容が安全であることを示すように指示を行う。

・　　納品受領～検収

システムの納品にあたり、設計書等の開発に関わるドキュメントの中で、仕様書において提示した推奨される技術とその利用方法が正しく実装されていることをチェックする。その際に、「推奨される利用方法」を参照する。

## 1.6. リストガイドの維持管理について

暗号技術の進展、および暗号技術に対する攻撃の進展に伴い、本リストガイドの維持管理を以下のように行う。

・　　年に１回を目処に、定期的に安全性に関する新たな研究成果を盛り込んだうえで、推奨される利用方法についての見直しを行う。

・　　標準的なセキュリティ技術、暗号アルゴリズムに対する攻撃が発生し、緊急の安全性に対する問題が発生した場合には、可能な限り修正情報を公開するとともに、リストガイドに反映の上、改版されたリストガイドを公開する。

# 2. エンティティ認証

## 2.1. 通信相手の認証

### 2.1.1. 実現するセキュリティ機能

本対策技術では，ネットワークに接続されている 2 者間（A と B とする）の通信において、通信相手が意図した正しい通信相手であることを確認する機能を実現する。通信相手の認証技術においては、A にとって通信相手が意図している B である場合には、正しい通信相手であることが確認できるとともに、意図している B で無い場合には、正しい通信相手ではないことが確認できる必要がある。

通信相手の認証技術には、一方向認証と相互認証が存在する。一方向認証は、A が通信相手 B を認証するが、B は A のことを認証しない形態である。相互認証は A が B を認証し、同時に B が A を認証する形態である。A と B は一方向認証、相互認証のどちらにおいても何らかの秘密情報を事前に持っておく必要がある。その秘密情報の保持方法と証明方法により認証技術の違いが生じる。

A と B の 2 者間の認証において，信頼できる第 3 者が認証にかかわる形態もある。A が通信相手 B を認証する際、第 3 者が正しく B であることを保障するものである。

### 2.1.2. 想定される脅威

通信相手の認証技術で想定される脅威の一つに盗聴がある．攻撃者Cは、任意のA,Bの 2 者間の通信を盗聴した上で、この技術を利用する任意のエンティティと通信することを想定する。ただし、AとBの 2 者が秘密鍵などの秘密情報を保持しており、この秘密情報を認証に用いる場合、2.2.4(2)で示される鍵管理技術により秘密は保持されており、秘密情報の漏洩は想定しないものとする。

想定される脅威の一つとして成りすましがある。認証時に使用する検査関数やハッシュ関数、乱数が危殆化している場合、攻撃者 C は A と B のやり取りを盗聴している間に、何らかの方法で秘密情報そのものや認証情報の推測が出来る場合がある。この場合，A は正しい通信相手 B ではない C を正しい通信相手として認証してしまうことが可能になる。

中間者攻撃（Man-In-The-Middle Attack）も想定される脅威の一つとなる。攻撃者 C は A と B の認証中に、その通信経路をのっとることができるとする。A と B がやり取りしている電文が固定している場合、攻撃者 C は A と B の認証経路に割り込み、メッセージを中継することにより、気づかれることなく盗聴したり通信内容に介入したりすることが可能になる場合がある。

### 2.1.3. 対策方針

#### (1) 必須対策方針

ネットワークに接続された 2 者間の通信においては、攻撃者が通信路の盗聴を行い、お

よび任意のエンティティと通信することを想定するため、通信データにデジタル署名やメッセージ認証子を付加することにより、改ざん防止を行うことが必要である。

また、盗聴したデータを再利用する中間者攻撃への対抗として、個々のセッションに乱数などのセッション毎に値が異なるデータを含める必要がある。

認証技術はAとBが秘密情報を持っていることを証明することがその技術の根幹となっている。その証明方法は秘密情報そのものを通信経路上に公開しない方式である必要がある。

認証技術に検査関数を使用している場合、検査関数の出力長を十分大きくする必要がある。

### (2) 推奨対策方針

ネットワークに接続された２者間の通信においては、通信路の盗聴および任意のエンティティの通信が可能であるため，経路の暗号化をすることを考慮されたい。

また、証明書を使用した認証を行う場合、信頼の起点となる機関が発行する証明書を使用することを考慮されたい。

秘密情報を格納する場合、HSM[1]など耐タンパー性のある電子機器に保存することを考慮されたい。

[1] HSM: Hardware Secyrity Module : 暗号化やデジタル署名などの処理に伴う鍵管理を強化するために、鍵を耐タンパー性のあるデバイスで管理した上で処理を行うハードウエア

### 2.1.4. 通信相手の認証で利用される対策技術

#### (1) SSL

#### (ア) 技術概要

SSL とは公開鍵証明書を用いて認証を行い、TCP レベルで暗号化を行う通信プロトコルの一種である。Web サーバと Web ブラウザ間の一方向認証／相互認証および通信の暗号化を行うことができ、一般的な Web ブラウザにも標準で搭載されているため、インターネットショッピング等を実現するシステムでも広く利用されている。SSL は公開鍵暗号、共通鍵暗号、ハッシュ関数、署名および擬似乱数生成系の応用技術を用いて構成される。

ここでは、クライアントとサーバの間の認証を想定する。相互認証の場合、クライアント、サーバ双方が通信相手を認証し、一方向認証の場合はクライアントがサーバを認証することとする。相互認証の場合は、双方が検証可能な公開鍵証明書をあらかじめ発行されているものとする。一方向認証の場合、サーバがあらかじめ公開鍵証明書の発行を受けているものとする。

SSL 通信は以下の手順で実行される.

1. クライアント上で使用できる暗号アルゴリズムの一覧をサーバに送付する
2. サーバが暗号通信に使用する暗号アルゴリズムを決定する
3. 決定した暗号アルゴリズムをクライアントに送付する
4. サーバ証明書をクライアントに送付する
5. クライアントが、保持しているルート CA の CA 証明書を使用して、サーバ証明書を検証する
6. 擬似乱数生成系を使用し、Premaster-Secret を生成する
7. クライアントが、サーバ証明書の公開鍵を使用して、Premaster-Secret を暗号化する
8. 暗号化した Premaster-Secret をサーバに送付する
9. クライアントが、Premaster-Secret からセッション鍵を生成する
10. サーバが、クライアントから送付された Premaster-Secret を、サーバの秘密鍵を使用して復号する
11. サーバが、Premaster-Secret からセッション鍵を生成する
12. クライアントが、サーバに対して終了通知を送付する
13. サーバ～クライアント間で、セッション鍵を使用した暗号通信を開始する

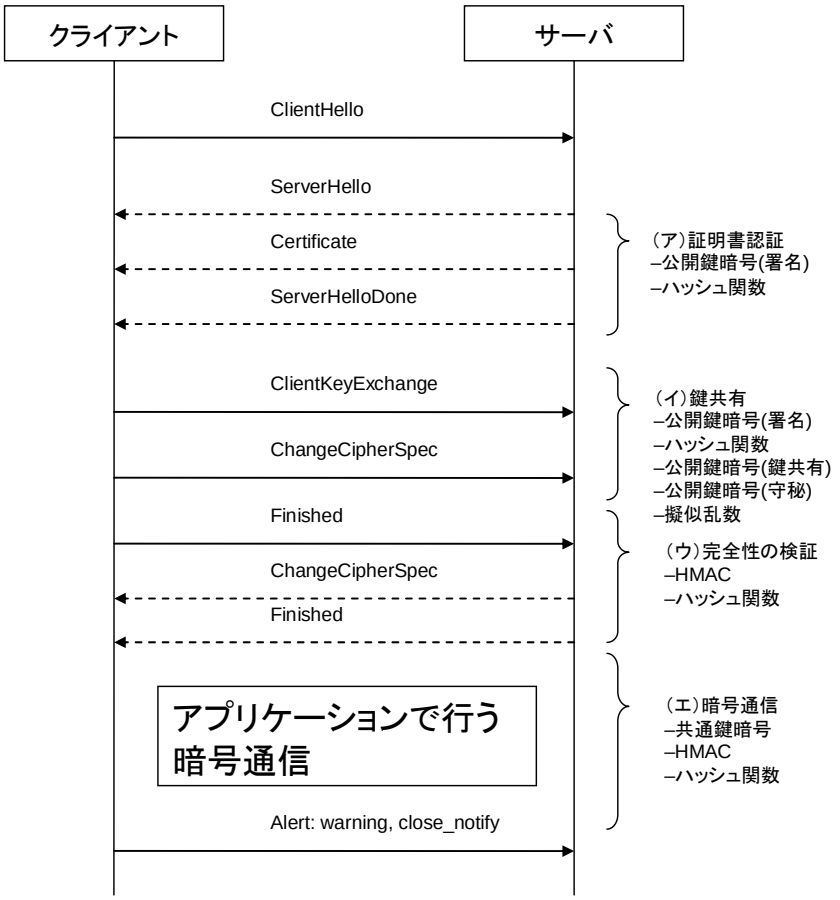



図 1　SSL のシーケンス例(サーバ認証)

SSL 通信上の、暗号利用方法は以下のとおり。

### (i)証明書認証

SSL 通信開始時にサーバ認証をおこなう。上記手順の 5.において行っているのがサーバ認証である。クライアントが保持しているルート CA の証明書を使用してサーバから送られてきたサーバ証明書チェーンをチェックし、サーバが信頼できるものかを確認する。SSL 通信において、サーバ認証は必ず行われる。

クライアント認証も実施する場合には、別途、クライアントからサーバに証明書が送付され、認証がおこなわれる。

### (ii)鍵共有

サーバ認証のほかに、SSL 通信開始時に行うことは鍵共有である。

クライアントでセッション鍵のもととなる Premaster-Secret を擬似乱数生成系で作成し、サーバの公開鍵で暗号化しサーバに送付する。サーバはクライアントから送られてきた Premaster-Secret を復号し、そこからセッション鍵を生成する。

鍵共有では、正しい通信相手と鍵共有をしていることを確認するため、送受信と合わせて署名生成／検証による認証を実施する。

なお、セッション鍵とは SSL 通信時のセッションで使用される共通鍵のことである。

**(iii)完全性の検証**

ハンドシェイクの最後に、サーバ～クライアント間の通信が改ざんされていなかったことを確認するため、ハンドシェイク中に受け取ったデータを HMAC でダイジェスト化してお互いに送信する。受信側は自らが保存していたハンドシェイクメッセージを同じ鍵でダイジェスト化し、送信側のダイジェストと一致するかどうかを確認する。

**(iv)暗号通信**

鍵共有で生成したセッション鍵を用いて電文を暗号化しながら、サーバ～クライアント間で、暗号通信をおこなう。暗号化通信時にもメッセージが改ざんされていないかどうか、完全性の検証を行う。

### （イ）備えるべき条件

SSL は、PKI 環境が存在し、エンティティが公開鍵／秘密鍵のペアと公開鍵証明書を所有していることが運用の前提である。また、サーバとクライアントが双方で同じバージョンの SSL をサポートし、少なくとも一つの同じ暗号アルゴリズム集合をサポートしている必要がある。

### （ウ）標準化動向

SSL の最新版は、IETF によって TLS1.1 として RFC4346 において標準化されている。

### （エ）暗号アルゴリズムの安全性と SSL の安全性の関係

**(i)証明書認証**

証明書認証では、署名の検証、証明書の検証に、公開鍵暗号（署名）およびハッシュ関数を利用する。

公開鍵証明書は、比較的長期間（1 年から 3 年が一般的）有効期限を持つため、この有効期間の間に偽造が行われない安全性を持つデジタル署名アルゴリズム、および鍵長を選択する必要がある。デジタル署名アルゴリズムのベースとなる公開鍵暗号アルゴリズムまたはハッシュ関数に脆弱性が発生した場合、公開鍵証明書を偽造される可能性があるため、意図しないサーバまたはクライアントを正当なものと誤って認証する可能性がある。

**(ii)鍵共有**

鍵共有では、公開鍵暗号（鍵共有）と公開鍵暗号（署名）または、公開鍵暗号（守秘）

と公開鍵暗号（署名）を利用する。また Premaster-Secret の生成時に擬似乱数生成系を使用する。

鍵共有と合わせて認証に利用されている公開鍵は公開鍵証明書内に含まれるため、デジタル署名と同程度の期間使用される。このため鍵長の選択は、証明書に対する基準と同等の基準で行う。

鍵共有に利用されている公開鍵暗号（鍵共有）／公開鍵暗号（守秘）アルゴリズムが危殆化した場合は、Premaster-Secret が漏洩し暗号通信内容を盗聴されてしまう可能性がある。

擬似乱数生成系には

- 同一の値が生成されないよう生成される値に十分な長さがあること
- 生成される値に偏りがないこと
- 生成される値が予測できないこと

が要求される。

乱数として同一の値が生成される場合や生成される値が予測できる場合には、過去の電文を記録している悪意ある第三者が、同一の乱数による過去の電文を用いて、正当なユーザになりすますなど、攻撃の余地を与えることになる。

### (iii)完全性の検証

完全性の検証では、HMAC を利用する。

完全性の検証に利用されている HMAC、あるいはそのベースとなっているハッシュ関数に脆弱性が発生した場合、何らかの方法で通信が改ざんされた場合に改ざんを検知できない可能性がある。

### (iv)暗号通信

暗号通信には共通鍵暗号と、(iii)と同様に完全性の検証のため HMAC およびハッシュ関数を利用する。

暗号通信に利用されている共通鍵はセッションごとに更新される。通常の HTTP 通信では、セッションは永続的ではないが、VPN などセッションを保持する場合は、定期的に共通鍵を更新する必要がある。

暗号通信に利用されている共通鍵暗号アルゴリズムに脆弱性が発生した場合、通信内容を盗聴されてしまう可能性がある。

完全性の検証に利用されている HMAC については、(iii)同様。

## （オ）推奨される利用方法

### (i)証明書認証

公開鍵暗号（署名）:

- RSA（RSASSA-PKCS1-v1_5） 2048bit 以上
- DSA 2048bit 以上

- ECDSA 224bit 以上

ハッシュ関数：SHA-1 ※

**(ii)鍵共有**

公開鍵暗号（署名）、ハッシュ関数：(i)と同様

公開鍵暗号（守秘）：

- RSA（RSAES-PKCS1-v1_5） 2048bit 以上 ※

公開鍵暗号（鍵共有）：

- DH 2048bit 以上
- ECDH 192bit 以上

擬似乱数生成系：SSL においては独自に擬似乱数生成系が規定されている。

**(iii)完全性の検証**

HMAC：HMAC-SHA-1

**(iv)暗号通信**

HMAC：(iii)同様

共通鍵暗号：

- AES 128bit 以上
- Camellia 128bit 以上

※ 利用は推奨されないが、RFC4346 の規定上他のアルゴリズムは選択できない。RFC4346 の規定が変更されるなどで、推奨されるアルゴリズムに変更できるようになった場合は、暗号の切り替え等を検討することが推奨される。

### (2) SSH

#### (ア) 技術概要

SSH (Secure SHell)は、クライアント/サーバのアーキテクチャを使用して 2 者間で安全な接続を確立するためのプロトコルである。SSH は Unix で使用されていた rsh (Remote SHell)と同じ操作を公開された経路を通しても安全に行えるようにすることを目的として開発されている。

SSH はリモートからログインするユーザに対して、強力な暗号方式による安全な経路、サーバ・ユーザ認証によるなり済まし防止、メッセージの改ざん防止を行った通信を提供する。またポートフォワードを利用した簡易的な VPN 機能を提供することも出来る。

現在 SSH には 2 つのバージョンが存在する。SSH Version 1 (SSH-1)はプロトコル上の脆弱性がいくつか存在するが、SSH Version 2 (SSH-2)ではそれらの脆弱性は修正されている。ここでは現在主流である SSH-2 で使用される認証方式と暗号アルゴリズムを解説する。

SSH-2 は 3 つの認証方式を標準でサポートしている。

- 公開鍵による認証
- パスワードベース認証
- ホストベース認証

ここでは暗号アルゴリズムと認証の安全性に注目するため、主に公開鍵による認証について触れる。

SSH-2 の認証シーケンスを以下に示す。

14. ホスト公開鍵をサーバからクライアントに送信する
15. 初めて接続するサーバであれば、ホスト公開鍵を保存する。2 回目以降の接続であれば、登録済みの鍵と照合し、サーバのなりすましが行われていないかを確認する
16. クライアント・サーバ双方で鍵共有用の公開鍵・秘密鍵を作成する
17. 鍵共有の公開鍵をお互いに送信して、鍵共有の準備をする
18. 鍵共有を行ない、共通鍵を作成する
19. 公開鍵暗号(署名)を利用したチャレンジ＆レスポンスにより、クライアント認証を行う

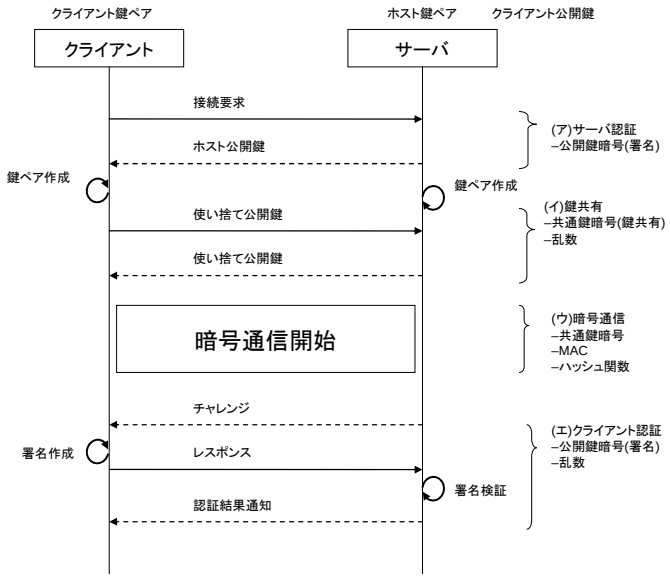



図 2 SSH-2 のシーケンス

暗号利用方法は以下のとおり。

### (i)サーバ認証

SSH クライアントはホスト公開鍵の検証によりサーバ認証を行っている。サーバ認証の方式は SSH プロトコルで規定されていないが、PKI を用いた証明書の検証を行うこともできる。多くの実装ではこれまでに接続したサーバ名と公開鍵を記憶しておき、現在の接続先が成りすましていないか確認を行う”known hosts”ファイルによる検証を行っている。

### (ii)鍵共有

通信の暗号化に使用する共通鍵のために、サーバとクライアント間で公開鍵暗号(鍵共有)を使用し、鍵共有を行う。ここで使用する公開鍵は毎回作成して破棄するため、使い捨て(Ephemeral)公開鍵とも呼ばれる。

### (iii)暗号通信

鍵共有で生成したセッション鍵を用いて電文を暗号化しながら、サーバ～クライアント間で、暗号通信をおこなう。同時にメッセージが改ざんされていないかどうか、完全性の検証を行う。

### (iv)ユーザ認証

ユーザ認証には 3 つの方式がある。

- 公開鍵による認証：サーバに登録してある公開鍵とペアとなる秘密鍵を持っていることをユーザが証明することで認証を行う方式である。ここでは主にこの認証

方式について述べる。

- パスワード認証：クライアントがユーザ名とパスワードを送信し、ホストがその二つを検証して認証を行う方式である。
- ホストベース認証:サーバが特定のホストからのみ接続を認める認証方式である。この認証はホストが厳密に管理されており、シビアなセキュリティが必要な環境か、バッチジョブのような特定の目的のために用いられる場合などに限るべきである。

### （イ）備えるべき条件

SSH は同じバージョンの SSH を使用するサーバとクライアントが存在し、事前にサーバの公開鍵ペア、ユーザの公開鍵ペアを作成し、サーバにユーザの公開鍵を登録しておく必要がある。

### （ウ）標準化動向

SSH-2 は IETF によって標準化が行われ、RFC として公開されている。

- The Secure Shell (SSH) Protocol Assigned Numbers (RFC 4250)
- The Secure Shell (SSH) Protocol Architecture (RFC 4251)
- The Secure Shell (SSH) Authentication Protocol (RFC 4252)
- The Secure Shell (SSH) Transport Layer Protocol (RFC 4253)
- The Secure Shell (SSH) Connection Protocol (RFC 4254)
- Using DNS to Securely Publish Secure Shell (SSH) Key Fingerprints (RFC 4255)
- Generic Message Exchange Authentication For The Secure Shell Protocol (SSH) (RFC 4256)
- The Secure Shell (SSH) Transport Layer Encryption Modes (RFC 4344)

### （エ）暗号アルゴリズムの安全性と SSH の安全性の関係

#### (i)サーバ認証

サーバ認証には公開鍵(署名)を利用する。

サーバ・クライアント認証で使用される公開鍵は、再登録をしない限り永続的に使用される。このため比較的長期の安全性を持つ公開鍵アルゴリズム、および鍵長を選択する必要がある。

#### (ii)鍵共有

鍵共有には公開鍵暗号(鍵共有)および乱数を利用する。

鍵共有に利用されている公開鍵暗号は、認証が行われるたびに生成され、使い捨てされるため鍵長の選択基準は高くなくとも良い。しかし短い鍵長を利用した場合、全数探索に

よって解読されてしまう危険性がある

擬似乱数生成系には

- 同一の値が生成されないよう生成される値に十分な長さがあること
- 生成される値に偏りがないこと
- 生成される値が予測できないこと

が要求される。

乱数として同一の値が生成される場合や生成される値が予測できる場合には、過去の電文を記録している悪意ある第三者が、同一の乱数による過去の電文を用いて、正当なユーザになりすますなど、攻撃の余地を与えることになる。

**(iii)暗号通信**

暗号通信には共通鍵暗号と完全性の検証のため MAC およびハッシュ関数を利用する。

暗号通信に利用される共通鍵はセッションごとに更新される。SSH のセッションは設定にもよるがシェルの使用時間と同程度維持されると考えると、鍵空間の狭い共通鍵暗号を選択すると通信の盗聴を許してしまう危険性がある。完全性の検証に利用されている MAC およびハッシュ関数に脆弱性が発生した場合、何らかの方法で通信が改ざんされた場合に改ざんを検知できない可能性がある。

**(iv)ユーザ認証**

ユーザ認証には公開鍵暗号(署名)と乱数を利用する。

ユーザ認証で使用する公開鍵(署名)の安全性については（i）と同様、乱数については(ii)と同様。

### （オ）推奨される利用方法

**(i)サーバ認証**

公開鍵暗号（署名）:

- RSA（RSASSA-PKCS1-v1_5） 2048bit 以上
- DSA 2048bit 以上

**(ii)鍵共有**

公開鍵暗号（鍵共有）: DH 2048bit 以上

ハッシュ関数：SHA-256

擬似乱数生成系：

- PRNG based on SHA-1 in ANSI X9.42-2001 Annex C.1
- PRNG based on SHA-1 for general purpose in FIPS 186-2 (+ change notice 1) Appendix 3.1
- PRNG based on SHA-1 for general purpose in FIPS 186-2 (+ change notice 1) revised Appendix 3.1

### (iii)暗号通信

共通鍵暗号：AES128bit 以上

MAC：HMAC-SHA-1

### (iv)ユーザ認証

公開鍵暗号(署名)：(i)と同様

擬似乱数生成系：(ii)と同様

## 2.2. ユーザ認証

### 2.2.1. 実現するセキュリティ機能

本対策技術は，システム等において、利用者（ユーザ）がそのシステム等を利用する権限を持っているか確認するなどの目的のために、ユーザを特定し正当な利用者であることを確認する機能を実現する。

ユーザ認証においては、システムにとって相手が正当なユーザである場合には、正しい相手であることが確認できるとともに、正当なユーザで無い場合には、正しい相手ではないことが確認できる必要がある。

ユーザ認証の形態には，利用者が記憶しているパスワードなどの情報による認証，バイオメトリスクによる認証，利用者が所持している IC カード等のデバイスによる認証（機器認証など）がある．

以下，バイオメトリクス以外の認証について扱う。

ユーザ認証では，ユーザは何らかの秘密情報を事前に持っておく必要がある。その秘密情報の保持方法と証明方法により認証技術の違いが生じる。

ユーザとシステムの 2 者間の認証において信頼できる第三者が認証にかかわる形態もある。システムが相手ユーザを認証する際、第三者が正当な利用者であることを保障するものである。

### 2.2.2. 想定される脅威

ユーザ認証技術で想定される脅威の一つに、攻撃者が任意の利用者とシステムとの間のやりとりを盗聴し，利用者とシステムとの間の通信に介入できるとする。また秘密情報を格納したデバイスの紛失時を考慮し、デバイスに対して自由にアクセスを行うことができるとする。

想定される脅威の一つとして成りすましがある。認証時に使用する検査関数やハッシュ関数、乱数が危殆化している場合、攻撃者はシステムとユーザのやり取りを盗聴している間に、何らかの方法で秘密情報そのものや認証情報の推測ができるとする。この場合システムは正しいユーザではない攻撃者を正しい相手として認証してしまうことが可能になる。

中間者攻撃も想定される脅威の一つとなる。攻撃者はシステムとユーザの認証中に、その通信経路をのっとることができるとする。システムとユーザがやり取りしている電文が固定している場合、攻撃者はシステムとユーザの通信経路に割り込み、メッセージを中継することにより、気づかれることなく盗聴したり通信内容に介入したりすることが可能になる場合がある。

秘密情報を保存しているデバイスにバックドアやデバック用メモリダンプ，あるいは実装上の不備などが存在する場合、攻撃者がデバイスを入手したときに，電力解析攻撃や故障解析攻撃も想定される脅威になる。

### 2.2.3. 対策方針

#### （ア）必須対策方針

攻撃者がシステムと利用者との間の通信路を盗聴可能である場合には、通信データにデジタル署名やメッセージ認証子を付加することにより、改ざん防止を行うことが必要である。

また、盗聴したデータを再利用する中間者攻撃への対抗として、個々のセッションの乱数などのセッション毎に値が異なるデータを含める必要がある。

認証技術は，ユーザが秘密情報を持っていることを証明することがその技術の根幹となっている。その証明方法は秘密情報そのものを通信系路上に公開しない方式である必要がある。

認証技術に検査関数を使用している場合、検査関数の出力長を十分大きくする必要がある。

#### （イ）推奨対策方針

通信をしているエンティティ間の盗聴が可能な場合には、経路の暗号化をすることを考慮されたい。

また、証明書を使用した認証を行う場合、信頼の起点となる機関が発行する証明書を使用することを考慮されたい。

秘密情報を格納する場合、暗号機器に対する認証を通過した耐タンパー性のある機器に保存することを考慮されたい。

生体認証など本人固有の情報を元に認証を行うことを考慮されたい。

### 2.2.4. ユーザ認証で利用される対策技術

#### (1) ワンタイムパスワード

#### (ア) 技術概要

ワンタイムパスワードは遠隔地にある端末からネットワークを通じてサーバを利用する際に、アクセスしてくる利用者が正規のユーザかどうかを検証する認証技術の一つである。

ワンタイムパスワードは、パスワードをログイン毎、または数分に設定された短期間で自動的に更新することで，攻撃者がパスワードを推定することを難しくする。

ワンタイムパスワードはサーバとユーザの両方でパスワードを生成する。ユーザ側ではICカードや認証トークンなどの物理デバイスで自動的にパスワードを生成する。

代表的なワンタイムパスワードの実現方式には，チャレンジ&レスポンス方式と時刻同期方式がある.

#### (i)チャレンジ&レスポンス方式

1. ユーザはサーバに対してユーザ名を送信する
2. サーバは端末に対して乱数(チャレンジ)を送信する
3. ユーザはパスフレーズを端末に入力する。端末に備えられたソフトウェアがサーバから送られてきたチャレンジ文字列とユーザが入力したパスフレーズを一定の手順に従って演算する
4. 生成された結果(レスポンス)をサーバに対して送信する
5. サーバは受け取った情報を比較し、認証可否を決定する

上記4の演算部分は方式により異なるが、ハッシュ関数を使用している例が多い。

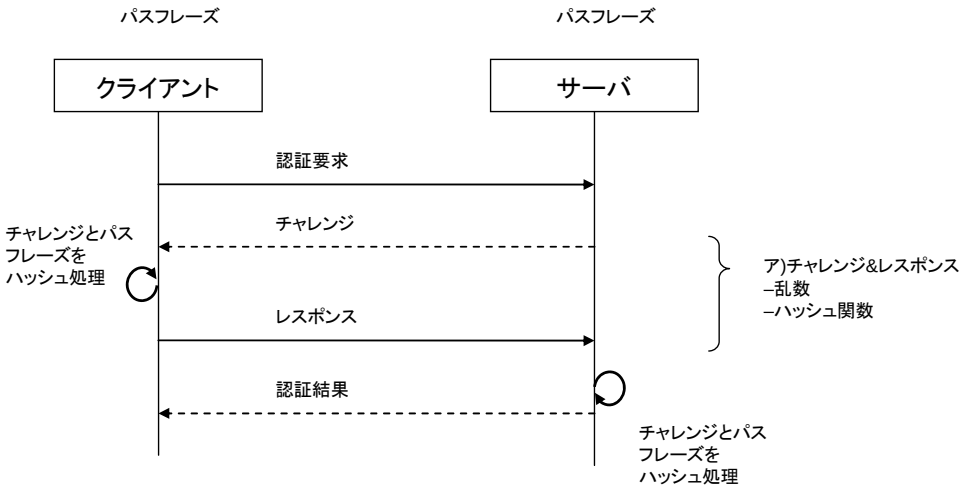



**図 3　チャレンジ&レスポンス方式のシーケンス**

### (ii)時刻同期方式

1. ユーザ側端末とサーバで時刻を同期させる
2. トークンコードと PIN と時刻を組み合わせたものを特定のアルゴリズムで変換する
3. 2.で生成した情報をサーバに送信する
4. サーバ側で 2.と同じ変換を行い、3.で受信したものと比較し認証可否を決定する

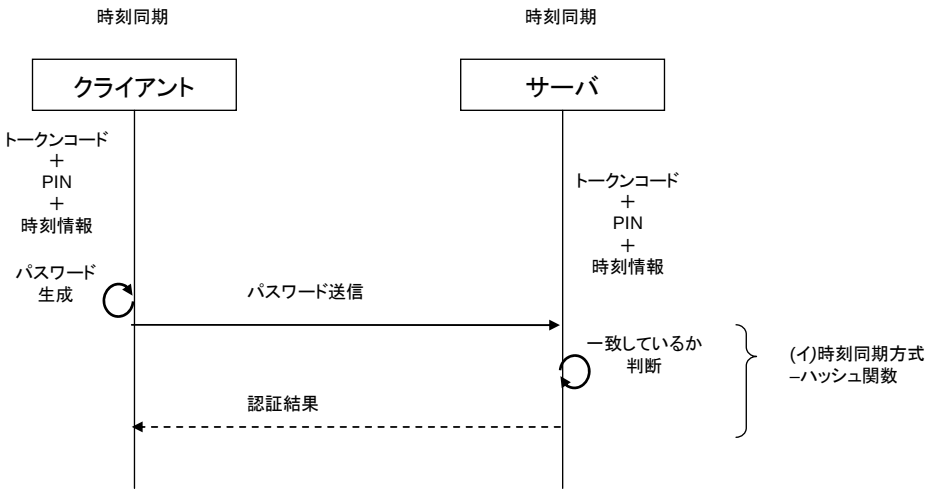



**図 4 時刻同期方式のシーケンス**

ワンタイムパスワードにおける暗号利用方法は以下のとおり。

### (iii)チャレンジ＆レスポンス方式

チャレンジ&レスポンス方式によるワンタイムパスワードはサーバからのチャレンジを元に特定のアルゴリズムによりパスフレーズとチャレンジを変換したものを端末がサーバに送信する。

### (iv)時刻同期方式

時刻同期方式によるワンタイムパスワードは、サーバ・クライアント間で事前に時刻を同期させておく必要がある。クライアントは認証を開始した時点の時刻と、トークンコード、PIN からパスワードを作成し、サーバに送信する。サーバはクライアントと同じ計算を行い、一致するかどうかを判断する。

### （イ）備えるべき条件

ワンタイムパスワードは、エンティティが同じ方式のワンタイムパスワードアルゴリズムを採用し事前にユーザ情報とパスフレーズをサーバに登録している必要がある。 また、時刻同期方式であれば、サーバとクライアントに PIN とパスフレーズを登録した上で、事前に時刻を同期させておく必要がある。

### （ウ）標準化動向

ワンタイムパスワードは特定の企業が開発し、非公開とされている技術が多い。OATH はワンタイムパスワード技術を標準化し自身の Web サイトで公開している。また OATH は IETF でワンタイムパスワード技術を公開している。IETF の keyprov WG でワンタイムパスワード技術の標準化が進行している。

- OATH (Initiative for Open AuTHentication)
  http://www.openauthentication.org/
- The S/KEY One-Time Password System (RFC1760)
- A One-Time Password System (RFC2289)
- HOTP: An HMAC-Based One-Time Password Algorithm (RFC4226)
- IETF keyprov WG http://www.ietf.org/html.charters/keyprov-charter.html

### （エ）暗号アルゴリズムの安全性とワンタイムパスワードの安全性の関係

#### (i)チャレンジ＆レスポンス方式

ワンタイムパスワードが利用している暗号技術はハッシュ関数と乱数である。

ワンタイムパスワードで使用するハッシュ関数は十分なダイジェスト長と衝突耐性、一方向性を持っている必要がある。これらに十分な強度を持たないハッシュ関数では、ハッシュ値から元の秘密情報が推測されることで，成りすましが可能になる危険性がある。

擬似乱数生成系には

- 同一の値が生成されないよう生成される値に十分な長さがあること
- 生成される値に偏りがないこと
- 生成される値が予測できないこと

が要求される。

乱数として同一の値が生成される場合や生成される値が予測できる場合には、過去の電文を記録している悪意ある第三者が、同一の乱数による過去の電文を用いて、正当なユーザになりすますなど、攻撃の余地を与えることになる。

#### (ii)時刻同期方式

時刻同期方式が利用している暗号技術はハッシュ関数である。

ハッシュ関数の安全性については、(i)と同様。

### （オ）推奨される利用方法

#### (i)チャレンジ＆レスポンス方式

ハッシュ関数:SHA-256、 SHA-384、 SHA-512

擬似乱数生成系：

- PRNG based on SHA-1 in ANSI X9.42-2001 Annex C.1、
- PRNG based on SHA-1 for general purpose in FIPS 186-2 (+ change notice 1) Appendix 3.1
- PRNG based on SHA-1 for general purpose in FIPS 186-2 (+ change notice 1) revised Appendix 3.1

#### (ii)時刻同期方式

ハッシュ関数：(i)と同様

### (2) Authenticated Key Exchange

#### （ア）技術概要

Authenticated Key Exchange（認証をともなう鍵共有技術）とは、鍵共有をおこなおうとする相手が正当な相手であるか認証した上で、鍵を交換・共有するための技術である。

認証をともなう鍵共有技術には、複数の方式があり、方式により利用する暗号技術も異なっている。鍵共有を利用する環境で利用できる暗号技術、システムに求められるセキュリティ要件等を勘案し、適切な方式を選択する必要がある。

認証をともなう鍵共有技術における認証では、一方向認証と相互認証の 2 つの認証方法がある。認証を実施するシステムの条件に応じて認証方法を選択する。

- 一方向認証：認証をおこなおうとするユーザのうち、一方のみが他方の正当性を確認する認証方式。逆方向の認証はおこなわない。
- 相互認証：認証をおこなおうとするユーザが、互いに相手の正当性を確認する認証方式。

ユーザ以外に信頼できる第三者機関が認証に関わる場合がある。信頼できる第三者機関とは、セキュリティに関して、ユーザに信頼される機関またはシステムを指す。認証をともなう鍵共有技術で利用される、信頼できる第三者機関は以下の 3 種。

- 認証局（CA）：公開鍵証明書を発行する認証機関。公開鍵暗号技術を用いる方式で利用する場合がある。
- 鍵配布センター（KDC）：鍵を交換しようとするユーザに、鍵を配布する機関。鍵の生成はセンターでおこない、各ユーザ向けに暗号化された状態で配布される。共通鍵暗号技術を用いる方式で利用する場合がある。
- 鍵変換センター（KTC）：鍵を交換しようとするユーザに、鍵を配布する機関。鍵の生成は一方のユーザがおこない、暗号化した状態でセンターに転送する。センターはもう一方のユーザ向けに鍵を暗号化しなおし、配布する。共通鍵暗号技術を用いる方式で利用する場合がある。

以下に、公開鍵暗号と共通鍵暗号を用いる３パス（３回の情報交換）による相互認証を例に、認証をともなう鍵共有手順の例を示す。

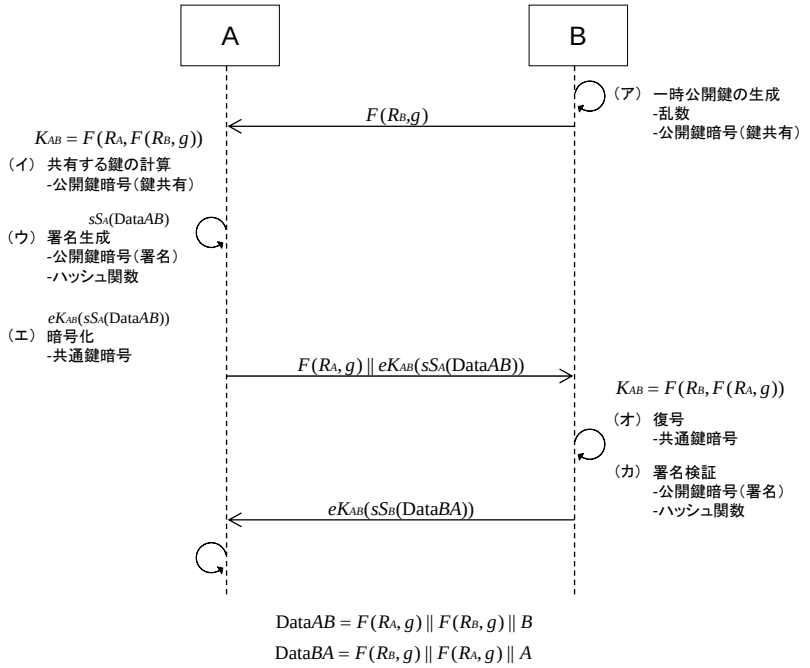



**図 5　3パスによる相互認証（Authenticated Key Exchange）**

ここで使用されている記号は以下のとおり。

A, B　：認証をおこなおうとするユーザ、または、その ID

F(a,g)　：a と g を引数にした、鍵共有関数（公開鍵暗号（鍵共有））F の演算結果の値。二つの値、F(a,g) と F(b,g) について、F(a,F(b,g))=F(b,F(a,g)) という数学上の性質を持つ。なお、F(a,g) と F(b,g) を知っても、a, b を知らなければ、F(a,F(b,g)) または、F(b,F(a,g)) を計算することは困難という性質も持つ

$R_X$　：ユーザ X により生成された乱数。本手順では、鍵共有用の一時的秘密鍵として機能する

g　：鍵共有関数 F について、公開されている要素

$eK_{XY}(Z)$　：データ Z を X と Y 間で共有される共通鍵 $K_{XY}$ で暗号化した暗号文

$sS_X(Z)$　：データ Z からユーザ X の秘密鍵 $S_X$ を用いて生成した署名

DataXY　：X から Y に向けて送信される、認証用データ

||　:データの単純な連結。データ D1 と D2 を連結したデータは D1||D2 と表現される

以下に手順を説明する。

1. B は、乱数 $R_B$ を生成し、$F(R_B,g)$を計算する。乱数 $R_B$ を一時的秘密鍵、$F(R_B,g)$を一時的公開鍵として保持し、$F(R_B,g)$を A に送信する。
2. A は、乱数 $R_A$ を生成し、$R_A$ を一時的秘密鍵、$F(R_A,g)$を一時的公開鍵として保持する。
3. A は、B から受信した $F(R_B,g)$と合わせて、B と共有する共通鍵 $K_{AB}$ ＝$F(R_A,F(R_B,g))$を計算する。
4. A は、DataAB（一時的公開鍵 $F(R_A,g)$，$F(R_B,g)$，送信先の ID である B を結合したデータ）から認証用の秘密鍵 $S_A$ を用いて署名を生成する。
5. A は、4.で生成した署名を、B と共有する共通鍵 $K_{AB}$ で暗号化する。
6. A は、一時的公開鍵 $F(R_A,g)$と 5.で作成した暗号文を B に送信する。
7. B は、一時的秘密鍵 $R_B$ と、A から受信した $F(R_A,g)$を用いて、A と共有する共通鍵 $K_{AB}$＝$F(R_B,F(R_A,g))$を計算する。鍵共有関数 F の性質から、2.で A が演算した $K_{AB}$ と、B が演算した $K_{AB}$ は、同一の値であると仮定できる。
8. B は、$K_{AB}$ を利用して、A から受信したデータの暗号化された部分を復号し、$sS_A$(DataAB)を得る。
9. B は、事前に安全な方法で入手しておいた A の公開鍵と署名の元のデータである DataAB（既に受信済みのデータ、自身で保持しているデータから作成する）を用いて、署名 $sS_A$(DataAB)を検証し、検証が OK になることを確認する。検証が OK になれば、$F(R_A,g)$，$F(R_B,g)$，B が一致しているため、正しく B 自身へのデータであること、過去のデータの再送ではないことが確認できる。また、A が A しか知らない A の秘密鍵 $S_A$ を用いて署名を生成したことが確認でき、A の正当性が確認できる。更に、暗号文の復号が正しくおこなえたため、A と B が保持している共通鍵 $K_{AB}$ が一致しており、鍵共有に成功したことが確認できる。
10. B は、A 同様に、DataBA（$F(R_B,g)$，$F(R_A,g)$，A）から認証用の秘密鍵 $S_B$ を用いて署名を生成し、共通鍵 $K_{AB}$ で暗号化して A に送信する。
11. A は、B 同様に、受信した暗号文を共通鍵 $K_{AB}$ で復号して署名 $sS_B$(DataBA)を得る。
12. A は、B 同様に、8.で得た署名を、事前に安全な方法で入手しておいた B の公開鍵と署名の元のデータを用いて検証し、検証が OK になることを確認、B の正当性と鍵共有の成功を確認する。

なお、上記手順で共通鍵 $K_{AB}$ は A が一時的秘密鍵である乱数 $R_A$ を生成した時点で決定される。ある任意の共通鍵 $K_{AB}$ を演算させるための乱数 $R_A$ を見つけることは困難であるため、容易には任意の共通鍵 $K_{AB}$ は指定できない。しかし演算に時間をかけることで値の一部を意図的に操作できる可能性はあるため、A からのデータ送信までの時間に制限時間を設けるなどの注意が必要である。

認証をともなう鍵共有(公開鍵暗号と共通鍵暗号を用いる 3 パスによる相互認証)では、以下の箇所で暗号技術を利用している。

**(i)一時公開鍵の生成**

毎回変化する鍵共有のための一時的な公開鍵を作成する。

毎回変化する値を用いることで、過去の認証データを保持している悪意ある第三者によるなりすましを防止する。値の生成には、乱数と公開鍵暗号（鍵共有）を用いる。

上記手順では、1.と 2.が相当する。

**(ii)共有する鍵の計算**

鍵を共有する相手の一時的公開鍵と、自身の一時的秘密鍵から、共有する鍵を計算する。鍵の計算には公開鍵鍵暗号（鍵共有）を用いる。

上記手順では、3.と 7.が相当する。

**(iii)署名生成**

自身の正当性を証明するために、自分自身しか知らない認証用の秘密鍵を用いて、署名を生成し送信する。署名生成には、ハッシュ関数と公開鍵暗号（署名）を用いる。

上記手順では、4.と 10.での署名生成が相当する。

**(iv)暗号化**

共通鍵が正しく共有できていることを確認するために、送信するデータを共有した鍵で暗号化する。暗号化には共通鍵暗号を用いる。

上記手順では、5.と 10.での暗号化が相当する。

**(v)復号**

共通鍵が正しく共有できていることを確認するために、受信した暗号文を共有した鍵で復号する。復号には共通鍵暗号を用いる。

上記手順では、8.と 11.での復号が相当する。

**(vi)署名検証**

相手の正当性を確認するために、相手から受信した署名を検証する。署名検証には、ハッシュ関数と公開鍵暗号（署名）を用いる。

上記手順では、9.と 12.が相当する。

### (vii)認証方法の種類

認証をともなう鍵共有技術には、複数の方式があり、方式により利用する暗号技術も異なっている。

以下に、利用する暗号技術毎の方式を紹介する。

#### i. 共通鍵暗号技術を用いる方式

共通鍵暗号を用いる方式では、ユーザの認証と鍵共有を、ユーザ間、または信頼できる第三者機関間と、事前に共有しておいた共通鍵を用いて実現する。

#### ii. 公開鍵暗号技術を用いる方式

公開鍵暗号を用いる方式では、事前に入手しておいた、相手の公開鍵、または公開鍵暗号（鍵共有）の相手の公開パラメータを用いてユーザを認証する。

#### iii. 弱い秘密を用いる方式

ISO において、人間が記憶出来る程度の長さのパスワードを使って認証と鍵共有を行う「弱い秘密を用いる方式」が標準化されている。弱い秘密を用いる方式では、弱い秘密としてパスワードなどを用いる。ユーザ間で事前に共有しておいたパスワード、またはパスワードとパスワードを元に変換処理をおこなったデータを利用してユーザを認証する。後者の方式は、通常クライアント-サーバ型の認証の場合に用いられる。

### （イ）備えるべき条件

### (i)共有されているべき情報

認証をともなう鍵共有技術では、認証をおこなおうとするユーザは、認証する相手の ID を事前に知っていることが前提となる。

また、認証に利用する秘密情報、または、それに対応する公開情報を事前に、入手していなければならない。

#### i. 共通鍵暗号技術を用いる方式

認証用の共通鍵を、ユーザ間、または信頼できる第三者機関間と共有しておく。認証用の共通鍵は、共有すべき相手だけが知っていること。

#### ii. 公開鍵暗号技術を用いる方式

認証しようとする相手の、公開鍵、または公開鍵暗号（鍵共有）の公開パラメータを入手しておく。入手した公開鍵、または公開パラメータは、確実に相手のものであると確認しておく必要がある。公開鍵を公開鍵証明書から入手する場合には、公開鍵の事前入手は必要ないが、公開鍵証明書の発行元である認証局の証明書を入手しておく必要がある。

なお、公開鍵に、対応する秘密鍵、または秘密パラメータは、認証しようとする相手だけが知っていること。

鍵共有も、相手の公開鍵、または公開鍵暗号（鍵共有）の公開パラメータを利用するが、ユーザを認証せず鍵共有のみを実現する場合には、事前の入手は必要ない場合もある。

iii. 弱い秘密を用いる方式

パスワードなどの弱い秘密、またはパスワードとパスワードを元に変換処理をおこなったデータをユーザ間で事前に共有しておく必要がある。

また、認証をおこなう相手と利用する演算方式合意し、それに合わせたパラメータのセットを共有しなければならない。

### (ii)備えるべき機能

各方式で利用する暗号技術を利用できなければならない。利用できる必要がある暗号技術は、認証の方式により異なる。

i. 共通鍵暗号技術を用いる方式

共通鍵による、データの暗号化、復号機能を備えていなければならない。共通鍵と合わせて、乱数、タイムスタンプ、シーケンスナンバーのうちのいくつかも利用する。また、MAC 関数を利用する場合もある。

ii. 公開鍵暗号技術を用いる方式

利用する公開鍵の種類に応じて、公開鍵暗号（署名）とハッシュ関数による署名生成と署名検証、公開鍵暗号（守秘）による暗号化と復号、公開鍵暗号（鍵共有）による演算機能を備えていなければならない。合わせて、乱数、タイムスタンプ、シーケンスナンバーのうちのいくつかも利用する。また、共通鍵暗号、MAC 関数を合わせて利用する場合もある。

iii. 弱い秘密を用いる方式

ハッシュ関数と乱数を利用する。また、認証をおこなう相手と合意した演算方式の演算関数を備えていなければならない。

### （ウ）標準化動向

Authenticated Key Exchange は、以下の ISO 規格として標準化されている。

- 11770-1:1996 Information technology -- Security techniques -- Key management -- Part 1: Framework
- ISO/IEC 11770-2:1996 Information technology -- Security techniques -- Key

management -- Part 2: Mechanisms using symmetric techniques

- ISO/IEC 11770-3:1999 Information technology -- Security techniques -- Key management -- Part 3: Mechanisms using asymmetric techniques
- ISO/IEC 11770-4:2006 Information technology -- Security techniques -- Key management -- Part 4: Mechanisms based on weak secrets

### （エ）暗号アルゴリズムの安全性と、Authenticated Key Exchange の安全性の関係

各方式で、認証用データの元の値に毎回違う値を利用し、認証用データの再利用を防ぐことを目的に乱数が利用される。

同一の認証用の秘密データを利用している期間など、短期間に同一の乱数が利用されると、過去に利用された認証用データが正しいデータとして利用できる。

このため、擬似乱数生成系には

- 同一の値が生成されないよう生成される値に十分な長さがあること
- 生成される値に偏りがないこと
- 生成される値が予測できないこと

が要求される。

乱数として同一の値が生成される場合や生成される値が予測できる場合には、過去の電文を記録している悪意ある第三者が、同一の乱数による過去の電文を用いて、正当なユーザになりすますなど、攻撃の余地を与えることになる。

また、共有する共通鍵の生成にも乱数が利用される。乱数の生成に偏りや予測可能な性質があると、共有した鍵が予測でき、その後の暗号通信を盗聴できる可能性がある。

以下に、各方式で利用する暗号アルゴリズムとの安全性の関係を示す。

#### i. 共通鍵暗号技術を用いる方式

認証用の共通鍵は、一定の期間同一のものを利用することが想定されるため、利用期間に則した安全性を持つ共通鍵暗号アルゴリズム、および鍵長を選択する必要がある。また、一定期間が経過した後に、認証用の共通鍵を更新する仕組みを備える必要もある。

認証用の共通鍵暗号アルゴリズムが危殆化した場合、MAC 関数が危殆化した場合には、意図しないユーザを正当なユーザと誤って認証する可能性がある。また、認証用の共通鍵で共有する鍵を暗号化して送受信するため、共有する鍵が漏洩し、その後の暗号通信が盗聴可能になる可能性もある。

#### ii. 公開鍵暗号技術を用いる方式

公開鍵暗号（署名）、公開鍵暗号（守秘）は、比較的長期間同一のものが利用されるため、

利用期間に則した安全性を持つ公開鍵暗号アルゴリズム、および鍵長、ハッシュ関数を選択する必要がある。また、一定期間が経過した後に、公開鍵を更新する仕組みを備える必要もある。なお、公開鍵暗号（署名）と公開鍵暗号（守秘）に同一の鍵を利用する場合には、利用期間の長い側に合わせて鍵長を選択すること。

公開鍵暗号（鍵共有）は、一定期間同一のものを利用する場合と、毎回鍵を生成し一時的にだけ利用する方法がある。利用期間に則した安全性を持つ公開鍵暗号アルゴリズム、および鍵長を選択する。

デジタル署名による認証をおこなう場合に、公開鍵暗号アルゴリズムまたはハッシュ関数に脆弱性が発生した場合、公開鍵証明書を偽造される可能性があるため、意図しないユーザを正当なユーザと誤って認証する可能性がある。公開鍵証明書を利用しない場合でも、署名の偽造が可能となるため、同様の問題が発生する。

公開鍵暗号（守秘）、公開鍵暗号（鍵共有）、MAC 関数を認証に利用する場合に、公開鍵暗号アルゴリズムまたは MAC 関数に脆弱性が発生した場合にも、意図しないユーザを正当なユーザと誤って認証する可能性がある。

また、各種公開鍵を利用して共有する鍵（または、その元となるデータ）を送受信するため、暗号化に利用する共通鍵が漏洩し、その後の暗号通信が盗聴可能になる可能性もある。

#### iii. 弱い秘密を用いる方式

ハッシュ関数が危殆化した場合、また、パスワードの変換に用いる演算関数が危殆化した場合、意図しないユーザを正当なユーザと誤って認証する可能性がある。

### （オ）推奨される利用方法

#### i. 共通鍵暗号技術を用いる方式

共通鍵暗号：

- AES 128bit 以上
- Camellia 128bit 以上
- CIPHERUNICORN-A 128bit 以上
- Hierocrypt-3 128bit 以上
- SC2000 128bit 以上

※対象となるデータの有効期限によっては、リスト掲載の 64 ビットブロック暗号も対象となる。

MAC：

・HMAC

SHA-1/256/384/512 と利用することを推奨

・CBC-MAC
・EMAC
・OMAC/CMAC
・XCBC-MAC（RFC 3566(AES-XCBC-MAC-96)として）
AES 128bit 以上、Camellia 128bit 以上、CIPHERUNICORN-A 128bit 以上、Hierocrypt-3 128bit 以上、SC2000 128bit 以上、を利用することを推奨

ii. 公開鍵暗号技術を用いる方式

公開鍵暗号（署名）：
- DSA 2048bit 以上
- ECDSA 224bit 以上
- RSA（RSASSA-PKCS1-v1_5） 2048bit 以上
- RSA-PSS 2048bit 以上

公開鍵暗号（守秘）：
- RSA-OAEP 2048bit 以上

公開鍵暗号（鍵共有）：
- DH 2048bit 以上
- ECDH 192bit 以上
- PSEC-KEM 224bit 以上（KEM（Key Encapsulation Mechanism）-DEM(Data Encapsulation Mechanism)構成における利用を前提とする。）

ハッシュ関数：
- SHA-256
- SHA-384
- SHA-512

※ 公開鍵証明書を利用する場合は、利用する公開鍵証明書も上記条件に準ずること

MAC：
・HMAC
SHA-1/256/384/512 と利用することを推奨
・CBC-MAC
・EMAC
・OMAC/CMAC
・XCBC-MAC（RFC 3566(AES-XCBC-MAC-96)として）
AES 128bit 以上、Camellia 128bit 以上、CIPHERUNICORN-A 128bit 以上、Hierocrypt-3 128bit 以上、SC2000 128bit 以上、を利用することを推奨

iii. 弱い秘密を用いる方式

ハッシュ関数：

- SHA-256
- SHA-384
- SHA-512

各演算方式で、演算時のドメインパラメータとして利用する値の桁数として以下を推奨する。

- 離散対数問題に基づく演算方式：2048bit 以上
- 楕円曲線上の離散対数問題に基づく演算方式：192bit 以上

### iv. 各方式共通

擬似乱数生成系（例示）：

- PRNG based on SHA-1 in ANSI X9.42-2001 Annex C.1
- PRNG based on SHA-1 for general purpose in FIPS 186-2 (+ change notice 1) Appendix 3.1
- PRNG based on SHA-1 for general purpose in FIPS 186-2 (+ change notice 1) revised Appendix 3.1

### (3) SASL/APOP

#### (ア) 技術概要

SASL (Simple Authentication and Security Layer)はコネクションベースのプロトコルに認証機能を追加する手法のフレームワークである。

SASL の仕様には

- 認証機能追加のためのコマンドの作り方
- チャレンジ＆レスポンスの概略
- 認証機能の IANA への登録手法
- 認証機能のサンプル

などが記述されている。各コネクションベースのプロトコルへは、これらを参考にして認証機能の追加を行うことになる。APOP は SASL の一派生プロトコルと解釈できる。

APOP は暗号研究者によって以下の観点で安全でないことが示されている。

- MD5 の脆弱性により，中間者攻撃が適用可能な条件がある（詳細は(4)）

SASL は以下の手順で実行される。

1. クライアントがサーバに対して認証要求を送信する
2. サーバは乱数を生成し、チャレンジとしてクライアントに送信する
3. クライアントは秘密情報とチャレンジを組み合わせて、特定の演算をした後、結果をクライアントに送信する
4. サーバは 3 と同じ演算を行い、3.で受け取った情報と比較をする
5. その認証メカニズムが必要とするならば 2.～4.を N 回繰り返す
6. サーバは認証結果をクライアントに送信する

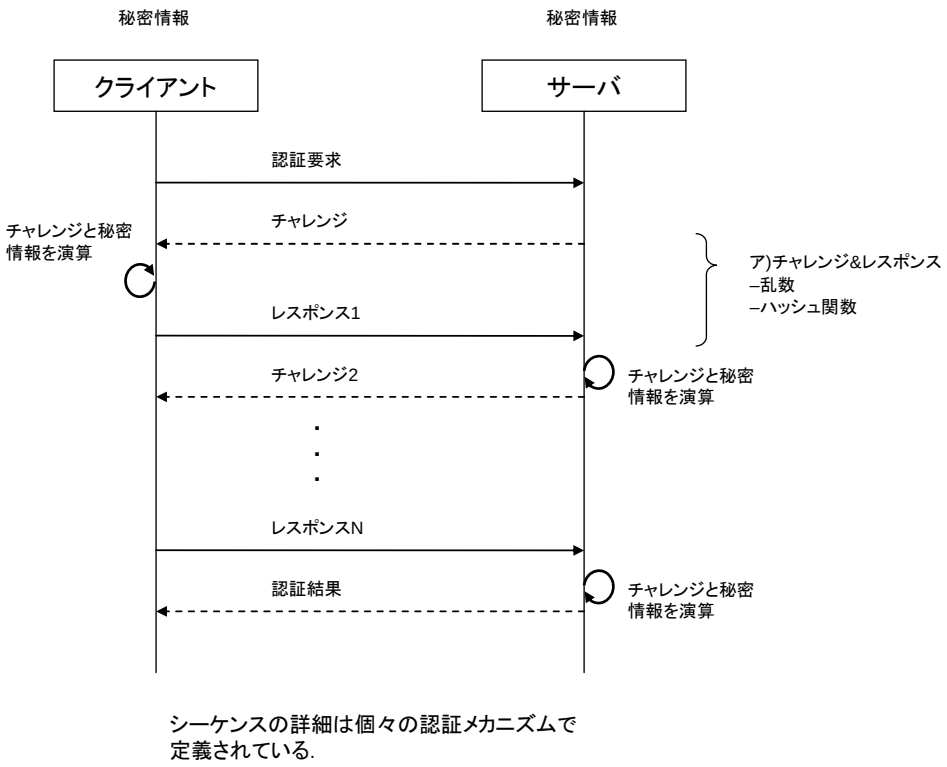



図 6　SASL のシーケンス例

SASL 認証上の、暗号利用方法は以下のとおり。

### (i)チャレンジ&レスポンス

サーバは認証要求を出したクライアントが正当なアクセス権限を持っていることを、パスフレーズなどの秘密情報を持っているかどうかで判断する。サーバは乱数を含むチャレンジをクライアントに送信する。クライアントは秘密情報とチャレンジを組み合わせて所定の演算を行い、その結果をサーバに送信する。サーバは同じ演算を行い、クライアントから受信した結果と比較して、認証の結果を判断する。

APOP の場合、上記手順中で行う演算を以下のように行う。

MD5 (チャレンジ || 秘密情報)

ここで、MD5( ) は MD5 ダイジェスト関数を表し、||は文字連結を表す。

#### （イ）備えるべき条件

SASL は特定のプロトコルに対して追加の認証機能が定義されていることが前提となる。また、クライアントとサーバが共有する秘密情報は外部に漏洩せず、サーバが認証をしよ

うとするクライアントの秘密情報を特定することが可能であることも動作の前提となる。

### （ウ）標準化動向

SASL フレームワーク自体の最新版は RFC2222 として IETF で標準化されている。

Simple Authentication and Security Layer (SASL) (RFC2222)

また SASL を実際にプロトコルに適用した応用も以下のように標準化されている。

The One-Time-Password SASL Mechanism (RFC2444)

Anonymous SASL Mechanism (RFC2245)

Using Digest Authentication as a SASL Mechanism (RFC2831)

### （エ）暗号アルゴリズムの安全性と SASL の安全性の関係

#### (i)チャレンジ&レスポンス

チャレンジ&レスポンスでは、乱数およびハッシュ関数を利用する。

同一の乱数がチャレンジとして利用されると、過去に利用された同一のレスポンス値が正しいレスポンス値として利用できる。

このため、擬似乱数生成系には

- 同一の値が生成されないよう生成される値に十分な長さがあること
- 生成される値に偏りがないこと
- 生成される値が予測できないこと

が要求される。

乱数として同一の値が生成される場合や生成される値が予測できる場合には、過去の電文を記録している悪意ある第三者が、同一の乱数による過去の電文を用いて、正当なユーザになりすますなど、攻撃の余地を与えることになる。

レスポンスの計算に使用するハッシュ関数は十分なダイジェスト長と衝突耐性、一方向性を持っている必要がある。これらに十分な強度を持たないハッシュ関数では、別の値から同じハッシュ値を生成することによるなり済ましを許してしまう危険性がある。

APOP は MD5 の脆弱性を利用した攻撃（偽装メールサーバによる攻撃）が可能であり，一定期間継続的にメールサーバーの成りすましができると，理論的には 61 文字，現実的には 31 文字までのパスワードを特定できることが指摘されている．実験的には 12 文字程度のパスワードならば,500～2000 回程度の試行（偽装したチャレンジへの応答）で特定できるという報告がある※

仕様として MD5 以外を使用することが出来ないので、APOP をメールの認証システムとして採用するのは避けるべきである．

暫定的にはクライアント側で偽装チャレンジのチェック（RFC で定められているチャレンジの書式を厳密にチェックする）ことが提案されている．

※情報処理推進機構「MD 5 の安全性の限界に関する調査研究」報告書, 2007 情財第 1071 号を参照.

### （オ）推奨される利用方法

#### (i)チャレンジ＆レスポンス

擬似乱数生成系：

- PRNG based on SHA-1 in ANSI X9.42-2001 Annex C.1、
- PRNG based on SHA-1 for general purpose in FIPS 186-2 (+ change notice 1) Appendix 3.1
- PRNG based on SHA-1 for general purpose in FIPS 186-2 (+ change notice 1) revised Appendix 3.1

ハッシュ関数：

- SHA-256
- SHA-384
- SHA-512

### (4) 共通鍵暗号技術を用いる方式

#### (ア) 技術概要

共通鍵暗号技術を用いるユーザ認証とは、認証を行うユーザ（またはシステム）間のみで共有されている認証用の共通鍵を用いて、相手を認証する技術である。

共通鍵暗号技術では、暗号化に使用する共有鍵を知っているものだけが、正しく暗号文を復号することができる。

認証しようとしている相手だけが知っているはずの共有鍵を用いて、暗号文による電文の交換をおこない、正しく情報を交換できていれば相手の正当性が保証される。

共通鍵暗号技術を用いるユーザ認証では、一方向認証と相互認証の 2 つの認証方法がある。認証を実施するシステムの条件に応じて認証方法を選択する。

- 一方向認証：認証をおこなおうとするユーザのうち、一方のみが他方の正当性を確認する認証方式。逆方向の認証はおこなわない。
- 相互認証：認証をおこなおうとするユーザが、互いに相手の正当性を確認する認証方式。

また、ユーザ以外に信頼できる第三者機関が認証に関わる場合がある。信頼できる第三者機関とは、セキュリティに関して、ユーザに信頼される機関またはシステムを指す。共通鍵暗号技術を用いるユーザ認証では、事前に双方のユーザとの間で認証用の共有鍵を共有している機関であり、認証をしようとする一方のユーザからの依頼を受けて、他方のユーザだけが復号できるデータを生成する。

以下に、３パス（３回の情報交換）による相互認証を例に、認証手順を説明する。

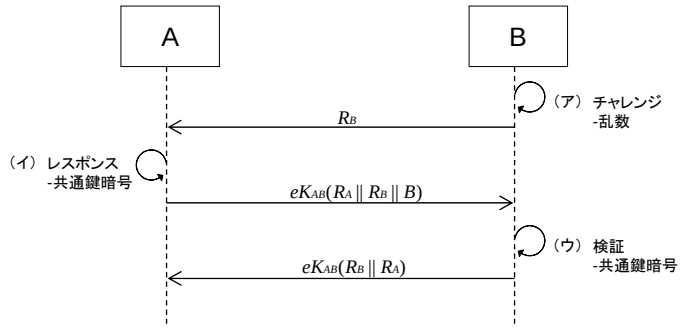



**図 7　3パスによる相互認証（共通鍵）**

ここで使用されている記号は以下のとおり。

A，B　　：認証をおこなおうとするユーザ、または、その ID

$R_X$　　：ユーザ X により生成された乱数

$eK_{XY}(Z)$　：データ Z を X と Y 間で共有されている認証用の共有鍵 $K_{XY}$ で暗号化した暗号文

:データの単純な連結。データ D1 と D2 を連結したデータは D1||D2 と表現される

以下に手順を説明する。

1. B は乱数 $R_B$ を生成し、A に送信する。
2. A は乱数 $R_A$ を生成し、$R_A$、B から受信した乱数 $R_B$、B の ID B を、認証用の共通鍵で暗号化して、認証用の暗号文として B に送信する。
3. B は、受信した暗号文を認証用の共通鍵を用いて復号し、B が自身の ID と一致していることと、自身が A に送信した乱数と $R_B$ が一致していることを確認する。乱数が一致していれば、A が正しい認証用の共通鍵を用いて暗号化したことが確認でき、A の正当性が確認できる。

4. B は、$R_B$、A から受信した乱数 $R_A$ を認証用の共通鍵で暗号化し、認証用の暗号文として A に送信する。
5. A は、受信した暗号文を認証用の共通鍵を用いて復号し、$R_B$、$R_A$ が一致していることを確認する。乱数が一致していれば、B 正しい認証用の共通鍵を用いて暗号化したことを確認し、B の正当性が確認できる。

なお、A から B の暗号文に B の ID を含めているのは、暗号文が B に向けての電文であることを示し、暗号文が A に対する reflection attack（送信されたデータをそのまま当該データの送信者に送りつける攻撃手法）に用いられるのを防ぐためである。

共通鍵暗号技術を用いるユーザ認証（3 パスによる相互認証）は、以下の箇所で暗号技術を利用している。

**(i)チャレンジ**

認証用の暗号文の元データの一部になる値を、認証したい相手に送信する。

毎回変化する値を用いることで、過去の認証データを保持している悪意ある第三者によるなりすましを防止する。値の生成には乱数を利用する。

上記手順では、1の$R_B$と、2で送信するデータに含まれる$R_A$がチャレンジに相当する。

**(ii)レスポンス**

チャレンジを元に、定められた演算の結果をチャレンジの送信元に返信する。

秘密情報を知らなければ正しい演算はおこなえないため、演算結果が正しいことを示すことで自身を認証させる。

共通鍵暗号技術を用いるユーザ認証では、レスポンスの演算に共通鍵暗号を用いる。

上記手順では、2で送信する暗号文、4で送信する暗号文がレスポンスに相当する。

**(iii)検証**

受信したレスポンスの正当性を確認する処理。認証用の暗号文を認証用の共通鍵を用いて復号し、内容を確認する。

正しく復号でき、復号された平文の内容が正しければ、認証用の共通鍵で暗号化されたことが確認できる。

上記手順では、3、5の暗号文の確認が検証に相当する。

### （イ）備えるべき条件

**(i)共有されているべき情報**

認証をおこなおうとするユーザは、認証する相手の ID を事前に知っていることが、前提となる。

事前に認証をおこなうユーザの間で認証用の共通鍵が共有されている、または、ユーザと信頼できる第三者機関との間で認証用の共通鍵が共有されていることが、共通鍵暗号技

術を用いるユーザ認証運用の前提となる。

信頼できる第三者機関を利用する場合、認証をおこなうユーザ双方がその機関を信頼していることが前提となるほか，認証用の共通鍵は、認証をおこなうユーザ間、または、信頼できる第三者機関でのみ共有されていることが前提となる。

#### (ii)備えるべき機能

認証をおこなうユーザは、双方とも認証用の共通鍵による、データの暗号化、復号機能を備えていなければならない。

共通鍵暗号技術を用いるユーザ認証では、乱数、タイムスタンプ、シーケンスナンバーのうちのいくつかを使用する。ある認証用の共有鍵を使い続ける間、これらの値を一意の値で生成できる機能を備えていなければなければならない。

また、タイムスタンプ（時刻証明には用いられない，単に時刻のみを示すデータ)、シーケンスナンバーを利用する場合は、相手から受け取った値の正当性を確認できる機構を備えている必要がある。タイムスタンプの場合は、双方が時計を合わせる機構を持つ必要がある。シーケンスナンバーの場合は、シーケンスナンバーの使用方法について事前に合意をしておき、場合によっては認証対象のユーザ毎に使用済みのシーケンスナンバーを記録する必要もある。

### （ウ）標準化動向

共通鍵暗号技術を用いるユーザ認証は、ISO/IEC 9798-2:1999 Information technology -- Security techniques -- Entity authentication -- Part 2: Mechanisms using symmetric encipherment algorithms として標準化されている。

### （エ）暗号アルゴリズムの安全性と、共通鍵暗号技術を用いるユーザ認証の安全性の関係

#### (i)チャレンジ

チャレンジでは、値の生成に擬似乱数生成系を用いる。

同一の鍵を利用している期間に同一の乱数がチャレンジとして利用されると、過去に利用された同一のレスポンス値が正しいレスポンス値として利用できる。

このため、擬似乱数生成系には

- 同一の値が生成されないよう生成される値に十分な長さがあること
- 生成される値に偏りがないこと
- 生成される値が予測できないこと

が要求される。

乱数として同一の値が生成される場合や生成される値が予測できる場合には、過去の電文を記録している悪意ある第三者が、同一の乱数による過去の電文を用いて、正当なユーザになりすますなど、攻撃の余地を与えることになる。

### (ii)レスポンス、検証

レスポンスとしての暗号文の作成と、レスポンスの検証では、共通鍵暗号を利用する。

認証用の共通鍵は、一定の期間同一のものを利用することが想定されるため、利用期間に則した安全性を持つ共通鍵暗号アルゴリズム､および鍵長を選択する必要がある｡また、一定期間が経過した後に、認証用の共通鍵を更新する仕組みを備える必要もある。

共通鍵暗号アルゴリズムが危殆化した場合、共通鍵暗号技術を用いるユーザ認証を採用しているシステムでは､意図しないユーザを正当なユーザと誤って認証する可能性がある。

## (オ)推奨される利用方法

### (i)チャレンジ

擬似乱数生成系（例示）:

- PRNG based on SHA-1 in ANSI X9.42-2001 Annex C.1
- PRNG based on SHA-1 for general purpose in FIPS 186-2 (+ change notice 1) Appendix 3.1
- PRNG based on SHA-1 for general purpose in FIPS 186-2 (+ change notice 1) revised Appendix 3.1

### (ii)レスポンス、検証

共通鍵暗号:

- AES 128bit 以上
- Camellia 128bit 以上
- CIPHERUNICORN-A 128bit 以上
- Hierocrypt-3 128bit 以上
- SC2000 128bit 以上

※対象となるデータの有効期限によっては、リスト掲載の 64 ビットブロック暗号も対象となる。

### (5) デジタル署名技術を用いる方式

#### （ア）技術概要

デジタル署名技術を用いるユーザ認証とは、持ち主だけが保持する秘密鍵と自分を認証しようとする相手に渡す公開鍵の 2 つの鍵からなる公開鍵暗号の鍵ペアを利用したデジタル署名を用いて、相手を認証する技術である。

デジタル署名技術では、秘密鍵で署名されたデータは公開鍵で検証できる。与えられた署名が認証対象のユーザの公開鍵で正しく検証できれば、認証対象のユーザだけが知っている秘密鍵を用いて署名されたことが確認できる。つまり、署名を作成したのは認証対象のユーザであり、署名の送り主が認証対象のユーザであることが確認でき、正当性が保証される。

デジタル署名技術を用いるユーザ認証では、一方向認証と相互認証の 2 つの認証方法がある。認証を実施するシステムの条件に応じて認証方法を選択する。

- 一方向認証：認証をおこなおうとするユーザのうち、一方のみが他方の正当性を確認する認証方式。逆方向の認証はおこなわない。
- 相互認証：認証をおこなおうとするユーザが、互いに相手の正当性を確認する認証方式。

また、ユーザ以外に信頼できる第三者機関が認証に関わる場合がある。信頼できる第三者機関とは、セキュリティに関して、ユーザに信頼される機関またはシステムを指す。公開鍵暗号技術を用いるユーザ認証では、公開鍵の持ち主を保証する公開鍵証明書を発行する認証局を指す。

以下に、３パス（３回の情報交換）による相互認証を例に、認証手順を説明する。

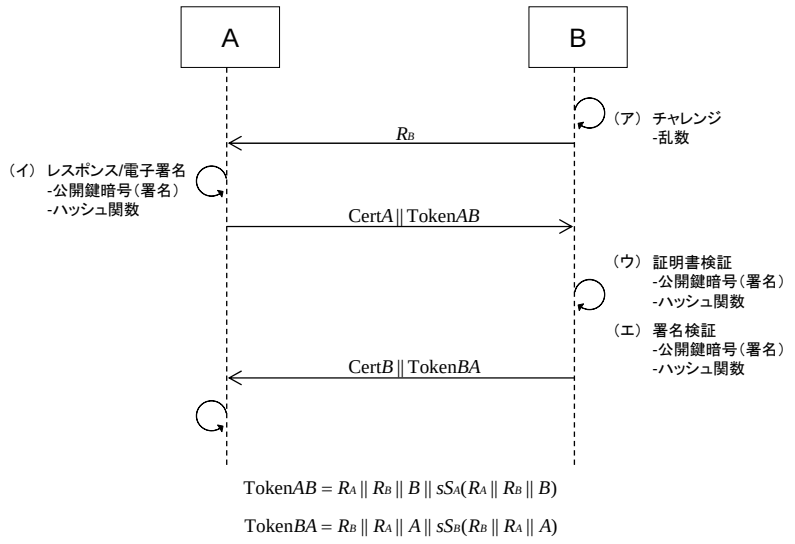



**図 8　3パスによる相互認証（公開鍵）**

ここで使用されている記号は以下のとおり。

| | |
|---|---|
| A，B | ：認証をおこなおうとするユーザ、または、その ID |
| $R_X$ | ：ユーザ X により生成された乱数 |
| CertX | ：認証局から発行されたユーザ X の公開鍵証明書。公開鍵、持ち主の ID、その他の情報が含まれる |
| TokenXY | ：X から Y に向けて送信される、認証用データ |
| $sS_X(Z)$ | ：データ Z からユーザ X の秘密鍵 $S_X$ を用いて生成した署名 |
| \|\| | :データの単純な連結。データ D1 と D2 を連結したデータは D1\|\|D2 と表現される |

以下に手順を説明する。

1. B は乱数 $R_B$ を生成し、A に送信する。
2. A は乱数 $R_A$ を生成し、$R_A$，$R_B$，送信先の ID である B を結合したデータから認証用の秘密鍵を用いて署名を生成する。元のデータと署名からなる TokenAB を作成して B に送信する。公開鍵証明書を用いる場合は、公開鍵証明書 CertA も同時に送信する。
3. 公開鍵証明書を用いる場合、B は、事前に入手しておいた認証局の

証明書を用いて CertA を検証し、証明書の有効性を確認する。証明書が有効であれば、CertA から A の公開鍵を取り出す。公開鍵証明書を用いない場合は、別の安全な方法で A の公開鍵を入手しておく。

4. B は TokenAB の署名を A の公開鍵を用いて検証し、検証が OK になることを確認する。元データの B が自身の ID と一致していること、自身が A に送信した乱数と $R_B$ が一致していることを確認する。乱数が一致していれば、A が A しか知らない A の秘密鍵を用いて署名を生成したことが確認でき、A の正当性が確認できる。
5. B は、A 同様の手順で TokenBA を作成して A に送信する。公開鍵証明書を用いる場合は、公開鍵証明書 CertB も同時に送信する。
6. A は、B 同様に、CertB または別の安全な方法で B の公開鍵を入手する。
7. A は、B 同様に、TokenBA の署名を検証し、A の正当性を確認する。

なお、TokenAB の署名に $R_A$ を含め、TokenBA の署名に $R_B$ を含めているのは、乱数を署名に含めることで、署名の元データを相手に恣意的に選択させないためである。$R_A$，$R_B$ が元データに含まれなければ、相手が送信してくる乱数（$R_B$ または $R_A$）で元データが決定してしまい、相手が恣意的に選択した乱数による選択平文攻撃（暗号解読攻撃上意味のあるデータから署名させ、秘密鍵を推測しようとする攻撃手法）を受ける可能性がある。

デジタル署名を用いるユーザ認証（3 パスによる相互認証）は、以下の箇所で暗号技術を利用している。

**(i)チャレンジ**

デジタル署名の元データの一部になる値を、認証したい相手に送信する。

毎回変化する値を用いることで、過去の認証データを保持している悪意ある第三者によるなりすましを防止する。値の生成には乱数を用いる。

上記手順では、1.の $R_B$ と、2.で送信する TokenAB に含まれる $R_A$ がチャレンジに相当する。

**(ii)レスポンス/署名**

チャレンジを元に、定められた演算の結果をチャレンジの送信元に返信する。

秘密情報を知らなければ正しい演算はおこなえないため、演算結果が正しいことを示すことで自身を認証させる。

デジタル署名技術を用いるユーザ認証では、レスポンスの演算にデジタル署名技術を用いる。デジタル署名技術は、ハッシュ関数と、公開鍵暗号（署名）で実現される。

上記手順では、2.で送信する TokenAB、7.で送信する TokenBA に含まれる署名がレスポンスに相当する。

### (iii)証明書検証

公開鍵証明書の正当性を確認する処理。公開鍵証明書の正当性は、公開鍵証明書に付与された認証局の署名で保証される。認証局の署名は、認証局自身の公開鍵証明書と公開鍵で検証するため、認証局自身の公開鍵証明書を事前に安全な方法で入手しておく必要がある。

証明書検証にはハッシュ関数と、公開鍵暗号（署名）を用いる。

上記手順では、3.の CertA の検証、5.の CertB の検証が証明書検証に相当する。

### (iv)署名検証

受信したレスポンスとしての署名の正当性を確認する処理。署名の元データを相手の公開鍵を用いて検証する。

検証の結果が OK であれば、署名は検証に利用した公開鍵と対をなす秘密鍵で生成されていることが確認できる。

署名検証にはハッシュ関数と、公開鍵暗号（署名）を用いる。

上記手順では、3.、6.が署名検証に相当する。

## （イ）備えるべき条件

### (i)共有されているべき情報

認証するユーザは、認証対象のユーザのものであることが証明されている公開鍵を入手していること。

認証対象のユーザの秘密鍵は、持ち主である認証対象のユーザだけが保持し、使用できること。

公開鍵の入手には公開鍵証明書を用いることもできる。公開鍵証明書を用いる場合は、公開鍵証明書を発行している認証局をユーザ双方が信頼していることが前提となる。また、認証するユーザは、公開鍵証明書を発行している認証局自身の証明書を安全な方法で入手している必要がある。

### (ii)備えるべき機能

認証対象のユーザは、認証に用いる秘密鍵による署名機能を備えていなければならない。相手を認証しようとするユーザは、認証に用いる公開鍵による署名の検証機能を備えていなければならない。相互認証をおこなう場合は、ユーザ双方が署名と検証機能を備えていなければならない。

また、公開鍵証明書を用いる場合には、公開鍵証明書の正当性を検証する機能を備えていなければならない。

デジタル署名を用いるユーザ認証では、乱数、タイムスタンプ、シーケンスナンバーの

うちのどれかを使用する。ある認証用の鍵ペアを使い続ける間、これらの値を一意の値で生成できる機能を備えていなければなければならない。

### （ウ）標準化動向

公開鍵暗号技術を用いるユーザ認証は、ISO/IEC 9798-3:1998 Information technology -- Security techniques -- Entity authentication -- Part 3: Mechanisms using digital signature techniques として標準化されている。

## （エ）暗号アルゴリズムの安全性と、デジタル署名技術を用いるユーザ認証の安全性の関係

### (i)チャレンジ

チャレンジでは、値の生成に擬似乱数生成系を用いる。

同一の鍵を利用している期間に同一の乱数がチャレンジとして利用されると、過去に利用された同一のレスポンス値が正しいレスポンス値として利用できる。

このため、擬似乱数生成系には

- 同一の値が生成されないよう生成される値に十分な長さがあること
- 生成される値に偏りがないこと
- 生成される値が予測できないこと

が要求される。

乱数として同一の値が生成される場合や生成される値が予測できる場合には、過去の電文を記録している悪意ある第三者が、同一の乱数による過去の電文を用いて、正当なユーザになりすますなど、攻撃の余地を与えることになる。

### (ii)レスポンス/署名、証明書検証、署名検証

レスポンスとしての署名、証明書検証、署名検証では、利用する証明書の発行を含め、公開鍵暗号（署名）およびハッシュ関数を利用する。

認証用の公開鍵は比較的長期間同一のものが利用されるため、利用期間に則した安全性を持つ公開鍵暗号アルゴリズム、および鍵長、ハッシュ関数を選択する必要がある。また、一定期間が経過した後に、認証用の公開鍵を更新する仕組みを備える必要もある。

公開鍵暗号アルゴリズムまたはハッシュ関数に脆弱性が発生した場合、公開鍵証明書を偽造される可能性があるため、意図しないユーザを正当なユーザと誤って認証する可能性がある。公開鍵証明書を利用しない場合でも、署名の偽造が可能となるため、同様の問題が発生する。

### （オ）推奨される利用方法

#### (i)チャレンジ

擬似乱数生成系（例示）：

- PRNG based on SHA-1 in ANSI X9.42-2001 Annex C.1
- PRNG based on SHA-1 for general purpose in FIPS 186-2 (+ change notice 1) Appendix 3.1
- PRNG based on SHA-1 for general purpose in FIPS 186-2 (+ change notice 1) revised Appendix 3.1

#### (ii)レスポンス/署名、署名検証、証明書検証

公開鍵暗号（署名）：

- DSA 2048bit 以上，ECDSA 224 以上
- RSA（RSASSA-PKCS1-v1_5、RSA-PSS） 2048bit 以上

ハッシュ関数：

- SHA-256，SHA-384，SHA-512

※ 公開鍵証明書を利用する場合は、利用する公開鍵証明書も上記条件に準ずること

### (6) 検査関数を用いる方式

#### (ア) 技術概要

検査関数を用いるユーザ認証とは、認証を行うユーザ（またはシステム）間でだけ共有されている認証用の共通鍵を用いて、相手を認証する技術である。検査関数として、MAC を利用する。

共有鍵は MAC の入力値となる。同一の共有鍵を入力値にして同一のデータを MAC で変換すると、同一の結果を得ることができる。

この性質を利用して、認証をしようとするユーザ間で変換する元データを共有し、MAC による変換の結果同一の値が得られることで、同一の共有鍵を保持していると確認し、つまりは正当なユーザであると認証する。

検査関数を用いるユーザ認証では、一方向認証と相互認証の 2 つの認証方法がある。認証を実施するシステムの条件に応じて認証方法を選択する。

- 一方向認証

認証をおこなおうとするユーザのうち、一方のみが他方の正当性を確認する認証方式。逆方向の認証はおこなわない。

- 相互認証

認証をおこなおうとするユーザが、互いに相手の正当性を確認する認証方式。

以下に、３パス（３回の情報交換）による相互認証を例に、認証手順を説明する。

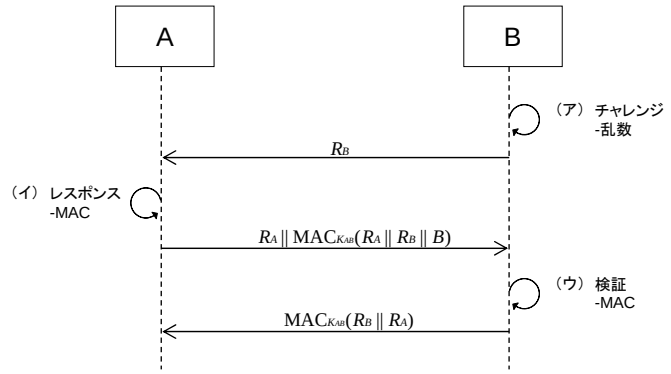



**図 9　３パスによる相互認証（検査関数）**

ここで使用されている記号は以下のとおり。

A，B　　：認証をおこなおうとするユーザ、または、その ID

$R_X$　　：ユーザ X により生成された乱数

$MAC_{KXY}(Z)$：X と Y 間で共有されている認証用の共有鍵 $K_{XY}$ を入力値にデータ Z を MAC で変換した結果

||　　:データの単純な連結。データ D1 と D2 を連結したデータは D1||D2 と表現される

以下に手順を説明する。

1. B は乱数 $R_B$ を生成し、A に送信する。
2. A は乱数 $R_A$ を生成し、$R_A$，$R_B$，送信先の ID である B を結合したデータを認証用の共通鍵を入力値にした MAC で変換する。MAC の変換結果を、$R_A$ と合わせて B に送信する。
3. B は、A から送信されてきた $R_A$ と自身が A に送信した $R_B$、自身の ID B を、認証用の共通鍵を入力値にした MAC で変換する。その変換結果と A から送信されてきた変換結果を比較して、値が一致していることを確認する。MAC での変換結果が一致していれば、A が正しい認証用の共通鍵を入力値に MAC を用いたことが確認でき、A の正当性が確認できる。
4. B は、$R_B$，$R_A$ を結合したデータを認証用の共通鍵を入力値にした MAC で変換し、変換結果を A に送信する。
5. A は、B 同様に、$R_B$，$R_A$ を MAC で変換し、B から送られてきた値を比較して、一致していることを確認する。一致していれば、B の正当性が確認できる。

なお、A から B に送る変換結果の元になるデータに B の ID を含めているのは、変換結果が B に向けての電文であることを示し、変換結果が A に対する reflection attack（受信したデータをそのまま送信者に送りつける攻撃手法）に用いられるのを防ぐためである。

MAC を用いるユーザ認証（3 パスによる相互認証）は、以下の箇所で暗号技術を利用している。

### (i)チャレンジ

毎回変化する MAC への入力値を、認証したい相手に送信する。

毎回変化する値を用いることで、過去の認証データを保持している悪意ある第三者によるなりすましを防止する。値の生成には乱数を利用する。

上記手順では、1の$R_B$と、2で送信するデータに含まれる$R_A$がチャレンジに相当する。

### (ii)レスポンス

チャレンジを元に、定められた演算の結果をチャレンジの送信元に返信する。

秘密情報を知らなければ正しい演算はおこなえないため、演算結果が正しいことを示すことで自身を認証させる。

検査関数を用いるユーザ認証では、レスポンスの演算に MAC を用いる。

上記手順では、2で送信するデータに含まれるMACでの変換結果、4で送信するデータに含まれるMACでの変換結果がレスポンスに相当する。

### (iii)検証

受信したレスポンスの正当性を確認する処理。

検査関数を用いるユーザ認証では、レスポンスの検証に MAC を用いる。

受信した MAC での変換結果と、自身で実行した変換結果が一致すれば、正しい認証用の共通鍵を入力値にした MAC で変換されたことが確認できる。

上記手順では、3、5の変換結果の確認が検証に相当する。

## （イ）備えるべき条件

### (i)共有されているべき情報

認証をおこなおうとするユーザは、認証する相手の ID を事前に知っていることが、前提となる。

事前に認証をおこなうユーザの間で認証用の共通鍵が共有されていることが、共通鍵暗号技術を用いるユーザ認証運用の前提となる。

共有されている認証用の共通鍵は、認証をおこなうユーザ間、または、双方が信頼している第三者でのみ共有されていること。

### (ii)備えるべき機能

認証をおこなうユーザは、双方とも MAC を利用できなければならない。

検査関数を用いるユーザ認証では、乱数、タイムスタンプ、シーケンスナンバーのうちのどれかを使用する。ある認証用の共有鍵を使い続ける間、これらの値を一意の値で生成できる機能を備えていなければなければならない。

## （ウ）標準化動向

検査関数を用いるユーザ認証は、ISO/IEC 9798-4:1999 Information technology -- Security techniques -- Entity authentication -- Part 4: Mechanisms using a

cryptographic check function として標準化されている。

### （エ）暗号アルゴリズムの安全性と、検査関数を用いるユーザ認証の安全性の関係

#### (i)チャレンジ

チャレンジでは、値の生成に擬似乱数生成系を用いる。

同一の鍵を利用している期間に同一の乱数がチャレンジとして利用されると、過去に利用された同一のレスポンス値が正しいレスポンス値として利用できる。

このため、擬似乱数生成系には

- 同一の値が生成されないよう生成される値に十分な長さがあること
- 生成される値に偏りがないこと
- 生成される値が予測できないこと

が要求される。

乱数として同一の値が生成される場合や生成される値が予測できる場合には、過去の電文を記録している悪意ある第三者が、同一の乱数による過去の電文を用いて、正当なユーザになりすますなど、攻撃の余地を与えることになる。

#### (ii)レスポンス、検証

レスポンスとしての変換結果の作成と、レスポンスの検証では、MAC を用いる。

認証用の共通鍵は、一定の期間同一のものを利用することが想定されるため、利用期間に則した安全性を持つ MAC、および共通鍵の鍵長を選択する必要がある。また、一定期間が経過した後に、認証用の共通鍵を更新する仕組みを備える必要もある。

MAC が危殆化した場合、MAC を用いるユーザ認証を採用しているシステムでは、意図しないユーザを正当なユーザと誤って認証する可能性がある。

### （オ）推奨される利用方法

#### (i)チャレンジ

擬似乱数生成系（例示）:

- PRNG based on SHA-1 in ANSI X9.42-2001 Annex C.1
- PRNG based on SHA-1 for general purpose in FIPS 186-2 (+ change notice 1) Appendix 3.1
- PRNG based on SHA-1 for general purpose in FIPS 186-2 (+ change notice 1) revised Appendix 3.1

#### (ii)レスポンス、検証

MAC :

・HMAC

SHA-1/256/384/512 と利用することを推奨

・CBC-MAC
・EMAC
・OMAC/CMAC
・XCBC-MAC（RFC 3566(AES-XCBC-MAC-96)として）
AES 128bit 以上、Camellia 128bit 以上、CIPHERUNICORN-A 128bit 以上、Hierocrypt-3 128bit 以上、SC2000 128bit 以上、を利用することを推奨

### (7) ゼロ知識を用いる方式

#### (ア) 技術概要

ゼロ知識を用いるユーザ認証とは、認証されるユーザが保持している秘密の情報を、認証しようとする相手に開示することなく、保持していることを証明することで認証する技術である。

相手を認証したいユーザは、認証されるユーザに対し、秘密情報を保持していれば正しく答えられる、「質問」に相当するデータをチャレンジとして送信する。認証されるユーザは、受け取ったデータを秘密情報で演算し、「回答」に相当するレスポンスとして送信する。相手を認証したいユーザは、送信した「質問」と受信した「回答」の整合性が取れているか検証し、整合性が取れていれば相手を正当なユーザとして認証する。

以下に、公開鍵暗号（守秘）を用いる方式によるゼロ知識証明を例に、認証手順を説明する。

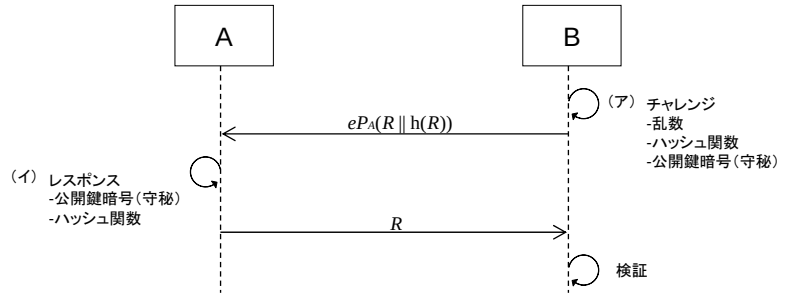



**図 10　公開鍵暗号（守秘）によるゼロ知識証明**

ここで使用されている記号は以下のとおり。

| | |
|---|---|
| A，B | ：認証をおこなおうとするユーザ。この例では、A が認証されるユーザ、B が A を認証しようとするユーザとなる |
| $eP_X(Z)$ | ：ユーザ X の公開鍵 $P_X$ を用いてデータ Z を暗号化した暗号文 |
| R | ：乱数 |
| h(Z) | ：ハッシュ関数 h によりデータ Z を変換した結果。Z のハッシュ値 |
| \|\| | :データの単純な連結。データ D1 と D2 を連結したデータは D1\|\|D2 |

と表現される

以下に手順を説明する。

1. B は乱数 R を生成する。生成した乱数をハッシュ関数で変換し、ハッシュ値 h(R)を作成する。
2. B は、R と R のハッシュ値を A の公開鍵 $P_A$ で暗号化し、A に送信する。
3. A は、B から受信した暗号文を、自身の秘密鍵で復号し、R と R のハッシュ値を得る。
4. A は、R をハッシュ関数で変換し、R のハッシュ値を作成する。自身が作成した R のハッシュ値と B から受信した暗号文に含まれていたハッシュ値を比較し、値が一致していることを確認、R が正しく受信できたこと、暗号文が正しく復号できたことを確認する。
5. A は、B に R を送信する。
6. B は、A から受信した R が、自身が生成した乱数 R と一致することを確認する。一致していれば、A は A しか知りえない、A の秘密鍵を使って R を得たことが確認でき、A の正当性が確認できる。

例とした方式では、R はただの乱数（平文）であるため、A が適当な値を送信して来て、たまたま値が一致している可能性もある。しかし、乱数に適切な長さがあれば、正当なユーザと認識しても問題が無い程度に、たまたま値が一致した可能性を小さくできる。例えば、乱数が 160bit の長さであれば、たまたま値が一致した確率は $2^{-160}$ であり、無視できるほど小さい。

公開鍵暗号（守秘）を利用する方式によるゼロ知識証明では、以下の箇所で暗号技術を利用している。

**(i)チャレンジ**

毎回変化する、「質問」を、認証したい相手に送信する。

「質問」の元データに、毎回変化する値を用いることで、過去の応答を保持している悪意ある第三者によるなりすましを防止する。値の生成には乱数を利用する。上記手順では、1で生成したRが毎回変化する乱数。

受信した相手が、受信した値が正しいことを確認できるよう、ハッシュ関数を利用して、送信する値のハッシュ値を作成する。上記手順では、1でRのハッシュ値を作成している。

認証しようとしている正当なユーザだけが、データを受信できるように、送信するデータを公開鍵で暗号化する。上記手順では、2でRとRのハッシュ値を暗号化している。

### (ii)レスポンス

チャレンジを元に、定められた演算の結果を送信元に返信する。

秘密情報を知らなければ正しい演算はおこなえないため、演算結果が正しいことを示すことで自身を認証させる。

公開鍵暗号（守秘）を用いる方式によるゼロ知識証明では、レスポンスの演算に公開鍵暗号（守秘）を用いる。

上記手順では、レスポンス値を得るために、3で受信した暗号文を公開鍵暗号（守秘）で復号している。

また、受信した値の確認のために、ハッシュ関数も、4で利用している。

## （イ）備えるべき条件

### (i)保持すべき情報

認証をおこなおうとするユーザは、認証する相手の ID を事前に知っていることが、前提となる。

認証をおこなう相手と利用する演算方式を合意し、それに合わせたパラメータのセットを共有しなければならない。

また、認証しようとするユーザは、認証されるユーザの、公開パラメータを安全に入手しておく必要もある。

認証されるユーザは、利用する演算方式に応じた、秘密情報を保持していなければならない。

公開鍵暗号（守秘）を利用する方式の場合は、公開鍵が公開パラメータに、秘密鍵が秘密情報に相当する。

### (ii)備えるべき機能

認証をするユーザ双方が、利用する演算方式の演算ができなければならない。また、ハッシュ関数と、乱数の生成機能を供えていなければならない。

公開鍵暗号を利用する方式の場合は、公開鍵暗号による暗号化、秘密鍵による復号が演算にあたる。

## （ウ）標準化動向

ゼロ知識を用いるユーザ認証は、ISO/IEC 9798-5:2004 Information technology -- Security techniques -- Entity authentication -- Part 5: Mechanisms using zero-knowledge techniques として標準化されている。

### （エ）暗号アルゴリズムの安全性と、ゼロ知識を用いるユーザ認証の安全性の関係

#### (i)チャレンジ

チャレンジでは、値の生成に擬似乱数生成系とハッシュ関数を用いる。また、利用する方式に応じた演算も行い、公開鍵暗号を利用する方式では、公開鍵暗号（守秘）を利用する。

同一の鍵を利用している期間に同一の乱数がチャレンジとして利用されると、過去に利用された同一のレスポンス値が正しいレスポンス値として利用できる。

このため、擬似乱数生成系には

- 同一の値が生成されないよう生成される値に十分な長さがあること
- 生成される値に偏りがないこと
- 生成される値が予測できないこと

が要求される。

乱数として同一の値が生成される場合や生成される値が予測できる場合には、過去の電文を記録している悪意ある第三者が、同一の乱数による過去の電文を用いて、正当なユーザになりすますなど、攻撃の余地を与えることになる。

利用する演算方式やハッシュ関数が危殆化した場合は、秘密情報を知らないにもかかわらず、正しいレスポンス値が作成できてしまう可能性があるため、意図しないユーザを正当なユーザと誤って認証する可能性がある。

公開鍵暗号（守秘）を用いる方式では、公開鍵暗号アルゴリズムが、演算方式に該当する。

#### (ii)レスポンス

レスポンス値は、利用する方式に応じた演算、ハッシュ関数を利用して計算する。演算方式またはハッシュ関数に脆弱性が発生した場合、秘密情報を知らないにもかかわらず、正しいレスポンス値が作成できてしまう可能性があるため、意図しないユーザを正当なユーザと誤って認証する可能性がある。

公開鍵暗号（守秘）を用いる方式では、公開鍵暗号アルゴリズムが、演算方式に該当する。

### （オ）推奨される利用方法

#### (i)チャレンジ

擬似乱数生成系（例示）:

- PRNG based on SHA-1 in ANSI X9.42-2001 Annex C.1
- PRNG based on SHA-1 for general purpose in FIPS 186-2 (+ change notice 1) Appendix 3.1

- PRNG based on SHA-1 for general purpose in FIPS 186-2 (+ change notice 1) revised Appendix 3.1

ハッシュ関数：

- SHA-256
- SHA-384
- SHA-512

公開鍵暗号（守秘）：

- RSA（RSA-OAEP） 2048bit 以上

公開鍵暗号（守秘）を用いる方式以外の、各演算方式で、演算時のパラメータとして利用する公開鍵の鍵長として以下を推奨する。

- 大きい数の素因数分解問題に基づく演算方式：2048bit 以上
- 離散対数問題に基づく演算方式：2048bit 以上
- 合成数による離散対数問題に基づく演算方式：2048bit 以上

### (ii)レスポンス

(i)に同じ

### (8) 手動データ転送を用いる方式

#### （ア）技術概要

手動データ転送を用いる認証とは、2 つの機器間で、人間（機器の管理者）による手動のデータ入力を利用して、相手を認証する技術である。

手動データ転送を用いる認証では、認証用のデータの作成には、検査関数や MAC 関数を用い、その入力値となるデータの入力や、認証結果の確認および確認結果の入力を人間が手動でおこなう。

手動データ転送を用いる認証では、共有しようとするデータが正しいこと、データを共有しようとしている相手（機器）が正当な相手であることが確認できる。

人間が認証に介在することで、通信経路を経由させずに認証の元となる秘密データをやり取りするため、事前に秘密情報を共有する必要は無い。このため、他の方式で要求される事前の安全な情報共有を本方式で実現することも可能である。

以下に、乱数の手動入力による認証を例に、認証手順を説明する。

なお、認証をおこなおうとする機器は、情報の入出力（ディスプレイと入力キー）機能を有している前提である。

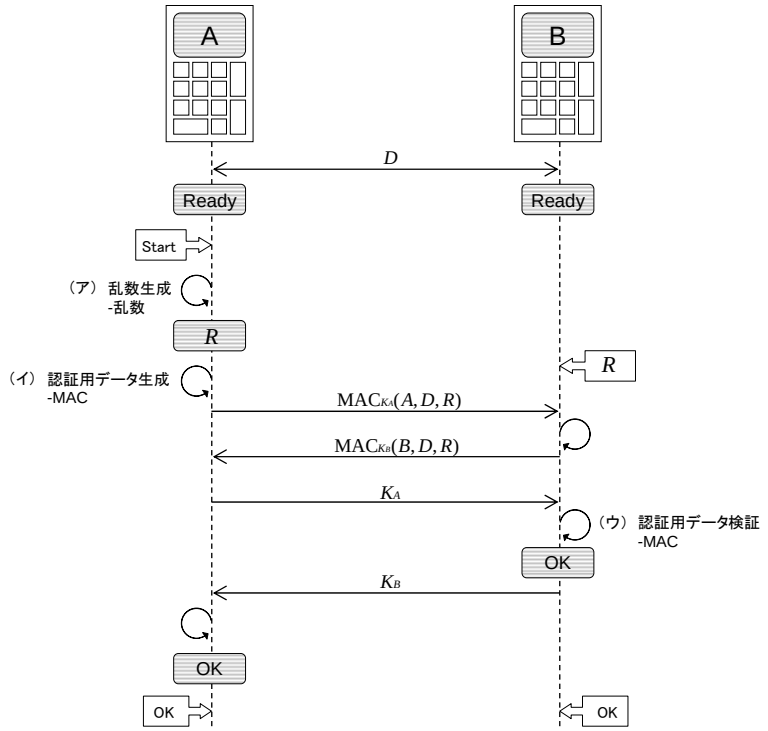



図 11　乱数の手動入力による認証

ここで使用されている記号は以下のとおり。

A，B　　：認証をおこなおうとする機器、または、その ID
D　　：A，B 間で共有しようとしているデータ
R　　：乱数
$MAC_{Kx}(Z)$：鍵 $K_X$ を入力値にデータ Z を MAC 関数で変換した認証用データ
$K_X$　　：機器 X が生成した鍵

以下に手順を説明する。

1. A, B は共有しようとするデータ D を共有する。一方が作成し他方に送信、またはは、相互に送受信したデータを結合して作成する。
2. A，B は、D の共有が完了し認証の準備が整ったことを表示する。
3. 機器の管理者は、A に認証の開始命令を入力する。
4. A は、乱数 R を生成し、表示する。
5. 機器の管理者は、A に表示された乱数を B に入力する。
6. A は、鍵 $K_A$ を生成し、自身の ID A、共有しようとしているデータ D、自身で生成した乱数 R を、$K_A$ を利用した MAC で変換し、その結果の認証用データを B に送信する。
7. B は、鍵 $K_B$ を生成し、自身の ID B、共有しようとしているデータ D、管理者により入力された乱数 R を、$K_B$ を利用した MAC で変換し、その結果の認証用データを A に送信する。
8. A は、B からの認証用データの受信を確認してから、鍵 $K_A$ を B に送信する。
9. B は、A の ID A、データ D、乱数 R を、A から受信した鍵 $K_A$ を利用して MAC で変換し、その結果を A から受信した認証用データと比較する。値が一致していれば、データ D、乱数 R が一致していることが確認できるため、データ D が正しく共有されていること、機器の管理者が正しい機器と認識し乱数を生成させた正しい認証相手であることが確認できる。B は認証成功を表示する。
10. B は、鍵 $K_B$ を A に送信する。
11. A は、B の ID B、データ D、乱数 R を、B から受信した鍵 $K_B$ を利用して MAC で変換し、その結果を B から受信した認証用データと比較する。値が一致していれば、データ D が正しく共有されていること、機器の管理者が正しい機器と認識し乱数を入力した正しい認証相手であることが確認できる。A は認証成功を表示する。

12. 機器の管理者は、A，B 双方で認証が成功していることを確認し、A，B 双方に認証完了を入力する。A，B は、データ D を正しく共有されたデータとして保管する。

データ D が、A が生成した A の公開鍵で、A が B に送信して共有したと仮定すると、上記の手順で､B は公開鍵証明書を利用せずに A の公開鍵を安全に入手することができる。

手動データ転送を用いる認証（乱数の手動入力による認証）は、以下の箇所で暗号技術を利用している。

**(i)乱数生成**

手動データ転送で共有する乱数を生成する。値の生成には乱数を利用する。

上記手順では、4の乱数Rの生成が相当する。

**(ii)認証用データ生成**

認証に用いるデータを生成する。認証に用いるデータの生成には、MAC 関数を利用する。

上記手順では、6のAによる認証用データの生成、3のBによる認証用データの生成が相当する。

**(iii)認証用データ検証**

受信した認証用データを検証する。検証には、MAC 関数を利用する。

上記手順では、9のBによる検証、11のAによる検証が相当する。

**（イ）備えるべき条件**

**(i)共有されているべき情報**

認証をおこなおうとする機器は、認証する相手の ID を事前に知っていることが、前提となる。

**(ii)備えるべき機能**

認証をおこなおうとする機器は、情報の入出力機能を有している前提である。入力は最低限、成功, 失敗の入力ができる 1 つか 2 つのボタンが必要である。出力は最低限、成功、失敗の表示ができるランプなどが必要である。標準的には、英数字の表示と数字などのボタンのセットが必要である。手動データ転送を用いる認証方式のうち、搭載されている機能に応じた認証方式を選択して利用する。

認証をおこなおうとする機器間には通信手段が整えられている必要がある。無線 LAN などが考えられる。この通信手段は、暗号化などの保護を必要としない。

認証をおこなう機器は、認証に利用する暗号技術とアルゴリズムを事前に合意し、合意した検査関数または MAC 関数を利用できなければならない。また、鍵を利用する関数を利用する場合には、鍵の生成ができる機能も必要である。

機器の管理者が出力の確認、機器への入力を直接実施できる必要がある。一人で 2 台を同時に管理するか、信頼できる連携手段（電話など）を有した二人の管理者が個々に管理する。

### （ウ）標準化動向

手動データ転送を用いるユーザ認証は、ISO/IEC 9798-6:2005 Information technology -- Security techniques -- Entity authentication -- Part 6: Mechanisms using manual data transfer として標準化されている。

### （エ）暗号アルゴリズムの安全性と、手動データ転送を用いる認証の安全性の関係

#### (i)乱数生成

認証用データの入力値として乱数を用いる。

ただし、認証用の関数で利用する鍵は毎回新たに生成され、乱数がネットワーク上を流れることも無いことから、厳密な乱数である必要はない。

管理者が間違いなく確認・入力できる程度の長さの乱数が要求される。

管理者が乱数を考える方式の場合で、同じ値を乱数として利用していると認証用のデータの再利用によるなりすましが可能になる場合がある。管理者に毎回違う値を考えさせるか、機器側で生成する方式を採用する必要がある。

#### (ii)認証用データ生成

認証用データの生成に、検査関数または MAC 関数を用いる。

検査関数または MAC 関数が危殆化した場合、意図しない機器を正当な機器と誤って認証する可能性がある。

関数内で利用する共通鍵暗号は認証用データから認証用の秘密情報が漏洩しないよう、安全なアルゴリズムと鍵長を利用する必要がある。

関数内で利用する秘密鍵は、毎回新しいものを生成する。生成される鍵に偏りがあり同一の鍵が利用されると、過去に利用された認証用データが正しいデータとして利用できる可能性がある。

このため、鍵生成に利用する擬似乱数生成系には

- 同一の値が生成されないよう生成される値に十分な長さがあること
- 生成される値に偏りがないこと
- 生成される値が予測できないこと

が要求される。

### （オ）推奨される利用方法

#### (i)乱数生成、認証用データ生成

擬似乱数生成系（例示）:

- PRNG based on SHA-1 in ANSI X9.42-2001 Annex C.1
- PRNG based on SHA-1 for general purpose in FIPS 186-2 (+ change notice 1) Appendix 3.1
- PRNG based on SHA-1 for general purpose in FIPS 186-2 (+ change notice 1) revised Appendix 3.1

MAC：

・HMAC

SHA-1/256/384/512 と利用することを推奨

・CBC-MAC

・EMAC

・OMAC/CMAC

・XCBC-MAC（RFC 3566(AES-XCBC-MAC-96)として）

AES 128bit 以上、Camellia 128bit 以上、CIPHERUNICORN-A 128bit 以上、Hierocrypt-3 128bit 以上、SC2000 128bit 以上、を利用することを推奨

### (9) IC カード

#### (ア) 技術概要

IC カード、IC カードリーダと組み合わせて、IC カード内に保管されている鍵情報や公開鍵証明書情報を使用してユーザ認証に利用できるデバイスである。

IC カードを IC チップの機能により分類すると、メモリカードと IC カードに分かれる。メモリカードは秘密情報を保存する記憶領域を持つカードであり、IC カードは CPU などの制御機構をメモリと同時に備えるカードである。ここでは IC カードを対象とする。

また、IC へのアクセス方法により接触型と非接触型の 2 つに分けることができる。接触型は外部との通信に金属端子を使用する。非接触型は電磁誘導を起電力とし、外部との通信に無線を使用する。

IC カードによる認証の例を以下に示す。

1. リーダは IC カードに対して認証要求を出す
2. IC カードはチップ内にある秘密情報を利用して、認証要求に応答する
3. リーダと IC カードは、通信方法を折衝する
4. リーダと IC カードは 3 で折衝した方式に基づいて通信を行う

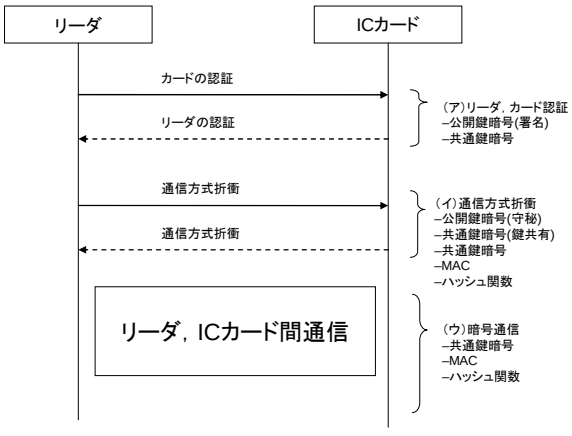



**図 12　IC カードの認証**

IC カードの認証上の、暗号利用方法は以下のとおり。

### (i)リーダ、カード認証

IC カードの通信は最初に、お互いが正しい通信相手であるかの認証を行う。ここでの認証は、チャレンジ&レスポンス方式や、公開鍵暗号(署名)などによって行われる。

### (ii)通信方式折衝

特に非接触カードを使用する場合その通信は傍受可能であるため、盗聴から守る必要がある。IC カードとリーダは後に行う暗号通信で使用する共通鍵や、完全性検証のために使用する鍵の折衝を行う。

### (iii)暗号通信

通信方式折衝で確立した通信方式や鍵で電文を暗号化しながら、リーダ～IC カード間で、暗号通信をおこなう。暗号化通信時にもメッセージが改ざんされていないかどうか、完全性の検証を行う。

## （イ）備えるべき条件

IC カードにおける認証は、IC カード内に保管されている秘密情報が外部に漏洩することがなく、また秘密情報を外部に取り出さずにその秘密情報を保持していることが示すことができる必要がある。

## （ウ）標準化動向

IC カードには国際標準から業界標準まで様々な標準が存在する。

### (i)EMV 仕様(業界標準)

Europay、MasterCard、VISA のクレジットカード会社 3 社が策定したクレジットカード向け IC カードの共通仕様。本仕様は ISO/IEC 7816 に基礎を置いている。

### (ii)ISO/IEC 7816 シリーズ

外部端子付き IC カードの物理的、機能的条件等について、基本部分の互換性を確保するための国際規格。利用分野を特定せずに規格化できる最小部分のみ規定している。

### (iii)JIS X 6319(JICSAP)

JICSAP は IC カードシステム利用促進協議会が制定した IC カード仕様。カード製造者、リーダライタ製造者、システムインテグレータ等の IC カードシステム関係ベンダ間のアプリケーション層での相互の互換性をとることを目的として制定された。JIS X 6319 は JICSAP Ver2.1 を元に 2005 年に制定された

### (iv)全銀協 IC キャッシュカード標準仕様(業界標準)

全国銀行協会が制定した銀行業務向けの多機能 IC カードの規格。

### (v)Card OS 仕様

Java カード、MULTOS カードなどの仕様があり、IC カード上にアプリケーションを搭載することが可能になっている。

### （エ）暗号アルゴリズムの安全性と、IC カードの安全性の関係

#### (i)リーダ、カード認証

リーダ、カード認証では検証に、公開鍵暗号（署名）、ハッシュ関数および乱数を利用する。

公開鍵証明書は、比較的長期間（1 年から 3 年が一般的）有効期限を持ち、またカード内の秘密鍵の更新が比較的困難であるため、有効期間に則した安全性を持つデジタル署名アルゴリズム、および鍵長を選択する必要がある。デジタル署名アルゴリズムのベースとなる公開鍵暗号アルゴリズムまたはハッシュ関数に脆弱性が発生した場合、公開鍵証明書を偽造される可能性があるため、意図しないサーバまたはクライアントを正当なものと誤って認証する可能性がある。

レスポンスの計算に使用しているハッシュ関数に脆弱性が発生した場合、悪意のある第三者の成りすましを防げない可能性がある。

期間に同一の乱数がチャレンジとして利用されると、過去に利用された同一のレスポンス値が正しいレスポンス値として利用できる。

このため、擬似乱数生成系には

- 同一の値が生成されないよう生成される値に十分な長さがあること
- 生成される値に偏りがないこと
- 生成される値が予測できないこと

が要求される。

乱数として同一の値が生成される場合や生成される値が予測できる場合には、過去の電文を記録している悪意ある第三者が、同一の乱数による過去の電文を用いて、正当なユーザになりすますなど、攻撃の余地を与えることになる。

#### (ii)通信方式折衝

通信方式折衝では、公開鍵暗号（鍵共有）、共通鍵暗号、MAC および乱数を利用する。鍵共有の際に、相手認証するための公開鍵暗号（署名）の公開鍵は証明書内に含まれる。この公開鍵の鍵長の選択は証明書による基準に従う。

鍵共有に使用される公開鍵暗号(鍵共有)アルゴリズムが危殆化した場合は、共通鍵が漏洩し暗号通信内容を盗聴されてしまう可能性がある。

完全性の検証に利用されている共通鍵暗号、MAC およびハッシュ関数に脆弱性が発生した場合、何らかの方法で通信が改ざんされた場合に改ざんを検知できない可能性がある。

乱数については,(i)と同様である。

#### (iii)暗号通信

暗号通信には共通鍵暗号と、完全性の検証のため MAC およびハッシュ関数を利用する。

暗号通信に利用されている共通鍵暗号アルゴリズムに脆弱性が発生した場合、通信内容を盗聴されてしまう可能性がある。

MAC およびハッシュ関数については、(ii)と同様である。

### （オ）推奨される利用方法

#### (i)リーダ、IC カード認証

公開鍵暗号（署名）:

- RSA（RSASSA-PKCS1-v1_5、 RSA-PSS） 2048bit 以上
- DSA 2048bit 以上
- ECDSA 224bit 以上

ハッシュ関数：SHA-256、 SHA-384、 SHA-512

#### (ii)通信方式折衝

公開鍵暗号（署名)、ハッシュ関数：(i)と同様

公開鍵暗号（守秘）: RSA（RSA-OAEP） 2048bit 以上

公開鍵暗号（鍵共有）:

- DH 2048bit 以上
- ECDH 192bit 以上
- PSEC-KEM 224bit 以上（KEM（Key Encapsulation Mechanism）-DEM(Data Encapsulation Mechanism)構成における利用を前提とする。）

擬似乱数生成系：

- PRNG based on SHA-1 in ANSI X9.42-2001 Annex C.1
- PRNG based on SHA-1 for general purpose in FIPS 186-2 (+ change notice 1) Appendix 3.1
- PRNG based on SHA-1 for general purpose in FIPS 186-2 (+ change notice 1) revised Appendix 3.1

#### (iii)完全性の検証

MAC：

・HMAC

SHA-1/256/384/512 と利用することを推奨

・CBC-MAC

・EMAC

・OMAC/CMAC

・XCBC-MAC（RFC 3566(AES-XCBC-MAC-96)として）

AES 128bit 以上、Camellia 128bit 以上、CIPHERUNICORN-A 128bit 以上、Hierocrypt-3 128bit 以上、SC2000 128bit 以上、を利用することを推奨

共通鍵暗号：

- AES 128bit 以上
- Camellia 128bit 以上
- CIPHERUNICORN-A 128bit 以上

- Hierocrypt-3 128bit 以上
- SC2000 128bit 以上

※対象となるデータの有効期限によっては、リスト掲載の 64 ビットブロック暗号も対象となる。

### (iv)暗号通信

MAC、ハッシュ関数：(ii)と同様

共通鍵暗号：(iii)と同様

### (10) TCG/TPM

#### (ア) 技術概要

Trusted Computing Group（TCG)とは、信頼できるコンピュータ（ソフトウェア・ハードウェア）環境を実現する標準仕様の開発・普及を行うための業界団体（非営利組織）である。参加企業は半導体ベンダ、PC ベンダ、ソフトウェアベンダ、システム開発ベンダ、ネットワークベンダなどである。

Trusted Platform Module（TPM）とは、TCG が仕様策定しているセキュリティチップで、暗号コプロセッサや不揮発メモリを備え、チップ内部で暗号鍵の生成・格納、暗号処理ができる。TPM は公開鍵暗号（暗号化と署名)、ハッシュ関数、乱数生成などの暗号技術を用いている。

TPM をパーソナルコンピュータ（PC）に搭載することで、例えば暗号通信の鍵を TPM で保管することができる。そして,この暗号通信を行うための仕様は Trusted Network Connect（TNC）Specification で規定している。TPM の応用例としては暗号通信、電子メールやファイルの暗号化などの他、機器認証やプラットフォーム構成の検証が挙げられる。

TPM は，PC などに搭載する際のコストを低減するために TPM の機能は必要最低限に絞られている。TPM 応用を、パーソナルコンピュータのソフトウェアと組み合わせて実現できるようにソフトウェアスタックの仕様が TCG Software Stack (TSS) Specification で定められている。

TPM は以下の基本機能を持つ。

- 鍵生成：公開鍵暗号の鍵ペア生成、共通鍵暗号用の鍵生成を行う。公開鍵暗号として RSA 鍵生成の実装が必須である。
- 暗号化・署名：暗号化/復号、署名生成を行う。RSA アルゴリズムの実装は必須である。共通鍵暗号は TPM 内部利用のみで、Vernam one-time-pad の実装が必須である。

  暗号化（公開鍵暗号、PKCS#1 を参照）

  TPM_ES_RSAESOAEP SHA1 MGF1

  TPM_ES_RSA ESPKCSV15

  暗号利用モード（共通鍵暗号、NIST SP800-38A を参照）

  TPM_ES_SYM_CTR

  TPM_ES_SYM_OFB

  署名（公開鍵暗号，PKCS＃ 1 を参照）

  TPM_SS_RSASSAPKCS1v15_SHA1

  TPM_SS_RSASSAPKCS1v15_DER

TPM_SS_RSASSAPKCS1v15_INFO

- ハッシュ，MAC：
  一方向ハッシュ関数として FIPS180-1 に規定された SHA-1 ハッシュアルゴリズムの実装が必須である。MAC として、RFC2104 に準拠した HMAC のサポートが必須と規定されている。RFC2104 にて K と記載されている鍵サイズは 20 バイト、B と記載されているブロックサイズは 64 バイトと規定されている。
- 乱数生成（RNG）：鍵生成、署名生成、Nonce などで使用する乱数を提供する。TPM の内部利用のみ。真性乱数（物理乱数）か擬似乱数かは任意である。
- 鍵保管：TPM で生成した暗号鍵を安全に保持する。TPM 内部の不揮発性メモリに保持するのは、EK，SRK（後述）の２つであり、残りの暗号鍵は SRK で暗号化して、TPM 外部（ハードディスクなど）に保管する。
- ハッシュ値保管（PCR）：プラットフォームの構成情報として、BIOS，ブートローダ，OS などのハッシュ値を Platform Configuration Registers(PCR)と呼ぶ 160bit のレジスタに保持する。

TPM では以下の鍵（Key）の種類を規定している。

- Endorsement Key：TPM 内部に Endorsement key（EK）と呼ぶ公開鍵暗号の鍵ペア（2048 ビットの RSA 鍵）を１つ保持する。EK は TPM の製造時に生成される TPM チップごとに異なる鍵であり、TPM 自体を識別（認証）するために用いられる。また、EK の利用は TPM の一部の内部機能に制限されている。これによりプライバシーとセキュリティを両立させようとしている。
- Attestation Identify Key：Attestation Identify Key（AIK）はプラットフォームの構成を検証する際に用いられる公開鍵暗号の鍵ペアであり、プラットフォームの構成情報を保持する Platform Configuration Register(PCR)に AIK で署名を付与する。
- Storage Key：Storage Key（SK）はデータを保護するための公開鍵暗号の鍵ペアであり、 Storage Root Key(SRK)を頂点とする親子構造（階層構造）を持つ。SRK は TPM 内部に保管され、その他の SK 及び AIK は、SRK で暗号化されて、TPM 外部に保管される。

TPM では以下の証明書（Credential） の種類を規定している。各 Credential は X.509 や XML で実現される。

- Endorsement or EK credential：EK を生成する際に発行する credential。
- Conformance credential：TPM が規格に準拠していることを示す credential。複

数の Conformance credential を持つことができる。

- Platform credential：プラットフォームメーカにより発行される credential。
- Validation credential：プラットフォームある部分の機能が期待されたとおり機能していることを確認するためにデータを取得した際に得られるべき値を示す credential。
- Identity or AIK credentia：AIK の証明書。プライベート CA によって発行される。

### （イ）備えるべき条件

公開鍵暗号アルゴリズムとして RSA をサポートすることが必須であり、その鍵長は少なくとも 512bit,1024bit,2048bit の 3 種類をサポートし、暗号化と署名に使用する。RSA 鍵生成のための素数判定アルゴリズムは P1363 の規定に従うように規定されている。公開鍵指数 e は 2^16+1 に限る。推奨の RSA 鍵長は 2048bit である。ただし、EK として用いる場合は鍵長 2048bit の RSA であること、SK, SRK, AIK として用いる場合は RSA2048bit と等価もしくはそれ以上の暗号強度があることが必須である。CRT を用いて RSA を実装する場合には、故障利用攻撃への対処を行っていることが必須となっている。

ハッシュ関数として FIPS180-1 の SHA-1 をサポートすることが必須である。MAC 関数として、RFC2104 の HMAC をサポートすることが必須である。他の公開鍵暗号アルゴリズムをサポートすることは TPM の仕様としては許容されている。

### （ウ）標準化動向

TPM は Trusted Computing Group により、“Trusted Platform Module(TPM) Specification”として規定されており、“Part 1 Design Principles”, “Part 2 TPM Structures”, “Part 3 Commands”の 3 部からなる。2007 年 7 月 9 日付で、TCG TPM Specification Version 1.2 Level 2 Revision 103 が発行されている。

### （エ）暗号アルゴリズムの安全性と TPM の安全性の関係

TPM では、暗号化/復号および署名生成/検証として公開鍵暗号（RSA）及びハッシュ関数（SHA-1）および HMAC-SHA-1 を用いている。EK, SRK 以外の暗号鍵は、SRK によって（あるいは他の SK によって）暗号化されて TPM 外部に保管されるため、SRK が漏洩すると EK 以外の全ての暗号鍵の漏洩につながる。AIK credential の X.509 による証明書フォーマットの規定では署名アルゴリズムとして SHA-1 with RSA Encryption と規定されており、この署名鍵の鍵長が短いと TPM（を用いた応用サービス）の安全性に影響を及ぼす可能性がある。

### （オ）推奨される利用方法

**TPM での推奨暗号アルゴリズムおよび鍵長は、以下の通りである。**

公開鍵暗号（守秘）：RSA-OAEP, RSA-PKCS1-v1_5 (2048bit)

公開鍵暗号（署名）：RSASSA-PKCS1-v1_5 (2048bit)

※Credential の署名は 1024bit 以上

ハッシュ関数　　：SHA-1

※TPM の規定上、必須になっている。

MAC 関数　　：HMAC-SHA-1（RFC2104）

# 3. 通信路の暗号化

## 3.1. 拠点間の暗号化

### 3.1.1. 実現するセキュリティ機能

ネットワークに接続されている２者間（A と B とする）の通信において、２者間の通信内容の第三者からの盗聴を防止する機能を実現する。

通信路の暗号化技術には２者間で事前に秘密情報を共有しておき、その秘密情報を用いて暗号通信を行う形態と、鍵交換技術により通信を行う際に２者間で鍵交換を行う形態の技術があるが、最終的には暗号通信を開始する前の段階には２者間で暗号化に使用する鍵が共有される。

また、通信路の暗号化を行う際には、併せて通信相手の認証を行うことも多く、上記鍵交換を認証の過程で行う形態もある。

通信路の暗号化においては、以下がセキュリティ要件としてあげられる。

・　暗号通信を行う２者間での通信が秘匿されており、第三者による盗聴が行えないこと。そのためには以下の要件を満たす必要がある。

　　鍵交換が安全に行われること

　　暗号用の鍵が脆弱でないこと

　　通信を長時間行う場合には暗号用の鍵の更新を考慮すること

### 3.1.2. 想定される脅威

通信路の暗号化において、攻撃者が任意の A,B の２者間の通信を盗聴できるとする。攻撃者は通信を盗聴した上で、暗号用の鍵を盗み出すかもしくは推測することにより、通信内容を盗聴するという脅威が想定される。

以下、想定しうる脅威を挙げる。

・　鍵の漏洩による暗号通信の盗聴

　　事前共有鍵であれば、暗号通信を行う両拠点に同じ鍵を設定する運用の際に鍵の漏洩が起こる可能性がある。

　　通信時に鍵交換を行う場合において、攻撃者 C が２者間の通信を乗っ取ることができるとすると、２者間で第三者が鍵の交換手順に入り込むことによる攻撃（中間者攻撃：Man-in-the-middle attack）により鍵の漏洩が起こる可能性がある。

・　なりすまし

　　攻撃者 C が A および B が接続しているネットワークに接続可能とすると、C は２者（A,B）のどちらかになりすますことにより、暗号通信内容の漏洩が発生する可能性がある。

・　　鍵の推測による暗号通信の盗聴

暗号通信に用いる鍵が脆弱である場合、総当り攻撃などで鍵が推測される可能性がある。

### 3.1.3. 対策方針

#### (1) 必須対策方針

・事前鍵共有であれば安全な手順や運用で鍵の交換が行うことにより、鍵の漏洩を防止する

・通信時に鍵交換を行うのであれば、第三者のなりすましや２者間に第三者が中継を行うことによる攻撃（中間者攻撃：Man-in-the-middle attack）を防止すること。

・全数攻撃などで鍵が推測されてしまわないよう、その時点で充分に安全であることが確認されている暗号アルゴリズムおよび鍵長を利用すること

#### (2) 推奨対策方針

・通信を長期間行う場合に、同じ鍵を使い続けないように通信中に鍵を更新する等の対策を実施すること

### 3.1.4. 通信路の暗号化で利用される対策技術

#### (1) PIN の暗号化

##### （ア）技術概要

PIN(Personal Identification Number)とは認証のためにユーザが保持するコードまたはパスワードのことであり、ユーザ認証目的のために広く利用されている。

セキュリティの観点から PIN は適切に保護して管理する必要がある。

PIN の保護の手段としては物理的に保護する方法及び、暗号化して保護する方法がある。

暗号的な観点から PIN の保護（管理）に必要な観点はたとえば以下のようなものがある。

(a)　　PIN の生成

・　　PIN を乱数により生成する場合、安全な擬似乱数生成系を使用するなど、適切な乱数生成方法をとる必要がある。

(b)　　PIN の保護

・　　PIN は物理的に保護するか、暗号化するべきである。

・　　異なるユーザが同じ PIN の値を持った場合でも、同じ暗号文にならないようにすべきである。

・　　暗号化して保管した PIN は置き換えられないように保護するべきである。

(c)　　PIN の検証

・　PIN を暗号化して保管するための暗号鍵と、伝送する時に使う暗号鍵は異なる鍵を使用するべきである。（例えば、認証のためにユーザが入力した PIN を保護する暗号鍵と、その検証のためにシステムが保持している PIN を保護する暗号鍵は異なる暗号鍵を使用すべきである。）
・　暗号的手法にて PIN を検証する方法として MAC(Message Authentication Code)がある

### （イ）備えるべき条件

PIN の保護において備えるべき条件として共通鍵暗号または公開鍵暗号（守秘）にて暗号化/復号が行えること。PIN の生成に乱数を用いている場合には乱数生成系にて乱数が生成できること。

### （ウ）標準化動向

ATM や POS についての PIN の管理方法は ISO9564 Banking － Personal Identification Number(PIN) management and security として標準化されている。

### （エ）暗号アルゴリズムの安全性と、PIN の暗号化の安全性の関係

PIN の暗号化は共通鍵暗号または公開鍵暗号（守秘）により行われる。これらの暗号が危殆化した場合には、保管されている暗号化された PIN が漏洩する恐れがある。
また、PIN の生成時に乱数を用いている場合においては、生成に用いている擬似乱数生成系には
・　同一の値が生成されないよう生成される値に十分な長さがあること
・　生成される値に偏りがないこと
・　生成される値が予測できないこと
が要求される。
乱数として同一の値が生成される場合や生成される値が予測できる場合には、生成された PIN を推測されてしまう恐れがある。

### （オ）推奨される利用方法

PIN の暗号化での推奨暗号アルゴリズムおよび鍵長は、以下の通りである。
PIN の暗号化に用いる共通鍵暗号アルゴリズム：
・　Camellia 128bit 以上
・　CIPHERUNICORN-A 128bit 以上
・　Hierocrypt-3 128bit 以上
・　AES 128bit 以上
・　SC2000 128bit 以上

※対象となるデータの有効期限によっては、リスト掲載の 64 ビットブロック暗号も対象となる。

PIN の暗号化に用いる公開鍵暗号アルゴリズム（守秘）：

・ RSA（RSA-OAEP） 2048bit 以上

PIN の生成に用いる乱数生成系：

・ PRNG based on SHA-1 in ANSI X9.42-2001 Annex C.1、

・ PRNG based on SHA-1 for general purpose in FIPS 186-2(+change notice 1) Appendix 3.1、

・ PRNG based on SHA-1 for general purpose in FIPS 186-2 (+change notice 1) revised Appendix 3.1

MAC：

・HMAC

SHA-1/256/384/512 と利用することを推奨

・CBC-MAC

・EMAC

・OMAC/CMAC

・XCBC-MAC（RFC 3566(AES-XCBC-MAC-96)として）

AES 128bit 以上、Camellia 128bit 以上、CIPHERUNICORN-A 128bit 以上、Hierocrypt-3 128bit 以上、SC2000 128bit 以上、を利用することを推奨

### (2) SSL-VPN

#### （ア）技術概要

SSL-VPN はインターネット VPN の一つであり、オープンなネットワークであるインターネット上でプライベートなネットワーク同士が仮想的な専用線を使用したいときに適用する技術・プロトコルの名称である。

SSL-VPN で使用する暗号プロトコルは SSL であるため、対策技術については「SSL」の章を参照。

### (3) IPsec-VPN

#### （ア）技術概要

IPsec-VPN はインターネット VPN の一つであり、オープンなネットワークであるインターネット上でプライベートなネットワーク同士が仮想的な専用線を使用したいときに適用する技術・プロトコルの名称である。VPN には主な接続形態に拠点間 VPN とリモートア

クセス VPN の 2 種類がある。

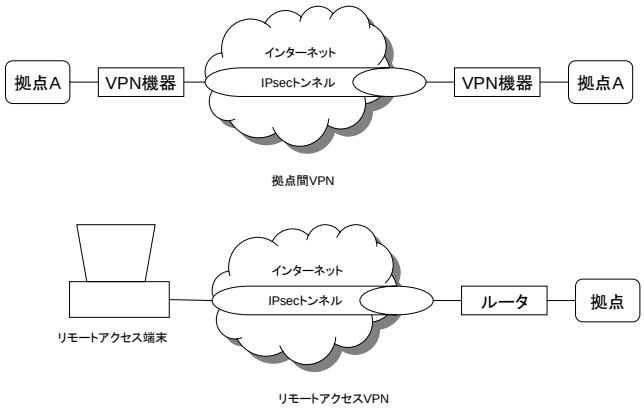



図 1 接続形態による VPN の区別

拠点間 VPN とリモートアクセス VPN のどちらも IPsec-VPN を用いて構成することが可能である。しかしリモートアクセス VPN の場合は NAT を間にはさむ場合もあり、相互運用性は拠点間 VPN に劣る。

IPsec-VPN で使用する暗号プロトコルは IPsec であるため、ここでは IPsec で使用する通信路の暗号化について記述する。

IPsec は ESP (Encapsulating Security Payload)、AH (Authentication Header)、IKE (Internet Key Exchange)の 3 つのプロトコルの総称である。IKE には IKEv1 と IKEv2 が存在するが、ここでは最新バージョンである IKEv2 について述べる。

ESP はメッセージの暗号化と改ざんされていないかの完全性の検証を行う。ESP が使用する暗号アルゴリズムは共通鍵暗号、MAC、ハッシュ関数である。

AH はメッセージが改ざんされていないかの完全性の検証だけを行う。AH で使用する暗号アルゴリズムは、共通鍵暗号、MAC、ハッシュ関数である。

IKEv2 は以下のサービスを提供する。

- 相手認証(事前鍵共有方式、デジタル証明書方式)
- SA (Security Association) の折衝と管理
- 共有鍵の管理

IKEv2 が使用する暗号アルゴリズムはで公開鍵暗号(署名)、公開鍵暗号(鍵共有)、共通鍵暗号、ハッシュ関数、MAC および擬似乱数である。SA とは単方向のコネクションにおいてどのような暗号アルゴリズム、認証アルゴリズムを使うか、またその有効期間などのセキュリティパラメータの集合である。

(a)　IPsec 通信の手順

ここではデジタル署名方式を使用した IKEv2 による鍵共有と、AH による改ざん検知、ESP による暗号化と改ざん検知を行うものとする。また、始動者は IPsec 通信の開始側を表し、応答者は IPsec 通信の受信側を表すとする。

IPsec 通信は以下の手順で実行される。

1. IKE_SA_INIT による IKE_SA 折衝
2. IKE_AUTH による相手認証と CHILD_SA 折衝
3. AH、 ESP による暗号通信
4. CREATE_CHILD_SA による CHILD_SA 折衝と鍵共有

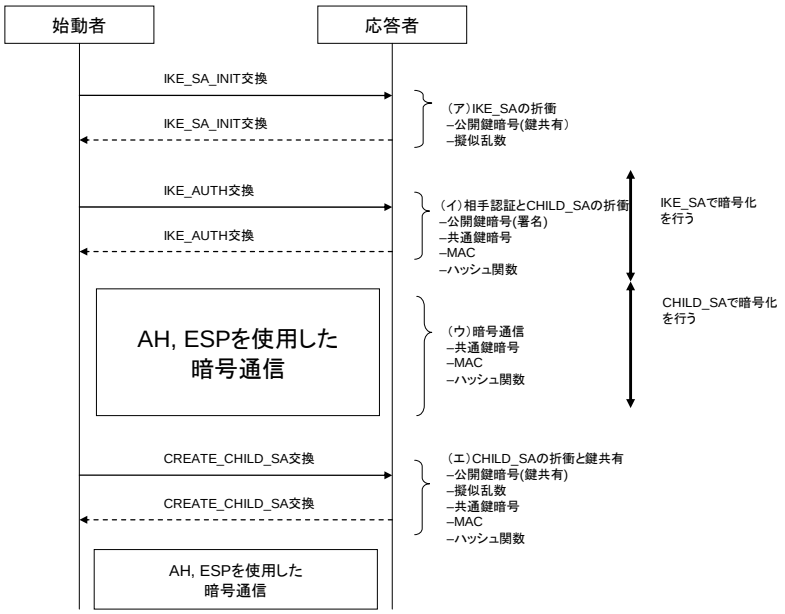



図 2　IPsec の接続手順

IPsec 通信上の、暗号利用方法は以下のとおり。

### (i)IKE のための鍵共有

IKE では AH、ESP のための鍵を交換する通信自体を暗号化するための SA 折衝と鍵共有を IKE_SA_INIT 交換で行う。 この手順で確立する SA を IKE_SA と言う。ここで確立された IKE_SA により、IKE_AUTH、CREATE_CHILD_SA のセキュリティが担保される。

始動者は自らが受け入れることができる SA に優先順位をつけて応答者に送信し、応答者はその中の一つを選択して始動者に伝えることにより折衝が完了する。

またここでは鍵共有を行う。

### (ii)相手認証と CHILD_SA の折衝

次に IKE は IKE_SA で確立した SA を用いて、CHILD_SA の折衝を行う。

始動者は自らが受け入れることができる CHILD_Sa に優先順位をつけて応答者に送信し、応答者はその中の一つを選択して始動者に伝えることにより折衝が完了する。また、デジタル署名を使用した認証を使用する場合、証明書を用いた相互認証を行う。

### (iii)暗号通信

確立した CHILD_SA を用いて AH と ESP は暗号通信を行う。

### (iv)CHILD_SA の折衝と鍵共有

SA の有効期限が近づいた場合、IKE は鍵を更新するため CHILD_SA の再交換を行う。PFS (Perfect Forward Security)を設定している場合は、DH、ECDH で使用する鍵の再交換も行う。ここでも IKE_SA を使用して暗号通信を行う。

PFS とはある時点で鍵が漏洩した場合でも、その時に使っていた鍵の有効期限以降の通信の安全性を保つ性質である。

## （イ）備えるべき条件

デジタル署名を用いた IKEv2、ESP、AH が備えるべき条件は、PKI 環境が存在し、始動者と応答者が公開鍵／秘密鍵のペアと公開鍵証明書を所有していることが運用の前提である。また、始動者と応答者が双方で同じバージョンの IPsec をサポートし、少なくとも一つの同じ SA を満たす暗号アルゴリズム群を実装している必要がある。

## （ウ）標準化動向

IPsec は、IETF の IPsec WG によって標準化されていたが、現在この WG は終了している。IPsec WG は IPsec の標準化文書を多数確定したが、その主なドキュメントをここに列挙する。

- Security Architecture for the Internet Protocol (RFC4301)

・ IP Authentication Header (4302)
・ IP Encapsulating Security Payload (ESP) (RFC4303)
・ Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH) (RFC4305)
・ Internet Key Exchange (IKEv2) Protocol (RFC4306)
・ Cryptographic Algorithms for Use in the Internet Key Exchange Version 2 (IKEv2) (RFC4307)

### （エ）暗号アルゴリズムの安全性と、IPsec-VPN の安全性の関係

#### (i)IKE のための鍵共有

IKE のための鍵共有では、鍵共有では、公開鍵暗号(鍵共有)と擬似乱数を使用する。
鍵共有に利用されている公開鍵暗号（鍵共有）アルゴリズムが危殆化した場合は、共通鍵が漏洩し暗号通信内容を盗聴されてしまう可能性がある。
擬似乱数生成系には
・ 同一の値が生成されないよう生成される値に十分な長さがあること
・ 生成される値に偏りがないこと
・ 生成される値が予測できないこと
が要求される。
乱数として同一の値が生成される場合や生成される値が予測できる場合には、過去の電文を記録している悪意ある第三者が、同一の乱数による過去の電文を用いて、正当なユーザになりすますなど、攻撃の余地を与えることになる。

#### (ii)相手認証と CHILD_SA の折衝

相手認証と CHILD_SA の折衝では証明書の検証と、署名に公開鍵暗号(署名)、暗号通信に共通鍵暗号、完全性の検証に共通鍵暗号、MAC、ハッシュ関数を利用する。
公開鍵証明書に利用されているデジタル署名は、比較的長期間（1 年から 3 年が一般的）の有効期限を持つため、この有効期間に則した安全性を持つデジタル署名アルゴリズム、および鍵長を選択する必要がある。デジタル署名アルゴリズムに含まれる公開鍵暗号アルゴリズムまたはハッシュ関数に脆弱性が発生した場合、任意の内容で公開鍵証明書を偽造される可能性があるため、意図しないサーバまたはクライアントを正当なものと誤って認証する可能性がある。
暗号通信に利用されている共通鍵暗号は SA で定義されている有効期限ごとに更新される。
暗号通信に利用されている共通鍵暗号アルゴリズムに脆弱性が発生した場合、通信内容を盗聴されてしまう可能性がある。
完全性の検証に利用されている共通鍵暗号、HMAC およびハッシュ関数に脆弱性が発生し

た場合、何らかの方法で通信が改ざんされた場合に改ざんを検知できない可能性がある。

### (iii)暗号通信

暗号通信には共通鍵暗号と、(ii)同様完全性の検証のため MAC およびハッシュ関数を利用する。

暗号通信の安全性は、(ii)の暗号通信と完全性の項目と同様。

### (iv)CHILD_SA の折衝と鍵共有

CHILD_SA の折衝と鍵共有には共通鍵暗号(鍵共有)と暗号通信、完全性の検証が行われる。

鍵共有については、(i)同様、暗号通信と完全性の検証については(ii)同様。

## (オ)推奨される利用方法

### (i)IKE のための鍵共有

公開鍵暗号（鍵共有）: DH 2048bit 以上、 ECDH 192bit 以上

公開鍵暗号（署名）:

- RSA（RSASSA-PKCS1-v1_5） 2048bit 以上
- DSA 2048bit 以上
- ECDSA 224bit 以上

擬似乱数生成系： IPsec において擬似乱数はその仕様の中でいくつか定義されている。システムで仕様内の乱数を適宜使用すること。

### (ii)相手認証と CHILD_SA の折衝

公開鍵暗号（署名）:

- RSA（RSASSA-PKCS1-v1_5） 2048bit 以上
- DSA 2048bit 以上
- ECDSA 224bit 以上

擬似乱数生成系：(i)同様

共通鍵暗号：AES128bit 以上、 Camellia 128bit 以上

※対象となるデータの有効期限によっては、リスト掲載の 64 ビットブロック暗号も対象となる。

MAC :

- HMAC-SHA-1
- HMAC-SHA-256
- HMAC-SHA-384
- HMAC-SHA-512

・ AES-CMAC(AES 128bit)
・ AES-XCBC-MAC(AES 128bit)

### (iii)暗号通信

共通鍵暗号、MAC：(ii)同様

### (iv)CHILD_SA の折衝と鍵共有

公開鍵暗号、擬似乱数生成系：(i)同様
共通鍵暗号、MAC：(ii)同様

参考文献
マスタリング IPsec 第 2 版　馬場達也著 オライリージャパン 2006 年 8 月
訳語は上記著書を参考にした

## (4) 無線 LAN（WEP ,WPA）

### （ア）技術概要

無線によるローカルエリアネットワーク（無線 LAN）は今や広く使われるネットワーク技術であり、その利便性において企業内ネットワークにとどまらず、広く家庭内においても普及している。無線ネットワークはその方式上、電波を利用していることから第三者による通信の傍受が非常に容易である。よって安全な通信を行うためには暗号による保護が必須である。
無線 LAN における暗号化方式として、WEP、WPA、WPA2 という規格がある。

(a) WEP(Wired Equivalent Privacy)

WEP は無線 LAN の暗号化規格であり、ストリーム暗号(RC4)によって通信内容を保護する。暗号化の方式は「図 3　WEP による暗号化」の通りである。

1. 送信したいメッセージとヘッダから CRC32(Cyclic Redundancy Check 32bit : 巡回冗長検査)により ICV(Integrity Check Value : 完全性検査値)を生成する。これは送信したいメッセージが通信中に改ざんされなかったことを確認するために使われる。
2. 送信側と受信側であらかじめ共有した共通鍵(WEP キー)と初期化ベクトル(IV)を連結したものを RC4 の鍵とする。
3. メッセージと ICV を結合したものを 2.で用意した RC4 の鍵で暗号化する。

WEP は暗号研究者によって以下の観点で安全でないことが示されている。

・　　鍵配布のメカニズムがない。
・　　IV の生成ルールが規定されておらず、IV が短いため鍵が推測されやすい実装が多い。
・　　一つの鍵を使い続けるため、鍵を推測されやすい。
・　　WEP キーが短い 40bit の RC4 を使用している場合はストリーム暗号自体が脆弱である。
・　　完全性チェックを CRC-32 で行っており、改ざんされる危険性がある。

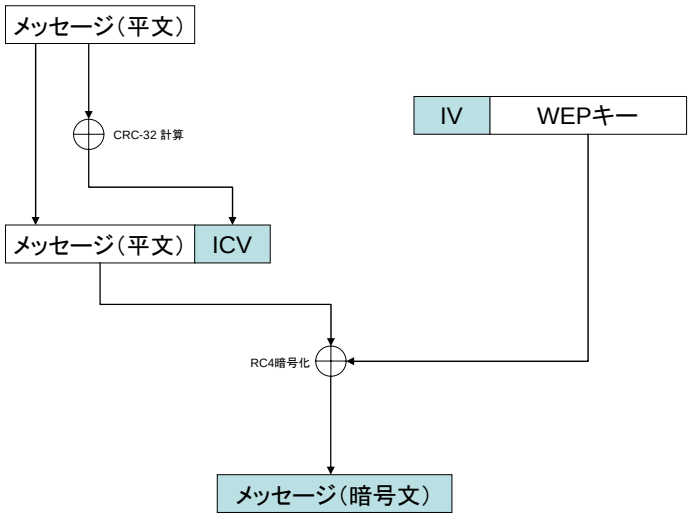



図 3　WEP による暗号化

WEP における暗号の利用箇所は次の通りとなる。
（ア）　共通鍵暗号
メッセージの暗号化に利用（RC4 ストリーム暗号）

(b)　　WPA
WEP の脆弱性を解決するため、2002 年 10 月に新たに WPA(WiFi Protected Access)が規定された。WPA では TKIP(Temporal Key Integrity Protocol)という暗号方式を採用し、WEP の脆弱性をカバーしている。具体的には IV の生成および RC4 の鍵（WEP での WEP キーに相当）を強化し、メッセージの完全性チェック(Message Integrity Check:MIC)もより強力な Michael アルゴリズムによる方式としている。

またWPAでは認証にネットワーク認証技術である802.1Xが利用できるようになっており、これを使用することにより送受信者で事前に鍵の共有を行わなくても、通信の開始時に自動的に鍵共有を行うことができる。802.1X は EAP(Extensible Authentication Protocol)と呼ばれる仕組みを備えており、認証の手段としてさまざまな方式を選択できるようになっている。例として EAP-TLS と呼ばれる、TLS プロトコル(RFC4346)を用いた証明書認証による方式がある。

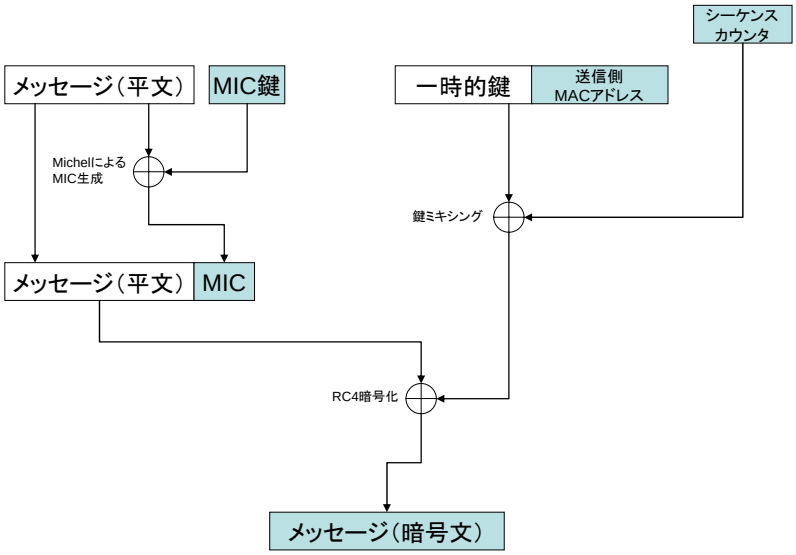



図 4　WPA(TKIP)の暗号化

WPA(TKIP)における暗号の利用箇所は次の通りとなる。

（イ）　共通鍵暗号

メッセージの暗号化に利用（RC4 ストリーム暗号）

（ウ）　MIC(メッセージ完全性チェック)

Michael による MIC 生成

(c)　WPA2

WPA は WEP の脆弱性に早期に対応するため、802.11i の標準化を待たず、早期にリリースされた。そのため 802.11i の仕様のうち、TKIP による暗号化を実装するにとどまっている。WPA2 では 802.11i の完成版の仕様を取り込んだ形で規定されている。

WPA での TKIP は既存の WEP の拡張という形で整備されたが、これは既存のハードウェア等もファームウェアのバージョンアップで対応できるようにするためである。これに対

して WPA2 ではより高いセキュリティを保てる暗号化方式である CCMP(Counter Mode with CBC-MAC Protocol)を採用している。CCMP の暗号化方式を図に示す。
なお、WPA2 は WPA 同様、802.1X による認証および鍵共有が可能である。

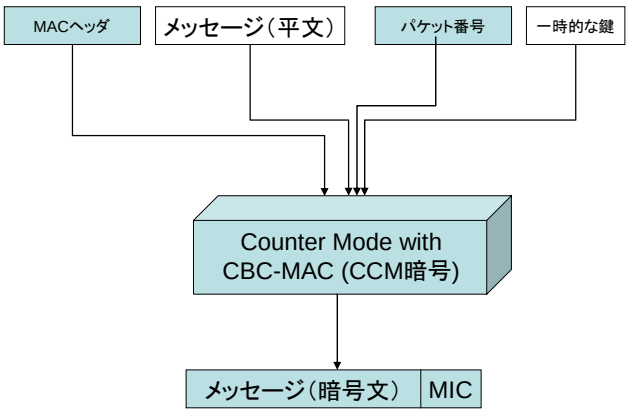



図 5　CCMP 暗号化方式

WPA2(CCMP)における暗号の利用箇所は次の通りとなる。
（エ）　共通鍵暗号
AES による Counter Mode with CBC-MAC

### （イ）備えるべき条件

(a)　　WEP
WEP においては、送信者/受信者間で WEP キーの事前共有が必要となる。逆に WEP キーが知られてしまえば、通信が盗聴可能となる。

(b)　　WPA/WPA2
WPA/WPA2 においては、認証手段により事前に備えるべき条件が異なる。
PSK（Pre-Shared Key）による認証の場合には、事前に送信者/受信者間で鍵の共有が必要となる。EAP による 802.1X 認証の場合で、例えば EAP-TLS を用いた場合には証明書による認証および鍵共有を行うために、事前に証明書の発行が必要となる。

### （ウ）標準化動向

WEP は IEEE802.11 で規格化されている。また、TKIP および CCMP は IEEE802.11i で規格化されている。また CCMP で利用される CCM(Counter Mode with CBC-MAC)は RFC3610 で標準化されている。

### （エ）暗号アルゴリズムの安全性と、無線 LAN の安全性の関係

(a) WEP

（ア）共通鍵暗号

WEP においては、メッセージの暗号化に 64bit または 128bit の RC4 が使用されている。64bit の RC4 は鍵長が充分でなく安全ではないため、使用するべきではない。また 128bit 以上の RC4 を使用したとしても、前述の通り WEP の仕組み自体が鍵を推測しやすい方式になっているため、RC4 の安全性や鍵長に関わらず、WEP を使用するべきではない。

(b) WPA

（イ）共通鍵暗号

WPA で暗号化方式として TKIP を使用する場合、WEP と同様 RC4 が使用される。使用される鍵長は 128bit である。128bit RC4 は安全であるとは言えないが、WEP に比べ鍵が推測しにくい仕組みとなっており、当面の使用は問題ない。

（ウ）MIC（メッセージ完全性チェック）

MIC 生成に利用されている Michael アルゴリズムが危殆化した場合、メッセージ完全性検証が正しく行われず、メッセージの改ざんが行われてしまう恐れがある。

(c) WPA2

（エ）共通鍵暗号

WPA2 で暗号化方式として CCMP を使用する場合、128bit AES を CCM(Counter Mode with CBC-MAC)で使用し暗号化しており、安全であるといえる。

### （オ）推奨される利用方法

(a) WEP

WEP は 暗号化方式の弱点から鍵を推測されやすくなっており、使用するべきではない。

(b) WPA

WPA は 128bit RC4 を使用しており、必ずしも安全であるとは言えないが、WEP に比べ鍵が推測されにくい仕組みになっており、当面の使用は問題ない。ただし、市販機器の対応状況を踏まえ、できるだけ早期に WPA2（CCMP）に移行すべきである。

(c)　　WPA2

WPA2(CCMP)については、以下の暗号を推奨とする。

暗号通信、MAC： CCM（カウンターモード AES-CBC-MAC、AES 128bit）

(d)　　WPA および WPA2 での認証および鍵共有を EAP-TLS で行う場合

証明書認証に利用する暗号アルゴリズムは、以下を推奨とする。

公開鍵暗号（署名）:

・　　RSA（RSASSA-PKCS1-v1_5） 2048bit 以上

・　　DSA 2048bit 以上

・　　ECDSA 224bit 以上

ハッシュ関数：SHA-1※

※　　利用は推奨されないが、EAP-TLS が RFC4346（TLS1.1）の規定に沿って実装されていれば他のアルゴリズムは選択できない。RFC4346 の規定が変更されるなどで、推奨されるアルゴリズムに変更できるようになった場合は、暗号の切り替え等を検討することが推奨される。

## 3.2. 鍵共有技術

### 3.2.1. 実現するセキュリティ機能

ネットワークに接続されている２者間（A と B とする）の通信において、２者間の通信内容の第三者からの盗聴を防止する機能を実現する。

通信路の暗号化技術には２者間で事前に秘密情報を共有しておき、その秘密情報を用いて暗号通信を行う形態と、鍵交換技術により通信を行う際に２者間で鍵交換を行う形態の技術があるが、最終的には暗号通信を開始する前の段階には２者間で暗号化に使用する鍵が共有される。

また、通信路の暗号化を行う際には、併せて通信相手の認証を行うことも多く、上記鍵交換を認証の過程で行う形態もある。

通信路の暗号化においては、以下がセキュリティ要件としてあげられる。

・　　暗号通信を行う２者間での通信が秘匿されており、第三者による盗聴が行えないこと。そのためには以下の要件を満たす必要がある。

　　鍵交換が安全に行われること

　　暗号用の鍵が脆弱でないこと

　　通信を長時間行う場合には暗号用の鍵の更新を考慮すること

### 3.2.2. 想定される脅威

通信路の暗号化においては、攻撃者が任意の A,B の２者間の通信を盗聴できるとする。攻撃者は通信を盗聴した上で、暗号用の鍵を盗み出すかもしくは推測することにより、通信内容を盗聴するという脅威が想定される。

以下、想定しうる脅威を挙げる。

・　鍵の漏洩による暗号通信の盗聴

事前共有鍵であれば、暗号通信を行う両拠点に同じ鍵を設定する運用の際に鍵の漏洩が起こる可能性がある。

通信時に鍵交換を行う場合において、攻撃者 C が２者間の通信を乗っ取ることができるとすると、２者間で第三者が鍵の交換手順に入り込むことによる攻撃（中間者攻撃：Man-in-the-middle attack）により鍵の漏洩が起こる可能性がある。

・　なりすまし

攻撃者 C が A および B が接続しているネットワークに接続可能とすると、C は２者（A,B）のどちらかになりすますことにより、暗号通信内容の漏洩が発生する可能性がある。

・　鍵の推測による暗号通信の盗聴

暗号通信に用いる鍵が脆弱である場合、総当り攻撃などで鍵が推測される可能性がある。

### 3.2.3. 対策方針

#### (1) 必須対策方針

・　事前鍵共有であれば安全な手順や運用で鍵の交換が行うことにより、鍵の漏洩を防止する

・　通信時に鍵交換を行うのであれば、第三者のなりすましや２者間に第三者が中継を行うことによる攻撃（中間者攻撃：Man-in-the-middle attack）に対して対策が施された鍵交換方法を用いること

・　全数攻撃などで鍵が推測されてしまわないよう、その時点で充分に安全であることが確認されている暗号アルゴリズムおよび鍵長を利用すること

#### (2) 推奨対策方針

・　通信を長期間行う場合に、同じ鍵を使い続けないように通信中に鍵を更新する等の対策を実施すること

### 3.2.4.鍵共有技術

#### (1)技術概要

鍵交換技術とは、鍵交換をおこなおうとする相手が正当な相手であるか認証した上で、相手と鍵を交換・共有するための技術である。
鍵交換技術には、複数の方式があり、方式により利用する暗号技術も異なっている。鍵交換を利用する環境で利用できる暗号技術、システムに求められるセキュリティ要件等を勘案し、適切な方式を選択する必要がある。

鍵交換の具体的な例、鍵交換の各方式の概要、利用する暗号技術、推奨される利用方法については、「Authenticated Key Exchange」の章を参照。

# 4. 蓄積データの暗号化

## 4.1. 蓄積データの暗号化

### 4.1.1. 実現するセキュリティ機能

重要な情報をシステム内に保管する場合、バックアップして２次媒体に保管する場合などにおいて、データを暗号化して第三者への漏洩を防止する機能を実現する。

蓄積データの暗号化技術は暗号化処理の高速から主に共通鍵暗号が使用される形態が多い。またその共通鍵を保護するためにさらに共通鍵やパスワード等がが用いられる形態や、利便性の観点から公開鍵暗号が用いられる形態がある。

蓄積データの暗号化においては、以下がセキュリティ要件としてあげられる。

・　暗号化されたデータが秘匿されており、第三者による解読が行えないこと。そのためには以下の要件を満たす必要がある。

　　暗号用の鍵が脆弱でないこと

　　暗号用の鍵が管理され漏洩しないようになっていること

### 4.1.2. 想定される脅威

蓄積データの暗号化において、攻撃者が暗号用の鍵の保管場所もしくは保持者に対してアクセスが可能であるとすると、攻撃者は暗号用の鍵を盗み出すことにより、暗号化したデータを解読するという脅威が想定される。

またアクセスできない場合においても暗号化されたデータを入手することができる場合には、攻撃者 C が暗号鍵を推測することにより、同様の脅威が想定される。

以下、想定しうる脅威を挙げる。

・　鍵の推測による暗号通信の盗聴

　蓄積データ暗号化に用いる鍵が脆弱である場合、総当り攻撃などで鍵が推測される可能性がある。

・　鍵の漏洩による蓄積データの漏洩

　共有データ等であればデータの暗号化/復号を行う鍵を複数のユーザに配布する必要がある。この運用の際に鍵の漏洩が起こる可能性がある。

### 4.1.3. 対策方針

#### (1) 必須対策方針

・　全数攻撃などで鍵が推測されてしまわないよう、その時点で充分に安全であることが確認されている暗号アルゴリズムおよび鍵長を利用すること

#### (2) 推奨対策方針

・　共有データ等であれば安全な手順や運用で鍵の配布を行うことにより、鍵の漏洩

を防止する

### 4.1.4. 蓄積データの暗号化で利用される対策技術

#### (1) ファイル暗号化（OpenPGP によるメッセージの暗号化）

#### （ア）技術概要

OpenPGP は 1991 年に Philip R. Zimmermann によって開発された PGP(Pretty Good Privacy)を元にし、IETF により RFC4880 として標準化されたメッセージ暗号化標準である。

OpenPGP では送信したいメッセージに対して暗号化および署名を行うことができ、公開鍵暗号（守秘/署名）、共通鍵暗号を用いて構成される。

(a)　暗号化

OpenPGP での暗号化は、共通鍵暗号および公開鍵暗号を組み合わせて行われている。暗号化は以下のように行われる。

1. 送信者は平文のメッセージを作成する。
2. 送信側の OpenPGP は（メッセージ毎に異なる）セッション鍵を生成する。
3. 送信側の OpenPGP は平文のメッセージをセッション鍵を共通鍵とする共通鍵暗号で暗号化する。
4. 送信側の OpenPGP はセッション鍵を受信者の公開鍵で暗号化する。
5. 暗号化したセッション鍵を 3 のメッセージに付加する。

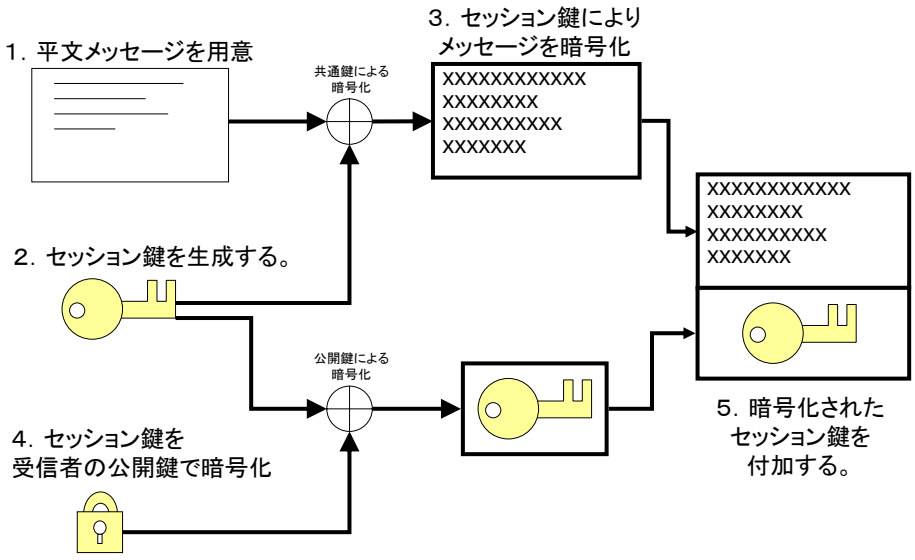


図 1　OpenPGP による暗号化

(b)　　復号

また、復号は以下のように行われる。

1.　　受信側の OpenPGP は送信されてきたメッセージから、暗号化されているセッション鍵を取り出し、セッション鍵を受信者の秘密鍵で復号する。結果として暗号化されていないセッション鍵が得られる。

2.　　受信側の OpenPGP は暗号化されたメッセージをセッション鍵を用いて復号し、平文のメッセージを得る。

図 2　OpenPGP による復号

※　　暗号化と同時に署名を行うこともでき、その際にはメッセージに署名を付与してから、メッセージと署名を合わせて暗号化する。

(c)　　署名

OpenPGP での署名は、ハッシュ関数および公開鍵暗号を組み合わせて行われている。署名は「図 3　OpenPGP による署名」のように行われる。

1.　　送信者は平文のメッセージを作成する。

2.　　送信側の OpenPGP はハッシュ関数によりメッセージのハッシュ値を生成する。

3.　　送信側の OpenPGP は送信者の秘密鍵とハッシュ値から署名値を生成する。

4.　　署名値をメッセージに付加する。

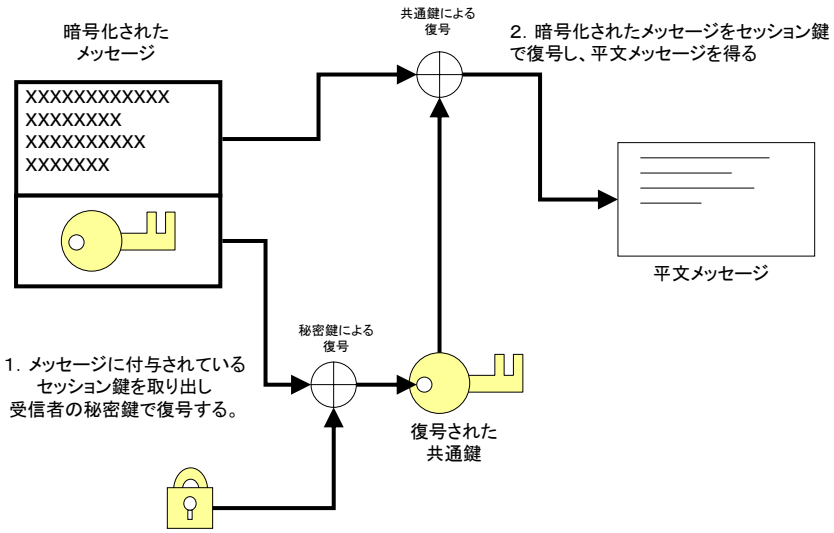



図 3　OpenPGP による署名

(d)　　署名の検証

また署名の検証は「図 4　OpenPGP による署名検証」のように行われる。

1.　　受信側の OpenPGP は署名値のコピーを取得する。

2.　　受信側の OpenPGP は受信したメッセージから新たに署名値を生成し、コピーしておいた署名値と一致することを検証する。

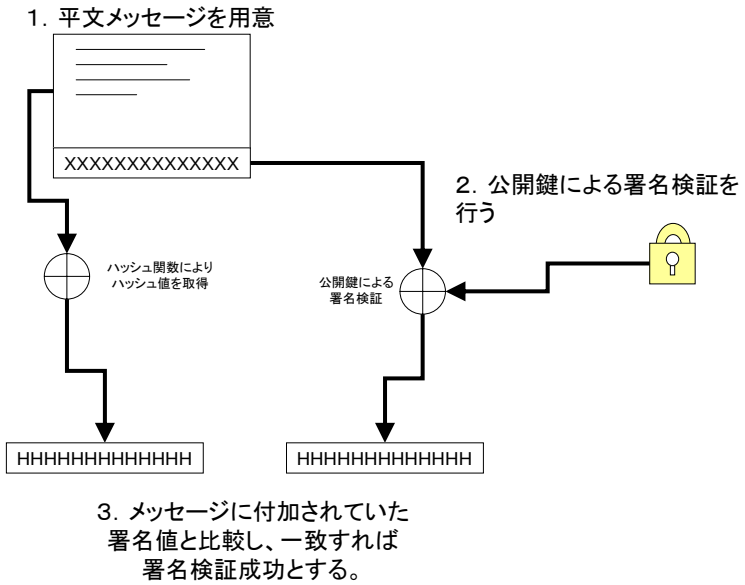



図 4　OpenPGP による署名検証

よって OpenPGP における暗号の利用箇所は次の通りとなる。

暗号化で利用する暗号：

**(i)乱数生成**

メッセージの暗号に用いるセッション鍵（共通鍵）の生成

**(ii)共通鍵暗号**

メッセージの暗号化、復号

**(iii)公開鍵暗号（守秘）**

セッション鍵（共通鍵）の暗号化、復号

署名で利用する暗号：

**(iv)ハッシュ関数**

署名時、検証時に利用

**(v)公開鍵暗号（署名）**

署名時、検証時に利用

### （イ）備えるべき条件

OpenPGP を利用するためには、メッセージの送信者、受信者間で自分の公開鍵を相手に通知しておく必要がある。

### （ウ）標準化動向

OpenPGP は IETF によって RFC4880 に「OpenPGP Message Format」として標準化されている。

### （エ）暗号アルゴリズムの安全性と、OpenPGP の安全性の関係

OpenPGP において、署名/検証ではハッシュ関数および公開鍵暗号、暗号化で共通鍵暗号および公開鍵暗号を使用している。したがって OpenPGP で使用している暗号技術は「乱数生成」、「共通鍵暗号」、「公開鍵暗号（守秘）」、「ハッシュ関数」、「公開鍵暗号（署名）」の 5 つとなる。

#### (i)乱数生成

#### (ii)共通鍵暗号

共通鍵暗号に脆弱性が発生した場合、セッション鍵（共通鍵）を推測することができてしまう恐れがあり、暗号化されたメッセージの復号が可能となる恐れがある。

また乱数生成に使用する擬似乱数生成系には

- 同一の値が生成されないよう生成される値に十分な長さがあること
- 生成される値に偏りがないこと
- 生成される値が予測できないこと

が要求される。

乱数として同一の値が生成される場合や生成される値が予測できる場合には、セッション鍵（共通鍵）を推測することができてしまう恐れがあり、暗号化されたメッセージの復号が可能となる恐れがある。

#### (iii)公開鍵暗号（守秘）

公開鍵暗号が危殆化した場合、暗号化されたセッション鍵（共通鍵）が復号され、それを利用してメッセージが復号が可能となる恐れがある。

#### (iv)ハッシュ関数

#### (v)公開鍵暗号（署名）

ハッシュ関数または公開鍵暗号（署名）が危殆化した場合、署名を偽造することが可能となるため、メッセージが改ざんされてしまう恐れがある。

### （オ）推奨される利用方法

OpenPGP での推奨暗号アルゴリズムおよび鍵長は、以下の通りである。

#### (i)乱数生成

OpenPGP においては擬似乱数生成系は RFC4086(Randomness Requirements for

Security)を参照し適切な擬似乱数生成系を使用すべきだとしている。本資料においては以下の乱数生成系を例示する。

・ PRNG based on SHA-1 in ANSI X9.42-2001 Annex C.1

・ PRNG based on SHA-1 for general purpose in FIPS 186-2 (+ change notice 1) Appendix 3.1

・ PRNG based on SHA-1 for general purpose in FIPS 186-2 (+ change notice 1) revised Appendix 3.1

**(ii)共通鍵暗号**

メッセージの暗号化に利用する暗号アルゴリズム： AES 128bit 以上

※対象となるデータの有効期限によっては、リスト掲載の 64 ビットブロック暗号も対象となる。

**(iii)公開鍵暗号（守秘）**

セッション鍵の暗号化に利用する暗号アルゴリズム：

・ RSA（RSAES-PKCS1-v1_5※） 2048bit 以上

**(iv)ハッシュ関数**

署名に利用するハッシュアルゴリズム：SHA-256、SHA-384、SHA-512

**(v)公開鍵暗号（署名）**

署名に利用する暗号アルゴリズム：

・ RSA（RSASSA-PKCS1-v1_5） 2048bit 以上

・ DSA 2048bit 以上

※ 利用は推奨されないが、RFC4880 規定上他のアルゴリズムは選択できない。RFC4880 規定が変更されるなどで、推奨されるアルゴリズムに変更できるようになった場合は、暗号の切り替え等を検討することが推奨される。

（参考）

なお、OpenPGP では複数の種類の暗号を組み合わせて使用することから、RFC4880(OpenPGP)の参考情報として NIST SP800-57 における鍵の強度の比較結果（等価安全性と鍵長の関係）を引用している。

公開鍵暗号の鍵長 ハッシュサイズ 共通鍵暗号の鍵長

1024 bit 160 bit 80 bit

2048 bit 224 bit 112 bit

3072 bit 256 bit 128 bit

7680 bit 384 bit 192 bit

15360 bit 512 bit 256 bit

### (2) DB の暗号化

#### (ア) 技術概要

データベース暗号化技術の例として Oracle 10g R2®以降に実装されている TDE (Transparent Data Encryption)がある。TDE はデータベースを透過的に暗号化/復号するため、ユーザがファイルアクセスに関して特別な操作を行わなくても自動的に暗号化/復号が行われる。

TDE では公開鍵暗号、共通鍵暗号を用いてこれを実現している。

(a) TDE による DB 暗号化の仕組み

TDE による DB の暗号化の際には、DB の列ごとに共通鍵を生成し、列ごとに暗号化を行う。この鍵を「列キー」と呼ぶ。また列キーはさらに「TDE マスターキー」と呼ばれる共通鍵で暗号化され保存される。

「TDE マスターキー」は「ウォレット」と呼ばれる専用のファイルに暗号化されて格納される。

暗号化は以下の手順で行われる。

1. DB に暗号化指定をした形で新しい表を作成すると、暗号化指定した列について「列キー」が生成される。
2. 「列キー」は「TDE マスターキー」により暗号化され DB 内に保存される。
3. 作成したテーブルに行を挿入すると、「列キー」によりデータの暗号化が行われる。

(b) TDE による DB 復号の仕組み

復号は以下のように行われる。

1. ユーザが暗号化されたテーブルの列にアクセスすると、その列に対応した「列キー」が取り出される。
2. 「列キー」を「TDE マスターキー」で復号する
3. 復号した「列キー」を用いて、対象の DB データを復号する。

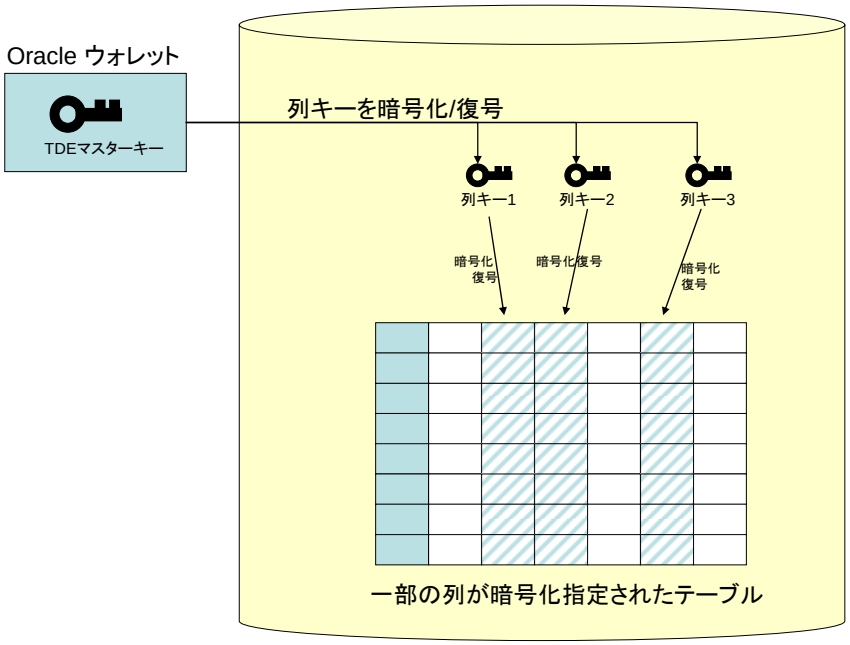



図 5　TDE による DB の暗号化/復号の仕組み

よって TDE における暗号の利用箇所は次の通りとなる。

**(i)乱数生成**

TDE マスターキーおよび列キーの生成

**(ii)共通鍵暗号**

DB の暗号化/復号、列キーの暗号化/復号

### （イ）備えるべき条件

TDE を利用するためには、「ウォレット」に「TDE マスターキー」を用意しておく必要がある。

### （ウ）標準化動向

TDE は Oracle の独自仕様であり標準化されていないが、利用している暗号アルゴリズムのは標準化されているものを使用している。

### （エ）暗号アルゴリズムの安全性と、TDE の安全性の関係

**(i)乱数生成**

使用する擬似乱数生成系には

・　同一の値が生成されないよう生成される値に十分な長さがあること
・　生成される値に偏りがないこと
・　生成される値が予測できないこと
が要求される。
乱数として同一の値が生成される場合や生成される値が予測できる場合には、TDE マスターキーおよび列キーを推測することができてしまう恐れがあり、暗号化されたファイルの解読が可能となる恐れがある。

### (ii)共通鍵暗号

TDE において、DB のデータを暗号化するための「列キー」で共通鍵暗号を、また「列キー」を暗号化するための「TDE マスターキー」においても共通鍵暗号を使用している。
共通鍵暗号に脆弱性が発生した場合、「TDE マスターキー」または「列キー」を推測することができてしまう恐れがあり、暗号化されたファイルの解読が可能となる恐れがある。
TDE マスターキーが解読されてしまった場合には DB 全体の情報が解読されてしまうことになるため、TDE マスターキーは列キーに比べより安全なアルゴリズムを利用し、厳重に保護されるべきである。

## (オ)推奨される利用方法

データベースの暗号化機能の実現においての推奨暗号アルゴリズムおよび鍵長は、以下の通りである。

### (i) 乱数生成

鍵生成に用いる擬似乱数生成系：
・　PRNG based on SHA-1 in ANSI X9.42-2001 Annex C.1
・　PRNG based on SHA-1 for general purpose in FIPS 186-2 (+ change notice 1) Appendix 3.1
・　PRNG based on SHA-1 for general purpose in FIPS 186-2 (+ change notice 1) revised Appendix 3.1

### (ii)共通鍵暗号

DB 暗号化に利用する共通鍵暗号アルゴリズム：
・　AES 128bit 以上
・　Camellia 128bit 以上
・　CIPHERUNICORN-A 128bit 以上
・　Hierocrypt-3 128bit 以上
・　SC2000 128bit 以上
※対象となるデータの有効期限によっては、リスト掲載の 64 ビットブロック暗号も対象となる。

また、Oracle の TDE における利用推奨暗号アルゴリズムおよび鍵長は、以下の通りである。

**(iii)乱数生成**

鍵生成の用いる擬似乱数生成系：とくに規定しない。

**(iv)共通鍵暗号**

DB 暗号化に利用する共通鍵暗号アルゴリズム：AES 128bit 以上

### (3) EFS（OS によるファイルの暗号化）

#### （ア）技術概要

OSによる暗号化ファイルシステムの例としてMicrosoft Windows 2000®以降に実装されている EFS(Encrypting File System)がある。EFS を利用することで、ユーザはファイルおよびディレクトリを暗号化し保護することができる。また、暗号化・復号は透過的に行われるため、ユーザがファイルアクセスに関して特別な操作を行わなくても自動的に暗号化/復号が行われる。

EFS では公開鍵暗号、共通鍵暗号を用いてこれを実現している。

(a)　EFS の暗号化の仕組み

EFS での暗号化は、基本的に共通鍵暗号により行われる。ファイルの暗号化に用いられる鍵 FEK(File Encryption Key)と呼ばれ、ファイル毎に生成される。またファイル暗号化に使用された FEK は、ファイルの復号の際に使用するため、ユーザの公開鍵で暗号化され保存される。これを「復号用 FEK」と呼ぶ。また、ユーザが公開鍵（秘密鍵）を紛失した際の回復用に、FEK を Recovery Agent の公開鍵で暗号化し保存する。これを「リカバリ用 FEK」と呼ぶ。

暗号化は以下の手順で行われる。

1. 暗号化したいファイルに対応する FEK を生成する。（乱数生成）
2. ファイルを FEK を鍵とした共通鍵暗号で暗号化する。（共通鍵暗号）
3. 暗号化に用いた FEK をユーザの公開鍵で暗号化し、「復号用 FEK」として保存する。（公開鍵暗号（守秘））
4. 同時に、同じ FEK を Recovery Agent の公開鍵で暗号化し、「リカバリ用 FEK」として保存する。（公開鍵暗号（守秘））

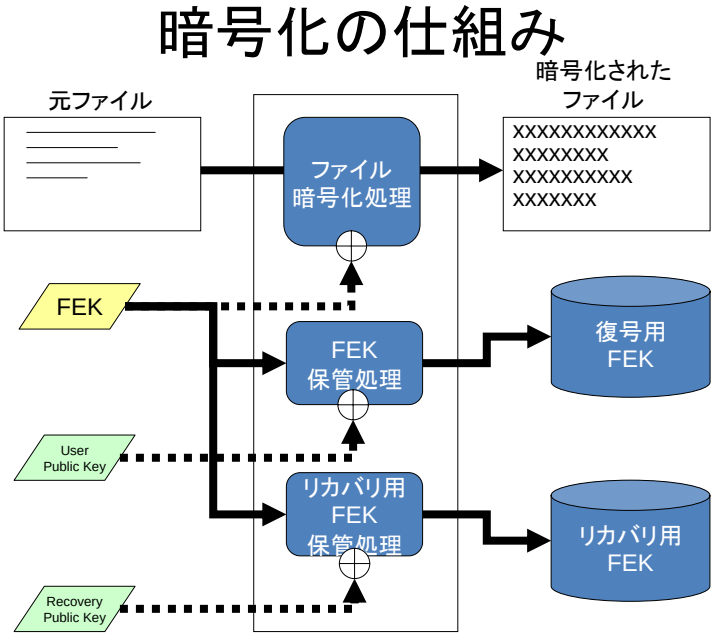



図 6　EFS 暗号化の仕組み

(b)　　EFS の復号の仕組み

復号は以下のように行われる。

1.　　復号したいファイルに対応する「復号用 FEK」を取得する。

2.　　復号用 FEK はユーザの公開鍵で暗号化されているため、これをユーザの秘密鍵で復号する。これにより FEK が得られる。(公開鍵暗号（守秘))

3.　　得られた FEK を用い、暗号化されたファイルを復号する。(共通鍵暗号)

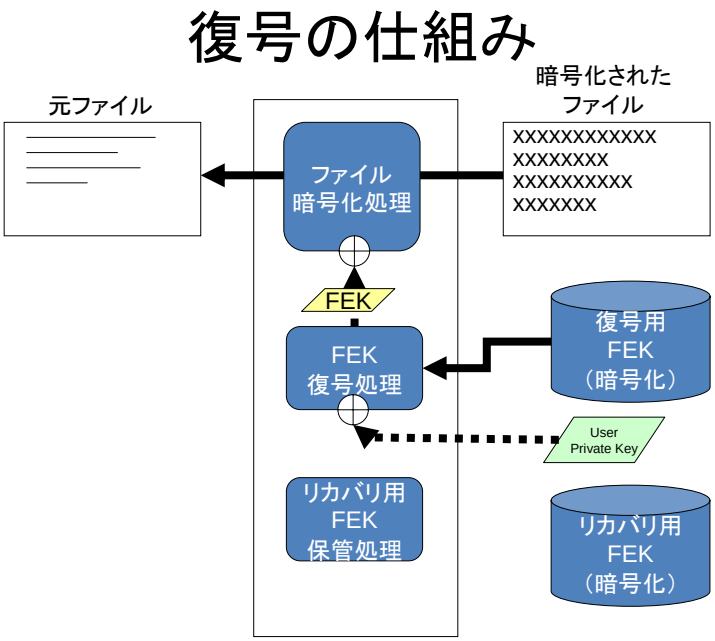



図 7　EFS 復号の仕組み

よって EFS によるファイル暗号化/復号における暗号の利用箇所は次の通りとなる。

**(i)乱数生成**

共通鍵（FEK）の生成

**(ii)共通鍵暗号**

ファイルの暗号化、復号

**(iii) 公開鍵暗号（守秘）**

共通鍵（FEK）の暗号化、復号

**(iv)証明書認証**

公開鍵暗号を利用する際には証明書認証を利用するため、公開鍵暗号（署名）およびハッシュ関数も利用する。

### （イ）備えるべき条件

EFS を利用するためには、OS が Windows2000®以降である必要があり、ファイルシステムは NTFS でフォーマットされている必要がある。また、Windows のユーザ用の電子証明書（および秘密鍵）が Windows の証明書ストアに格納されている必要がある。

### （ウ）標準化動向

EFS は Microsoft Corporation の独自仕様であり標準化されていないが、利用している暗

号アルゴリズムのは標準化されているものを使用している。

### （エ）暗号アルゴリズムの安全性と、EFS の安全性の関係

EFS で使用している暗号技術は「乱数」、「共通鍵暗号」、「公開鍵暗号（守秘）」および公開鍵証明書で利用する「共通鍵暗号（署名）」、「ハッシュ関数」の 5 つとなる。

#### (i)乱数生成

擬似乱数生成系には

- 同一の値が生成されないよう生成される値に十分な長さがあること
- 生成される値に偏りがないこと
- 生成される値が予測できないこと

が要求される。

乱数として同一の値が生成される場合や生成される値が予測できる場合には、FEK を推測することができてしまう恐れがあり、暗号化されたファイルの解読が可能となる恐れがある。

#### (ii)共通鍵暗号

共通鍵暗号に脆弱性が発生した場合、FEK を推測することができてしまう恐れがあり、暗号化されたファイルの解読が可能となる恐れがある。

#### (iii)公開鍵暗号（守秘）

#### (iv) 証明書認証

公開鍵証明書に利用されている公開鍵暗号アルゴリズムまたはハッシュ関数に脆弱性が発生した場合、任意の内容で公開鍵証明書を偽造される可能性があるため、暗号化されたファイルを解読されてしまう恐れがある。

### （オ）推奨される利用方法

EFS に類するファイル暗号化の仕組みを導入する場合の推奨暗号アルゴリズムおよび鍵長は、以下の通りである。

#### (i)乱数生成

共通鍵生成に利用する擬似乱数生成系

- PRNG based on SHA-1 in ANSI X9.42-2001 Annex C.1
- PRNG based on SHA-1 for general purpose in FIPS 186-2 (+ change notice 1) Appendix 3.1
- PRNG based on SHA-1 for general purpose in FIPS 186-2 (+ change notice 1) revised Appendix 3.1

#### (ii)共通鍵暗号

ファイルの暗号化に利用する共通鍵暗号アルゴリズム

- AES 128bit 以上

・ Camellia 128bit 以上
・ CIPHERUNICORN-A 128bit 以上
・ Hierocrypt-3 128bit 以上
・ SC2000 128bit 以上

※対象となるデータの有効期限によっては、リスト掲載の 64 ビットブロック暗号も対象となる。

### (iii)公開鍵暗号（守秘）

共通鍵の暗号化に利用する公開鍵暗号アルゴリズム

公開鍵暗号（守秘）: RSA（RSA-OAEP） 2048bit 以上

公開鍵暗号（鍵共有）:

・ PSEC-KEM 224bit 以上（KEM（Key Encapsulation Mechanism）-DEM(Data Encapsulation Mechanism)構成における利用を前提とする。）

### (iv) 証明書認証

公開鍵証明書の署名に利用する暗号アルゴリズム

公開鍵暗号（署名）:

・ RSA（RSASSA-PKCS1-v1_5） 2048bit 以上
・ DSA 2048bit 以上
・ ECDSA 224bit 以上

ハッシュ関数：SHA-256、SHA-384、SHA-512

また、EFS を用いる場合の推奨暗号アルゴリズムおよび鍵長は、以下の通りである。

### (v)乱数生成

共通鍵の生成に利用する擬似乱数生成系：特に規定しない。

### (vi)共通鍵暗号

ファイルの暗号化に利用する共通鍵暗号アルゴリズム：AES 256bit

### (vii)公開鍵暗号（守秘）

FEK の暗号化に利用する公開鍵暗号アルゴリズム：RSA 1024bit 以上※

### (viii)証明書認証

公開鍵証明書の署名に利用する暗号アルゴリズム

公開鍵暗号（署名）: RSA（RSASSA-PKCS-v1_5） 1024bit 以上※

ハッシュ関数：SHA-256、SHA-384、SHA-512

※ 利用は推奨されないが、EFS の実装上、他のアルゴリズムは選択できない場合がある。今後、推奨されるアルゴリズムに変更できるようになった場合は、暗号の切り替え等を検討することが推奨される。

# 5. 改ざん検知・時刻保証技術

## 5.1. 改ざん検知技術

### 5.1.1. 実現するセキュリティ機能

改ざん検知技術とは、ストレージ上や通信路上等にある電子データ（ドキュメントやメッセージ）の改ざんを検知する技術を提供する。

### 5.1.2. 想定される脅威

想定される脅威は、攻撃者による故意、あるいは、不注意による電子データの改ざんである。

### 5.1.3. 対策方針

#### (1) 必須対策方針

電子データに対して改ざん検知可能な情報を付加することにより、改ざん有無を検知できるようにする。

#### (2) 推奨対策方針

改ざん検知可能な情報に関する改ざんの脅威が想定されるため、改ざん検知可能な情報の改ざんを検知できるようにする。

### 5.1.4. 改ざん検知で利用される対策技術

#### (1) ハッシュ関数利用

##### (ア) 技術概要

ハッシュ関数とは、与えられた電子データから一定範囲の数値(ハッシュ値)を生成する関数である。ハッシュ関数は、次の二つの特徴を持つ。

・ 与えられた出力（ハッシュ値）に対して、この出力へ写像する入力（電子データ）を見つけることが計算量的に困難なこと。

・ 与えられた入力に対して、同じ出力へ写像する二つ目の入力を見つけることが計算量的に困難なこと。

・同じ出力へ写像する、任意の 2 つの入力のペアを見つけることが計算量的に困難であること。

このような性質を備えているハッシュ関数を利用することにより、電子データの改ざん有無を検証することができる。

(a)　　ハッシュ関数を用いた改ざん検知手順の例

1. ハッシュ値の生成とセキュアな保存・保管

・　　ハッシュ関数を用いて電子データのハッシュ値を求める。
・　　求めたハッシュ値を安全な領域に保存し、信頼のできるハッシュ値として管理する。

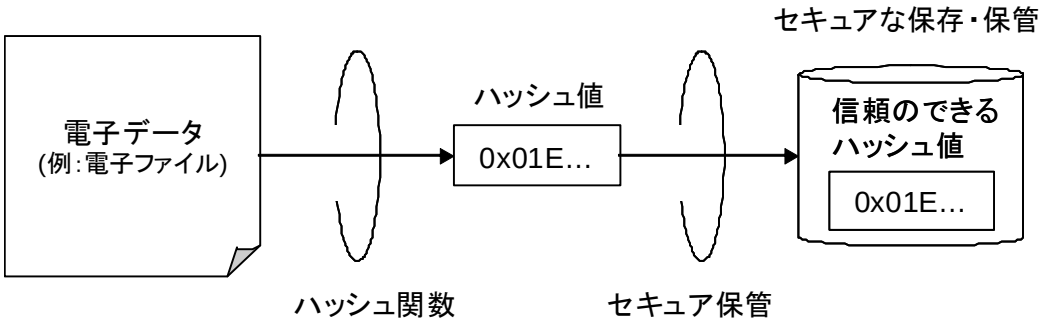



図 1　ハッシュ値の生成とセキュアな保管

2. ハッシュ関数を用いた電子データの改ざん検知
・　　改ざん検知対象となる電子データのハッシュ値を求める。
・　　求めたハッシュ値と保存された信頼のできるハッシュ値を照合する。
・　　ハッシュ値が同一であれば、非改ざんであることが分かる。

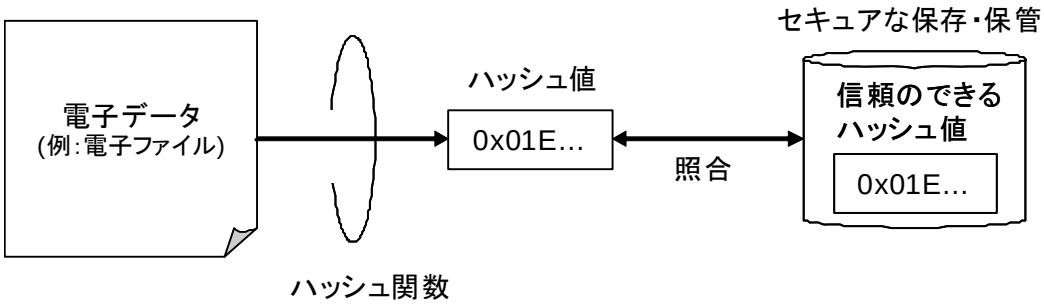



図 2　ハッシュ関数を用いた電子データの改ざん検知

### （イ）備えるべき条件

電子データの改ざん検知のためにハッシュ関数が利用される場合、照合対象となるハッシュ値がセキュアに保存・保管が行われており、そのハッシュ値が信頼できることが必要である。

### （ウ）標準化動向

ハッシュ関数は、ISO/IEC 10118 シリーズとして国際規格になっている。また、米国の FIPS 180-2 として標準化されている。

### （エ）暗号アルゴリズムの安全性とハッシュ関数利用の安全性との関係

ハッシュ関数は、暗号アルゴリズムの一種と言える。そのため、ハッシュ関数を利用した電子データの改ざん検知技術の安全性は、ハッシュ関数自体のアルゴリズムの安全性に依存する。

脆弱性を持つハッシュ関数が利用されると、攻撃者が都合良く改ざんした電子データを改ざんされたものではないと誤判定する可能性がある。そのため、安全性が確認されたハッシュ関数を使用することが重要である。また、使用しているハッシュ関数に脆弱性が見つかった場合は、速やかに、安全なハッシュ関数に切り替えることが必要になる。

### （オ）推奨される利用方法

文献[1] [2]を踏まえると、推奨されるハッシュ関数は、以下の通りである。

**表 1　推奨ハッシュ関数**

| 分類 | アルゴリズム | 備考 |
|---|---|---|
| ハッシュ関数 | SHA-2 シリーズ | SHA-256/SHA-384/SHA-512 |

## (2) MAC

### （ア）技術概要

MAC とは、メッセージ認証の機能を提供する技術である。メッセージの送信者は、メッセージ受信者と共有する鍵を用いてメッセージの改ざん有無を検知するためのメッセージ認証コードを作成する。受信者は、メッセージ認証コードを用いて受信したメッセージの改ざん有無を検証する。

(a)MAC による改ざん検知手順例

1. メッセージ送信者とメッセージ受信者は、予め、同じ鍵を共有しておく。
2. 送信者は、メッセージを作成する。
3. 送信者は、メッセージと共有している鍵から、ブロック暗号アルゴリズム、あるいは、専用ハッシュ関数を用いてメッセージ認証コードを計算する。
4. 送信者は、メッセージとメッセージ認証コードをメッセージ受信者に送信する。
5. 受信者は、メッセージとメッセージ認証コードを受信する。
6. 受信者は、共有している鍵と受信メッセージとから、ブロック暗号アルゴリズム、あるいは、専用ハッシュ関数を用いてメッセージ認証コードを計算し、送信者が計算したメッセージ認証コードと同一結果が得られれば、メッセージの非改ざん性が分かる。

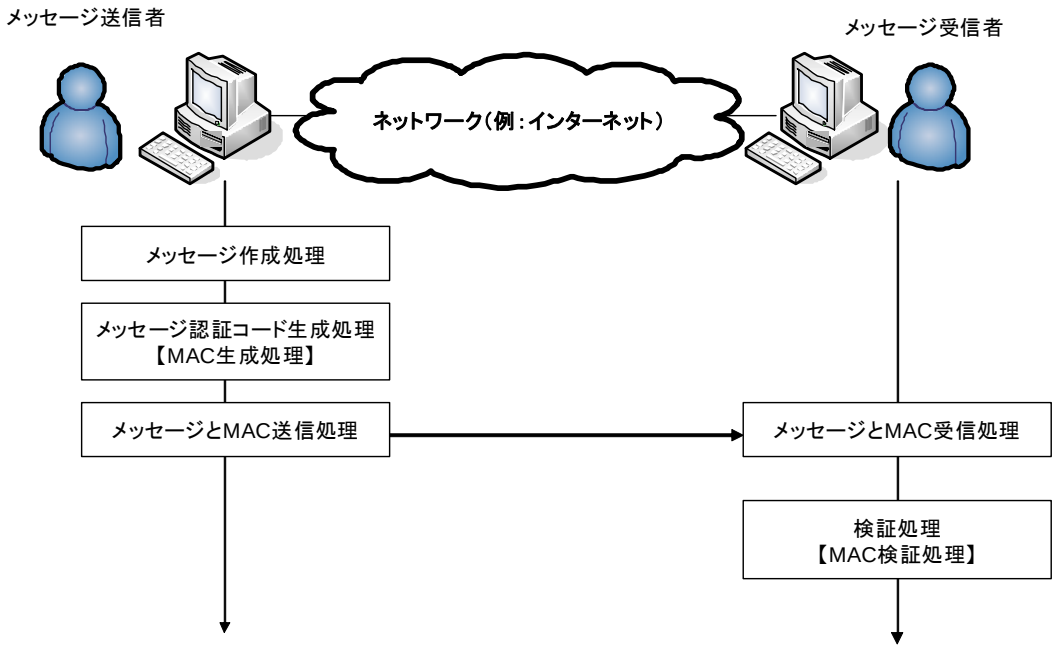



図 3　MAC による改ざん検知手順例

MAC での暗号利用方法は、以下の通りである。

### (i)MAC 生成処理

送信者は、ブロック暗号アルゴリズム、あるいは、専用ハッシュ関数を用いて、メッセージ認証コードを作成する。

### (ii)MAC 検証処理

受信者は、送信者が使用したブロック暗号アルゴリズム、あるいは、専用ハッシュ関数を用いて、メッセージ認証コードを検証する。

#### （イ）備えるべき条件

メッセージ送信者とメッセージ受信者が共有する鍵が第 3 者に漏えいしていないことが必要である。

#### （ウ）標準化動向

ISO/IEC 9797-1:1999 及び ISO/IEC 9797-2:2002 として国際規格となっている。また、IETF によって、RFC 2104(HMAC)、RFC 3566(AES-XCBC-MAC-96)として標準化されている。米国では NIST SP 800-38B として標準化されている。

### （エ）暗号アルゴリズムの安全性と MAC の安全性との関係

MAC 生成処理/検証処理において、ブロック暗号、あるいは、ハッシュ関数が利用される。MAC の安全性は、ブロック暗号、あるいは、ハッシュ関数の安全性に依存する。

脆弱性を持つブロック暗号、あるいは、ハッシュ関数が利用されると、攻撃者が都合良く改ざんしたメッセージ（電子データ）を改ざんされたものではないと誤判定する可能性がある。そのため、安全性が確認されたブロック暗号、あるいは、ハッシュ関数を使用することが重要である。また、使用しているブロック暗号、あるいは、ハッシュ関数に脆弱性が見つかった場合は、速やかに、安全なブロック暗号、あるいは、ハッシュ関数に切り替えることが必要である。

### （オ）推奨される利用方法

文献[3]を踏まえると、MAC での推奨暗号アルゴリズムは、以下の通りである。

**表 2　推奨 MAC**

| 分類 | アルゴリズム | 備考 |
|---|---|---|
| ブロック暗号を用いた MAC | CBC-MAC | 128 ビットのブロック暗号(※1)を使うことを推奨。 |
| | EMAC | |
| | OMAC/CMAC | |
| | XCBC-MAC(※3) | |
| 専用ハッシュ関数を用いた MAC | HMAC | SHA-1(※2)/256/384/512 を使うことを推奨。 |

※1 擬似ランダム置換族であることを仮定する。

※2 積極的な利用は推奨しないが、NIST におけるハッシュ関数利用ポリシー（http://csrc.nist.gov/groups/ST/hash/policy.html）では、2010 年以降であっても、HMAC で SHA-1 の利用を許している。

※3 RFC 3566(AES-XCBC-MAC-96)として。

## (3) デジタル署名利用

### （ア）技術概要

デジタル署名技術とは、電子データの真正性と完全性を保証する機能を提供する。デジタル署名のうち、PKI 技術をベースにしたデジタル署名技術は、広く利用されている技術である。ここでは、デジタル署名とは、デジタル署名を示すものとして記述する。

(a) デジタル署名による改ざん検知手順例

あるユーザが作成した電子ファイルに対してデジタル署名を付与し、ネットワークを介し

て相手にそのファイルを送信するという例を考える。このとき、デジタル署名を用いた電子ファイルの改ざん検知の手順は、次のようになる。

1. 電子ファイル送信者は、電子ファイルを作成する。
2. 送信者は、自分の秘密鍵を用いて、電子ファイルに対して署名処理を行う。
3. 送信者は、署名が付与された電子ファイルを送信する。
4. 電子ファイル受信者は、署名が付与された電子ファイルを受信する。
5. 受信者は、電子ファイルの署名を検証する。

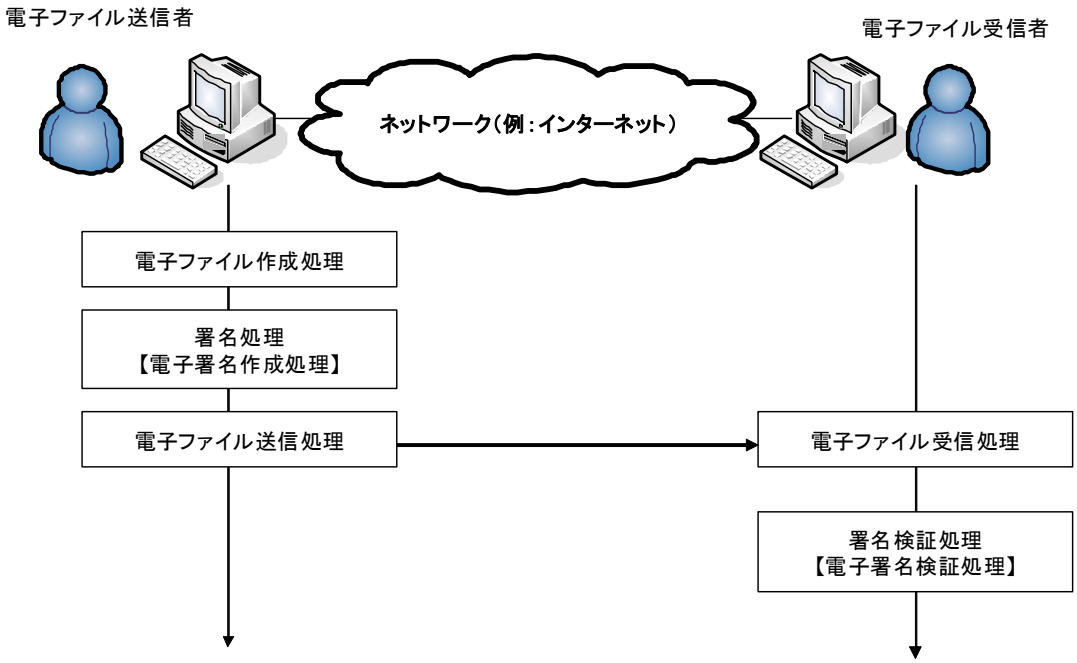



図 4 デジタル署名による改ざん検知手順例

デジタル署名における暗号利用方法は、以下の通りである。

### (i)デジタル署名作成

電子ファイル送信者は、デジタル署名作成時に、以下の手順を実行する。

・ 電子ファイル送信者は、ハッシュ関数を用いて電子ファイルのハッシュ値を求める。

・ そのハッシュ値を公開鍵暗号方式の秘密鍵を用いて署名付与を行う。

### (ii)デジタル署名検証

電子ファイル受信者は、以下の手順でデジタル署名を検証する。

・ トラストアンカとなる公開鍵証明書（例：認証局のルート CA 証明書）を基点として、送信者の公開鍵証明書を検証する。

・ ハッシュ関数を用いて電子ファイルのハッシュ値を求める。

・ 送信者の公開鍵証明書に含まれた公開鍵を用いて、署名検証を行う。この際、データフォーマットの全ビットの比較によりチェックを行う。

### （イ）備えるべき条件

PKI 環境が存在することが前提である。電子ファイル作成者は、署名に利用する公開鍵/秘密鍵のペア及び公開鍵証明書を所有している。また、認証局が公開する CP/CPS に基づいて運用が行われていることも必要である。

### （ウ）標準化動向

デジタル署名は、IETF において、RFC 3447 として標準化されている。また、米国の FIPS 186-2(DSS)として標準化されている。

### （エ）暗号アルゴリズムの安全性とデジタル署名利用の安全性との関係

デジタル署名では、デジタル署名対象の電子データのハッシュ化の手段としてハッシュ関数が使われている。また、求められたハッシュ値の暗号化のために、公開鍵暗号が利用されている。したがって、デジタル署名で使用されている暗号技術は、「公開鍵暗号」と「ハッシュ関数」となる。

脆弱性を持つ公開鍵暗号、あるいは、ハッシュ関数が利用されると、攻撃者が都合良く改ざんした電子ファイルを改ざんされたものではないと誤判定する可能性がある。そのため、安全性が確認された公開鍵暗号、あるいは、ハッシュ関数を使用することが重要である。また、使用している公開鍵暗号、あるいは、ハッシュ関数に脆弱性が見つかった場合は、速やかに、安全な公開鍵暗号、あるいは、ハッシュ関数に切り替えることが必要である。

### （オ）推奨される利用方法

文献[1][2]を踏まえると、デジタル署名での推奨暗号アルゴリズム及びパラメータは、以下の通りである。

**表 3　公開鍵証明書の発行に利用する推奨デジタル署名アルゴリズム（※）**

| 分類 | アルゴリズム | パラメータ |
|---|---|---|
| 公開鍵暗号 | RSASSA-PKCS1-v_1_5,<br>RSA-PSS | 鍵長 2048 ビット以上 |
| | DSA | 鍵長 2048 ビット以上 |
| | ECDSA | 鍵長 224 ビット以上 |
| ハッシュ関数 | SHA-2 シリーズ | SHA-256/SHA-384/SHA-512 |

※認証局が使用すべき推奨である。

表 4　メッセージへの署名に利用する推奨デジタル署名アルゴリズム

| 分類 | アルゴリズム | パラメータ |
|---|---|---|
| 公開鍵暗号 | RSASSA-PKCS1-v_1_5,<br>RSA-PSS | 鍵長 2048 ビット以上 |
| | DSA | 鍵長 2048 ビット以上 |
| | ECDSA | 鍵長 224 ビット以上 |
| ハッシュ関数 | SHA-2 シリーズ | SHA-256/SHA-384/SHA-512 |

### (4) S/MIME

#### （ア）技術概要

S/MIME とは、電子メールをセキュアに送受信する技術である。電子メールのフォーマットの一つである MIME(Multipurpose Internet Mail Extensions)データ対する認証、完全性、送信者の否認防止、ネットワーク上の機密性確保、を実現する。

S/MIME では、電子メールの送信者は、送信する電子メールに対してデジタル署名を付与できる。これにより、電子メールの認証及び改ざん検知が可能になる。

(a) S/MIME による改ざん検知手順例

S/MIME を用いた電子メールの改ざん検知の手順は、次のようになる。

1. 電子ファイル送信者は、電子メール本文を作成する。
2. 送信者は、自分の秘密鍵を用いて、電子メールに対して署名処理を行う。
3. 送信者は、署名が付与された電子メールを送信する。
4. 電子メール受信者は、署名が付与された電子メールを受信する。
5. 受信者は、電子メールの署名を検証する。

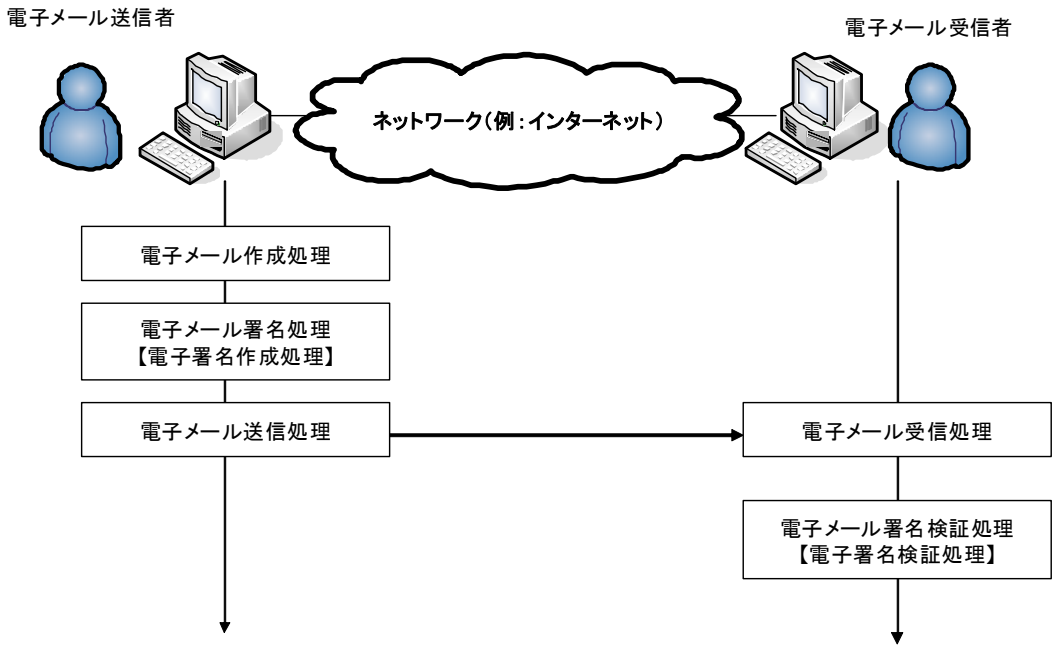



図 5　S/MIME による改ざん検知手順例

S/MIME における暗号利用方法は、以下の通りである。

### (i)デジタル署名作成

電子メール送信者は、以下の手順により電子メールに対するデジタル署名を作成する。

・ 電子メール送信者は、ハッシュ関数を用いて電子メールのハッシュ値を求める。

・ そのハッシュ値を公開鍵暗号方式の秘密鍵で署名生成を行う。

### (ii)デジタル署名検証

電子メール受信者は、以下の手順でデジタル署名を検証する。

・ トラストアンカとなる公開鍵証明書（例：認証局のルート CA 証明書）を基点として、送信者の公開鍵証明書を検証する。

・ ハッシュ関数を用いて電子メールのハッシュ値を求める。

・ 送信者の公開鍵証明書に含まれた公開鍵を用いて、署名検証処理を行う。この際、データフォーマットの全ビットの比較によりチェックを行う。

### （イ）備えるべき条件

PKI 環境が存在することが前提である。電子メールの送信者は、S/MIME に利用する公開鍵/秘密鍵のペア及び公開鍵証明書を所有している。また、認証局が公開する CP/CPS に基づいて運用が行われていることも必要である。

### （ウ）標準化動向

IETF にて、RFC 3850、RFC 3851、RFC 3852 として標準化されている。

### （エ）暗号アルゴリズムの安全性と S/MIME の安全性との関係

S/MIME では、メッセージの認証及び改ざん検知のために、CMS(Cryptographic Message Syntax)が使われている。

RFC 3370 では、CMS で使用される暗号アルゴリズムが記されており、具体的には、「公開鍵暗号」と「ハッシュ関数」となる。S/MIME の安全性は、これらの暗号技術の安全性に依存することになる。

脆弱性を持つ公開鍵暗号、あるいは、ハッシュ関数が利用されると、攻撃者が都合良く改ざんした電子メールを改ざんされたものではないと誤判定する可能性がある。そのため、安全性が確認された公開鍵暗号、あるいは、ハッシュ関数を使用することが重要である。また、使用している公開鍵暗号、あるいは、ハッシュ関数に脆弱性が見つかった場合は、速やかに、安全な公開鍵暗号、あるいは、ハッシュ関数に切り替えることが必要である。

### （オ）推奨される利用方法

文献[1][2]を踏まえると、S/MIME での推奨暗号アルゴリズム及びパラメータは、以下の通りである。

**表 5　公開鍵証明書の発行に利用する推奨デジタル署名アルゴリズム（※）**

| 分類 | アルゴリズム | パラメータ |
|---|---|---|
| 公開鍵暗号 | RSASSA-PKCS1-v_1_5,<br>RSA-PSS | 鍵長 2048 ビット以上 |
| | DSA | 鍵長 2048 ビット以上 |
| | ECDSA | 鍵長 224 ビット以上 |
| ハッシュ関数 | SHA-2 シリーズ | SHA-256/SHA-384/SHA-512 |

※認証局が使用すべき推奨である。

**表 6　電子メールへの署名に利用する推奨デジタル署名アルゴリズム**

| 分類 | アルゴリズム | パラメータ |
|---|---|---|
| 公開鍵暗号 | RSASSA-PKCS1-v_1_5,<br>RSA-PSS | 鍵長 2048 ビット以上 |
| | DSA | 鍵長 2048 ビット以上 |
| | ECDSA | 鍵長 224 ビット以上 |
| ハッシュ関数 | SHA-2 シリーズ | SHA-256/SHA-384/SHA-512 |

### (5) コード署名

#### （ア）技術概要

コード署名技術とは、バイナリ形式の実行ファイルやテキストベースのスクリプトなどのプログラムの開発者を認証する機能、及びプログラムの改ざん有無を確認する機能を提供する。

コード署名技術として、以下の技術が実用化されている。

・Microsoft 社の Authenticode

・Sun 社の Java SDK が提供するコード署名(Java コード署名)

両者の技術は、公開鍵暗号技術に基づくデジタル署名をプログラムに対して付与するものである。

(a)コード署名による改ざん検知手順例

コード署名を用いたプログラムの改ざん検知の手順は、次のようになる。

1.プログラム開発者は、プログラムを作成する。

2.開発者は、自分の秘密鍵を用いて、プログラムに対して署名処理を行う。

3.開発者は、署名が付与されたプログラムを公開する。

4.プログラム利用者は、署名が付与されたプログラムをダウンロードする。

5.利用者は、プログラムの署名を検証する。

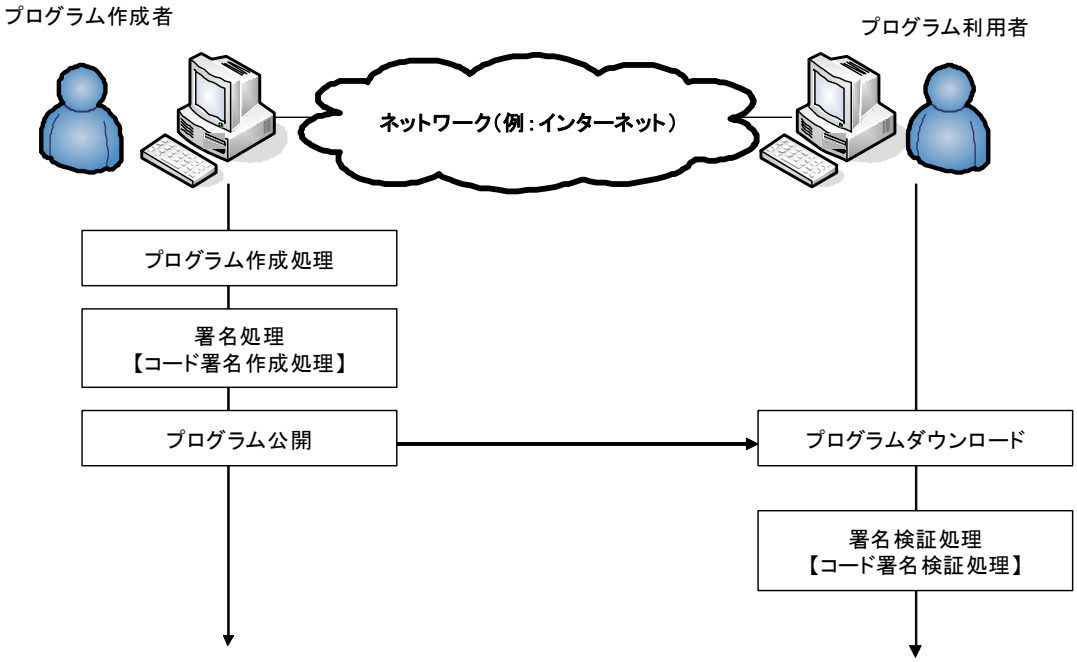



図 6　コード署名による改ざん検知手順例

コード署名における暗号利用方法は、以下の通りである。

(ア)コード署名作成

プログラム開発者は、以下の手順によりプログラムに対するコード署名を作成する。

・ プログラム開発者は、ハッシュ関数を用いてプログラムのハッシュ値を求める。

・ そのハッシュ値を公開鍵暗号方式の秘密鍵を用いて、署名生成処理を行う。

（イ）コード署名検証

プログラム利用者は、以下の手順でコード署名を検証する。

・ トラストアンカとなる公開鍵証明書（例：認証局のルート CA 証明書）を基点として、開発者の公開鍵証明書を検証する。

・ ハッシュ関数を用いてプログラムのハッシュ値を求める。

・ 開発者の公開鍵証明書に含まれた公開鍵を用いて、署名検証処理を行う。この際、データフォーマットの全ビットの比較によりチェックを行う。

### （イ）備えるべき条件

PKI 環境が存在することが前提である。コード署名の利用者は、コード署名に利用する公開鍵/秘密鍵のペア及び公開鍵証明書を所有している。また、コード署名の検証者は、コード署名を検証するための公開鍵証明書を保有している。また、認証局が公開する CP/CPS に基づいて運用が行われていることも必要である。

### （ウ）標準化動向

現状、標準化されたコード署名技術は存在しない。

### （エ）暗号アルゴリズムの安全性とコード署名の安全性

Microsoft 社の Authenticode は、PKCS#7 の技術が利用されている。そのため、Authenticode で使用されている暗号技術は、「公開鍵暗号」と「ハッシュ関数」となる。Authenticode の安全性は、これらの暗号技術の安全性に依存することになる。また、Java コード署名も同様に、「公開鍵暗号」と「ハッシュ関数」に依存する。

脆弱性を持つ公開鍵暗号、あるいは、ハッシュ関数が利用されると、攻撃者が都合良く改ざんしたプログラムを改ざんされたものではないと誤判定する可能性がある。そのため、安全性が確認された公開鍵暗号、あるいは、ハッシュ関数を使用することが重要である。また、使用している公開鍵暗号、あるいは、ハッシュ関数に脆弱性が見つかった場合は、速やかに、安全な公開鍵暗号、あるいは、ハッシュ関数に切り替えることが必要である。

### （オ）推奨される利用方法

文献[1][2]を踏まえると、Authenticode や Java コード署名における推奨暗号アルゴリズム及びパラメータは、以下の通りである。

**表 7　公開鍵証明書の発行に利用する推奨デジタル署名アルゴリズム（※）**

| 分類 | アルゴリズム | パラメータ |
|---|---|---|
| 公開鍵暗号 | RSASSA-PKCS1-v_1_5,<br>RSA-PSS | 鍵長 2048 ビット以上 |
| | DSA | 鍵長 2048 ビット以上 |
| | ECDSA | 鍵長 224 ビット以上 |
| ハッシュ関数 | SHA-2 シリーズ | SHA-256/SHA-384/SHA-512 |

※認証局が使用すべき推奨である。

**表 8　プログラムへの署名に利用する推奨デジタル署名アルゴリズム**

| 分類 | アルゴリズム | パラメータ |
|---|---|---|
| 公開鍵暗号 | RSASSA-PKCS1-v_1_5,<br>RSA-PSS | 鍵長 2048 ビット以上 |
| | DSA | 鍵長 2048 ビット以上 |
| | ECDSA | 鍵長 224 ビット以上 |
| ハッシュ関数 | SHA-2 シリーズ | SHA-256/SHA-384/SHA-512 |

## 5.2. 時刻保証技術

### 5.2.1.実現するセキュリティ

時刻保証技術とは、電子データの存在時刻を保証する機能を提供する。

### 5.2.2.想定される脅威

想定される脅威は、攻撃者による故意、あるいは、不注意による電子データの存在時刻の改ざんや電子データに含まれる時刻情報を改ざんすることである。

### 5.2.3.対策方針

#### (1)必須対策方針

電子データに対して、存在時刻を保証する情報を付加することにより、電子データの存在時刻を保証する。

#### (2)推奨対策方針

存在時刻を保証する情報の改ざんを検知するために、この情報の改ざんを検知できるようにする。

### 5.2.4. 時刻保証で利用される対策技術

#### (1)時刻認証

##### (ア)技術概要

時刻認証技術とは、電子データの存在時刻の証明とその時刻以降の非改ざん性を証明する機能を提供する。ここでは、実用サービスとして普及しているISO/IEC 18014-2、あるいは、RFC 3161で規格化されているデジタル署名を用いた時刻認証技術について説明する。

(a)時刻認証による時刻保証手順例

タイムスタンプ局と呼ばれるサービス事業者が、時刻認証のためのタイムスタンプサービスを提供する。タイムスタンプ利用者は、このサービスを利用し、電子データの時刻保証を実現する。

1. タイムスタンプ利用者は、ハッシュ関数を用いて、電子データのハッシュ値を求める。
2. タイムスタンプ利用者は、ハッシュ値をネットワーク上のタイムスタンプ局へ送信する。
3. タイムスタンプ局は、受信したハッシュ値に対してタイムスタンプ局が管理する信頼性の高い時刻情報を結合し、タイムスタンプ対象データを作成する。
4. タイムスタンプ局は、タイムスタンプ対象データに対してデジタル署名を付与し、タイムスタンプを作成する。

4. タイムスタンプ局は、作成したタイムスタンプをタイムスタンプ利用者に送信する。
5. タイムスタンプ利用者は、受信したタイムスタンプと元の電子データを関連付けて保管する。
6. タイムスタンプ利用者は、後日に、電子データの存在時刻及び非改ざん性を確認するために、タイムスタンプ検証を行う。

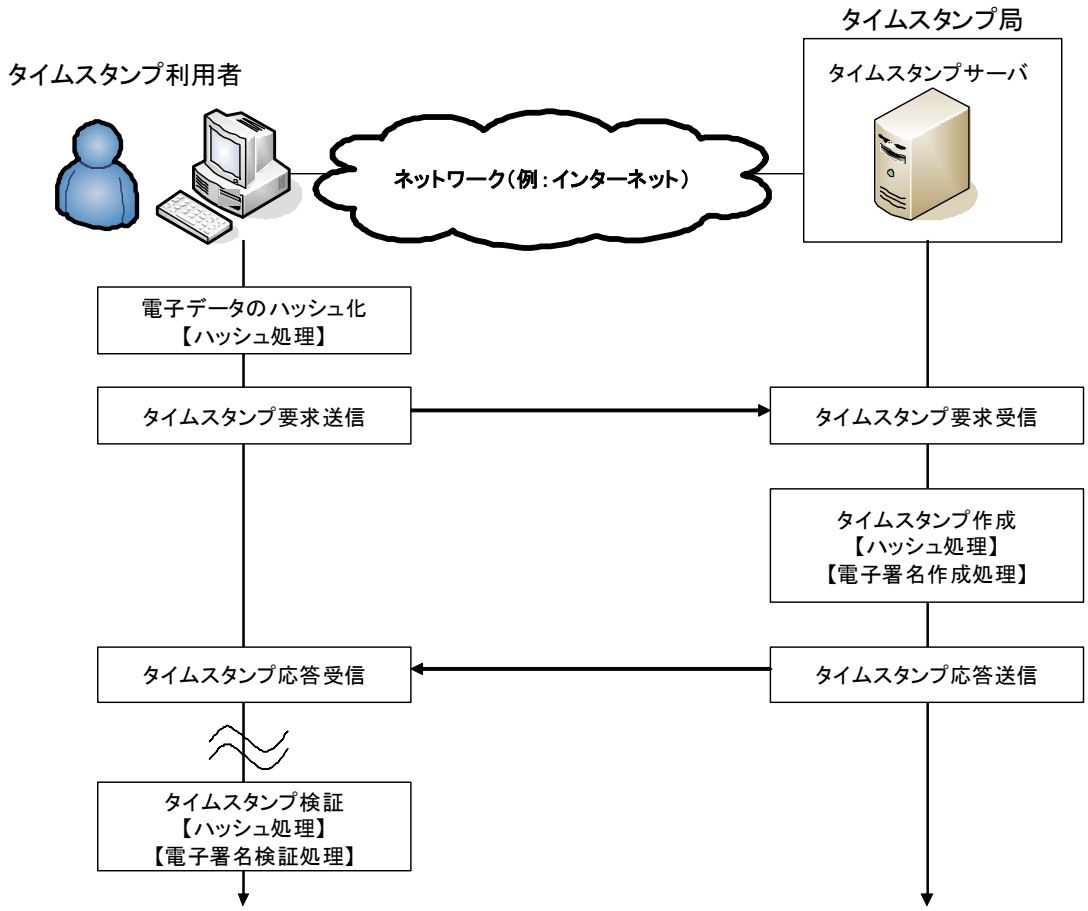



図 7　時刻認証による時刻保証手順例

時刻認証における暗号利用方法は、以下の通りである。

### (i)電子データのハッシュ値作成

タイムスタンプ利用者は、ハッシュ関数を用いて電子データのハッシュ値を求める。

### (ii)タイムスタンプ作成

タイムスタンプ局は、以下の手順によりタイムスタンプの署名を作成する。

・タイムスタンプ局は、ハッシュ関数を用いて、タイムスタンプ対象データ（タイムスタンプ利用者から受信したハッシュ値、時刻情報、などから構成される）のハッシュ値を求

める。
・ そのハッシュ値を公開鍵暗号方式の秘密鍵で署名生成処理を行う。

### (iii)タイムスタンプ検証

タイムスタンプ利用者は、以下の手順でタイムスタンプを検証する。
・ トラストアンカとなる公開鍵証明書（例：認証局のルート CA 証明書）を基点として、タイムスタンプ局の公開鍵証明書を検証する。
・ ハッシュ関数を用いてタイムスタンプに含まれたタイムスタンプ対象データのハッシュ値を求める。
・ タイムスタンプ局の公開鍵証明書に含まれた公開鍵を用いて、署名検証処理を行う。この際、データフォーマットの全ビットの比較によりチェックを行う。
・ 次に、ハッシュ関数を用いて、タイムスタンプと関係付けられた電子データのハッシュ値を求める。
・ 求めたハッシュ値と検証済みのタイムスタンプに含まれた該当ハッシュ値（電子データのハッシュ値に該当）を比較する。同一の値であれば、タイムスタンプの中に含まれた時刻時点から、電子データの改ざんが無いこと分かる。

また、時刻保証のサービスとして、電子公証制度に基づく電子確定日付の付与のサービスが知られているが、同サービスが上記と同様の方式を採用されているか否かは定かではない。仮に類似の方式を採用している場合には、適切な暗号アルゴリズムの実装を検討するうえで、本節の記述を参考にすることができると考えられる。

### （イ）備えるべき条件

PKI 環境が存在することが前提である。タイムスタンプ局は、タイムスタンプ作成に利用する公開鍵/秘密鍵のペア及び公開鍵証明書を所有している。タイムスタンプ検証者は、タイムスタンプ検証に必要な公開鍵証明書を保持している。また、認証局が公開する CP/CPS に基づいて運用が行われていることも必要である。

### （ウ）標準化動向

時刻認証技術に関しては、ISO/IEC 18014 シリーズとして国際規格となっている。また、IETF では、RFC 3161 として標準化されている。

### （エ）暗号アルゴリズムの安全性と時刻認証の安全性との関係

ISO/IEC 18014-2:2002 においてデジタル署名技術を利用した独立トークン方式のタイムスタンプ技術では、暗号技術として、「公開鍵関数」と「ハッシュ関数」が利用されている。このタイムスタンプ技術の安全性は、これらの暗号技術の安全性に依存する。

脆弱性を持つ公開鍵暗号、あるいは、ハッシュ関数が利用されると、攻撃者が都合良く改ざんした電子データを改ざんされたものではないと誤判定する可能性がある。また、その改ざんされたデータの存在時刻を誤判定（例：あたかも過去に存在したかのように誤判定）する可能性がある。そのため、安全性が確認された公開鍵暗号、あるいは、ハッシュ関数を使用することが重要である。また、使用している公開鍵暗号、あるいは、ハッシュ関数に脆弱性が見つかった場合は、速やかに、安全な公開鍵暗号、あるいは、ハッシュ関数に切り替えることが必要である。

### （オ）推奨される利用方法

文献[1][2]を踏まえると、時刻認証技術での推奨暗号アルゴリズム及びパラメータは、以下の通りである。

**表 9　公開鍵証明書の発行に利用する推奨デジタル署名アルゴリズム（※）**

| 分類 | アルゴリズム | パラメータ |
|---|---|---|
| 公開鍵暗号 | RSASSA-PKCS1-v_1_5,<br>RSA-PSS | 鍵長 2048 ビット以上 |
| | DSA | 鍵長 2048 ビット以上 |
| | ECDSA | 鍵長 224 ビット以上 |
| ハッシュ関数 | SHA-2 シリーズ | SHA-256/SHA-384/SHA-512 |

※認証局が使用すべき推奨である。

**表 10　タイムスタンプ対象データのハッシュ値生成時に利用する推奨ハッシュ関数**

| 分類 | アルゴリズム | パラメータ |
|---|---|---|
| ハッシュ関数 | SHA-2 シリーズ | SHA-256/SHA-384/SHA-512 |

**表 11　タイムスタンプに利用する推奨デジタル署名アルゴリズム**

| 分類 | アルゴリズム | パラメータ |
|---|---|---|
| 公開鍵暗号 | RSASSA-PKCS1-v_1_5,<br>RSA-PSS | 鍵長 2048 ビット以上 |
| | DSA | 鍵長 2048 ビット以上 |
| | ECDSA | 鍵長 224 ビット以上 |
| ハッシュ関数 | SHA-2 シリーズ | SHA-256/SHA-384/SHA-512 |

参考文献

[1] 総務省、経済産業省、“電子政府推奨暗号リスト”、平成 15 年 2 月 20 日、available at http://www.cryptrec.jp/images/cryptrec_01.pdf

[2] 独立行政法人 情報通信研究機構、独立行政法人 情報処理推進機構、” CRYPTREC Report 2006 “、平成 19 年、3 月、available at http://www2.nict.go.jp/y/y213/cryptrec_publicity/c06_wat_final.pdf

[3] 独立行政法人 情報通信研究機構、独立行政法人 情報処理推進機構、” CRYPTREC Report 2005 “、平成 18 年、3 月、available at http://www2.nict.go.jp/y/y213/cryptrec_publicity/c05_wat_v2.pdf

# 6.PKI (Public Key Infrastructure)

## 6.1.公開鍵の証明

政府機関等から公表されるデジタル文書の正当性を示すために、デジタル署名技術が利用される。デジタル署名技術では、デジタル文書の発行者が秘密に保有する署名生成鍵（秘密鍵）と、デジタル署名データを用いて当該デジタル文書に改ざん等がないことを確認するための署名検証鍵（公開鍵）が用いられる。署名検証鍵は広く公開されるが、その所有者、有効期限等は公開鍵証明書を用いて行われる。公開鍵証明書の発行は、認証局（Certification Authority: CA）が行う。公開鍵証明書は、CA の署名生成鍵で署名されており、公開されている CA の署名検証鍵を用いて検証することで、その公開鍵証明書の正当性が確認でき、署名検証鍵の所有者、有効期限を確認できることとなる。

また、秘密に保持すべき署名生成鍵が、署名方式の危殆化等の理由により漏洩する場合も考えられる。このため、公開鍵証明書が現在有効であるかどうかを随時確認する必要があり、この機能を実現するのが、証明書失効リストおよびOCSP（Online Certification Status Protocol）である。

### 6.1.1.実現するセキュリティ機能

#### (1)公開鍵証明書

公開鍵であるデジタル署名の署名検証鍵の正当性を確認する。

#### (2)証明書失効リスト

当該 CA が発行した公開鍵証明書の中で、失効した公開鍵証明書のリストであり、広く公開することで、公開鍵証明書の広く周知することができ、これにより利用者は公開鍵証明書の有効性を確認したうえで署名検証鍵を利用することができる。公開鍵証明書の失効の理由としては、有効期限切れ、ならびに、CA を含む署名検証鍵の漏洩等があり、特に後者は、公開鍵証明書には記載不可能な内容である。

### 6.1.2.想定される脅威

#### (1)公開鍵証明書

公開鍵証明書が偽造された場合、悪意ある第３者の成りすましが可能となる。また、特に CA の公開鍵証明書が偽造された場合には、当該 CA が発行した公開鍵証明書によって提供されたユーザの公開鍵証明書の信頼性も損なわれることとなる。

#### (2)証明書失効リスト

証明書失効リストが偽造された場合、本来は有効な証明書が無効と判断されたり、本来は無効な証明書が有効であると判断されたりする可能性が生じ、公開鍵証明書の有効性が担

保できなくなる。

### 6.1.3. 対策方針

#### (1)必須対策方針

· MD2, MD5 といった危殆化したハッシュ関数を使用せず、SHA-1 を利用する。

· RSA 署名、DSA 署名においては 1024bit 以上の鍵を使用し、ECDSA は 160bit 以上の鍵を使用する。

#### (2)推奨対策方針

· システムで使用する装置において対応できる限り、ハッシュ関数として、SHA-256/384/512 を利用する。

· RSA 署名、DSA 署名においては、2048bit 以上の鍵を使用し、ECDSA は 224bit 以上の鍵を使用する。

### 6.1.4. . 公開鍵の証明で利用される対策技術

#### (1)X.509 v03

**(ア)技術内容**

X.509 では、公開鍵証明書に記載する内容（プロファイル）を定めている。X.509v03 で規定されたプロファイルを図 1 1 に示す。証明書に示される signatureAlgorithm（署名アルゴリズム）および signatureValue（署名値）により、証明書に記載された内容の正当性を確認する。

| 領域名 | 説明 |
|---|---|
| tbsCertificate<br>(署名前証明書) | 証明書の基本的な情報と公開鍵を示す。 |
| version<br>(バージョン) | X.509 証明書のバージョン |
| serialNumber<br>(シリアル番号) | 証明書を一意に識別するための番号。発行者(CA)が割り当てる。 |
| signature<br>(アルゴリズム識別子) | 発行者が証明書に署名する際に用いるアルゴリズム。OID で指定する。 |
| issuer<br>(発行者) | 証明書を発行した機関(CA) の名前。X.500 識別名 (DN) において記述。 |
| validity<br>(有効期間) | 証明書の有効期間。 |
| notBefore<br>(開始時刻) | 証明書が有効になる時刻。 |
| notAfter<br>(終了時刻) | 証明書が無効になる時刻。 |
| subject | 証明書の所有者の名前。X.500 識別名 (DN) において記述。 |

| | |
|---|---|
| （主体者） | |
| subjectPublicKeyInfo<br>（主体者公開鍵情報） | 証明書所有者（主体者）の公開鍵に関する情報 |
| algorithm<br>（アルゴリズム） | 公開鍵のアルゴリズム名。OID で指定。 |
| subjectPublicKey<br>（主体者公開鍵） | 証明書所有者が所有している公開鍵 |
| subjectUniqueID<br>（主体者ユニーク識別子） | 主体者名を再利用した際に、主体者を識別するために使用。<br>省略可能。 |
| extensions<br>（拡張領域） | 証明書の拡張領域。 |
| signatureAlgorithm<br>（署名アルゴリズム） | 発行者が証明書に署名する際のアルゴリズム。OID で指定 |
| signatureValue<br>（署名値） | 発行者のデジタル署名値。 |

http://www.ipa.go.jp/security/pki/033.html　表３－４を基に作成

図 １１X.509 プロファイル

### （イ）備えるべき要件

検証者が利用可能な暗号プリミティブ及び鍵長を選択する。

### （ウ）標準化動向

X.509v02 は RFC 2459 で仕様が定められているが、RFC 3280 にて X.509 v03 の仕様が定められると共に、廃案となっている。

### （エ）X.509v03 の安全性と暗号アルゴリズムの安全性の関係

・　　発行される証明書の信頼性は、使用されるデジタル署名方式の安全性に依存する。

・　　従って、十分安全とされる鍵長、ならびに、ハッシュ関数を利用することにより、証明書の安全性は担保される。

### （オ）推奨される利用方法

| 利用目的 | 規格で定義される利用可能暗号[2] | 推奨される CRYPTREC 暗号とセキュリティパラメータ |
|---|---|---|
| 署名 | RSASSA-PKCS1<br>DSA[3]<br>ECDSA[4] | RSASSA-PKCS1_v1_5 （2048bit 以上）<br>RSA-PSS（2048bit 以上）<br>DSA（2048bit 以上）<br>ECDSA（224bit 以上） |
| ハッシュ関数 | SHA-1<br>MD2[5]<br>MD5 | SHA-256<br>SHA-384<br>SHA-512 |

## (2) CRL（Certification Revocation List）

### （ア）技術内容

失効された証明書の一覧が、CRL(証明書失効リスト)である。検証者は受け取った証明書が現在有効であるかどうかを CRL で確認する。CRL は、証明書を発行した CA が運用ポリシーに則り、即時的に、また定期的な周期で発行される。

CRL には、その正当性を示すために発行者（CA）の署名が付されており、検証者はこれを確認することで CRL の偽造の有無を確認することができる。

領域名　説明

tbsCertList

(署名前証明書リスト)　　CRL の署名対象部分。

　　version

(バージョン)　　CRL のバージョン。

　　signature

(署名アルゴリズム)　　CRL 発行者の署名アルゴリズム OID。

　　issuer

(発行者) CRL 発行者識別名。

　　thisUpdate

(今回更新日時)　CRL の更新日時。

　　nextUpdate

---

[2] W.Polk, R.Housley, L.Bassham, Algorithm and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Rivocation List (CRL) Profile, RFC 3279, April 2002.

[3] ハッシュ関数は SHA-1 を使用

[4] ハッシュ関数は SHA-1 を使用

[5] Privacy-Enhanced Mail で利用

(次回更新日時)　次回の CRL 更新日時。

revokedCertificates

(失効証明書のリスト)　失効された証明書の一覧。

crlExtensions

(CRL 拡張)　CRL の拡張領域

signatureAlgorithm

(署名アルゴリズム)　CRL 発行者の署名アルゴリズム。OID で表記。

signature

(署名)　CRL 発行者の署名値。

http://www.ipa.go.jp/security/pki/042.html　表 4－1 を基に作成

図 1 2 CRL フォーマット

### (イ)備えるべき要件

検証者が利用可能な暗号プリミティブ及び鍵長を選択する。

### (ウ)標準化動向

X.509v02 は RFC 2459 で仕様が定められているが、RFC 3280 にて X.509 v03 の仕様が定められると共に、廃案となっている。

### (エ)CRL の安全性と暗号アルゴリズムの安全性の関係

· CRL の信頼性は、使用されるデジタル署名方式の安全性に依存する。

· 従って、十分安全とされる鍵長、ならびに、ハッシュ関数を利用することにより、CRL の信頼性は担保される。

### (オ)推奨される利用方法

| 利用目的 | 規格で定義される利用可能暗号[6] | 推奨される CRYPTREC 暗号とセキュリティパラメータ |
|---|---|---|
| 署名 | RSASSA-PKCS#1<br>DSA[7]<br>ECDSA[8] | RSASSA-PKCS1_v1_5 （2048bit 以上）<br>RSA-PSS（2048bit 以上）<br>DSA (2048bit 以上)<br>ECDSA (224bit 以上) |
| ハッシュ関数 | SHA-1<br>MD2<br>MD5 | SHA-256<br>SHA-384<br>SHA-512 |

[6] W.Polk, R.Housley, L.Bassham, Algorithm and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Rivocation List (CRL) Profile, RFC 3279, April 2002.

[7] ハッシュ関数は SHA-1 を使用

[8] ハッシュ関数は SHA-1 を使用

## (3) OCSP（Online Certificate Status Protocol）

### （ア）技術内容

OCSP は、オンラインで証明書の失効情報を確認するためのプロトコルである。証明書の失効情報を知りたいクライアント（OCSP リクエスタ）は、サーバ（OCSP レスポンダ）に失効情報を問い合わせる。OCSP レスポンダは、OCSP リクエスタの問い合わせに対して、有効(good)、失効(revoked)、不明(unknown)のいずれかを返す。OCSP レスポンダには、信頼性を付与するために CA から証明書が発行される。OCSP レスポンダは、OCSP リクエスタへの回答に署名を付与して送信する。

OCSP リクエスタの証明書失効情報の問い合わせと OCSP レスポンダの回答はリアルタイムで行われるが、必ずしも問い合わせ時点での証明書失効情報が得られるわけではない。これは、OCSP レスポンダが持つ失効情報の更新が、CA から発行される CRL の周期や、OCSP レスポンダが CA へ失効情報を問い合わせるタイミングに依存するためである。

### （イ）備えるべき要件

· OCSP クライアントが検証に利用できるデジタル署名方式ならびに鍵長を選択する。

### （ウ）標準化動向

· OCSP は RFC 2560 で規定されている。

· 大規模 PKI 向けの拡張については、RFC 5019 で規定されている。

### （エ）OCSP の安全性と暗号アルゴリズムの安全性の関係

· OCSP を用いて取得される証明書失効情報の信頼性は、使用されるデジタル署名方式の安全性に依存する。

· 従って、十分安全とされる鍵長、ならびに、ハッシュ関数を利用することにより、証明書失効情報の信頼性は担保される。

（オ）推奨される利用方法

| 利用目的 | 規格で定義される利用可能暗号 | 推奨される CRYPTREC 暗号とセキュリティパラメータ |
|---|---|---|
| 署名 | RSASSA-PKCS#1<br>DSA[9] | RSASSA-PKCS1_v1_5 （2048bit 以上）<br>DSA (2048bit 以上) |
| ハッシュ関数 | SHA-1<br>MD2<br>MD5 | SHA-256<br>SHA-384<br>SHA-512 |

[9] ハッシュ関数は SHA-1 を使用

# 7. 暗号の利用場面と暗号鍵管理

## 7.1. はじめに

### 7.1.1. 本ガイドラインの目的

情報システムの構築・運用にあたっては、セキュリティ確保のために随所で暗号技術が活用されている。暗号の利用で必要となる暗号鍵の管理は、システムのセキュリティ要素である機密性・可用性・完全性を維持するために重要な役割を担っており、暗号鍵の管理が疎かであればシステムのセキュリティを大きく損なう可能性がある。そのため暗号鍵の生成から廃棄までのライフサイクルを考慮した管理手法を策定・確立することは、情報セキュリティシステムを維持するために必要不可欠である。

暗号鍵の管理については参照すべき日本語資料は少なく、実際に暗号を利用する情報セキュリティシステムの構築・運用において活用可能な、あるべき暗号鍵管理を示す文書の提示は強く望まれている。

本資料は、暗号鍵管理について、特に鍵のライフサイクルに注目し要点を示した。作成にあたっては NIST SP800-57 part 1（鍵管理に関する推奨事項、改訂版）を参考とした。

本資料は完成版ではなく、今後の議論を受けた改訂を続け、内容の充実をはかることを意図している。

### 7.1.2. 想定読者

主な想定読者は、情報システムの調達／運用の担当者、およびこれらから依頼・指示されシステムの構築・運用を行う者とする。

### 7.1.3. 本ガイドラインの構成

本ガイドラインは以下の構成である。

7.2 では、鍵管理に関する全般的な記述を行う。7.3 では鍵情報のライフサイクルと、各段階の概要、鍵情報のリスクと対策について一般論を示す。

7.4 ではより具体的な暗号利用場面における暗号鍵管理を示す。PKI を取り上げ、想定したシステムモデルにおける鍵管理について、管理上の脅威と対策の方向性を示す。

## 7.2. 暗号の利用と暗号鍵管理

### 7.2.1. 暗号鍵に係る脅威

安全とされる暗号を選び利用しているにも係らず実際にセキュリティ上の問題が生じる場合の多くは、暗号鍵の運用上の不備、暗号の実装上の問題に起因した暗号鍵の取り扱いの不備であると言える。

暗号の運用上の不備について以下のような脅威を例に挙げることができる。

- 同一の暗号鍵を適切な利用期間を超えて使用し続ける
  → 暗号鍵が漏えいする機会が増える
- 秘密鍵（共通鍵やプライベート鍵）を暗号化されていない状態で人間が読み出してコピーできる状態におく。あるいは、秘密鍵を誰でもアクセス可能な記録媒体に記録する
  → 暗号鍵が漏えいする機会が増え、誰が鍵を入手したかが判らない
- 暗号鍵自体や暗号化の対象情報との関連付けを失くしてしまう
  → 対象の情報を検証できない
- 漏えい等の問題が生じた鍵を使い続けてしまう（新たな暗号化や署名を行ってしまう）
  → 被害の拡大

暗号鍵の運用上の不備に基づく脅威に対して、適切な対策を講じるためには、暗号鍵を体系的に管理する必要がある。

### 7.2.2.暗号の用途と暗号鍵

#### (1)暗号の利用場面

さまざまな電子システムにおいては、セキュリティ確保のために暗号技術が利用されている。暗号鍵は、システム内部に組み込まれ自動的に処理され、管理者やエンドユーザの目に触れることなく用いられることも多いが、一方で、パスワード、IC カードやトークン、電子的なデータといった形態でより直接的に取り扱われ、人による管理を必要とする場合もある。

以下に、暗号の利用場面の例を、特にエンドユーザやシステム管理者が直接的に暗号鍵の管理運用に係る場合に注目して説明する。

#### (2) PKI

PKI 利用システムにおいては、公開鍵証明書以外にも通信路の機密性確保のため等の用途で複数の箇所で多様な暗号鍵が用いられている。運用について尤も考慮すべき暗号鍵としては、認証局および加入者（エンドユーザ）の鍵ペアが挙げられる。これらの鍵ペアのうち、特にプライベート鍵は高い機密性を要求されるため生成、保管、廃棄等の取り扱いに注意が必要となる。

本文書では、署名用途で用いられる PKI システムについて利用モデルを作成し、鍵管理上の考慮すべき事項について整理を行った。検討の結果を 7.4 に記述する。

#### (3) 蓄積データの暗号化

ファイル、ディスクなど暗号化を行う単位はさまざまある。暗号化製品でも組織的な暗号鍵管理に対応できる製品は比較的少なく、電子的な鍵情報や鍵に対応するパスワードの管理がエンドユーザに任せられている場合も多い。

組織に属するエンドユーザが個人的に電子的な暗号化鍵情報やパスワードを管理した場合には様々な問題が置きうる。以下にその例を挙げる。

- 暗号鍵情報が紛失する。エンドユーザ本人が忘却してしまう可能性、エンドユーザの異動等により失われる可能性がある。
- 暗号鍵へのアクセスが限定される保証がないため、暗号化された情報へのアクセスが制限されていることが保証できない。暗号鍵がコピーされていない等を確実にする必要がある。
- 問題がある鍵情報やパスワードで暗号化が行われる可能性がある。同一の鍵情報で多数の暗号化が行われている場合や、弱いパスワードが使割れる場合など。
- 暗号化された情報と暗号鍵の対応関係についての情報が失われやすい。鍵情報を頻繁に更新・再設定した場合に、どの鍵がどの暗号化情報と対応するかが不明確となる

このような問題がもととなり、暗号化した情報が後に復号できない事態が生じうる。

### (4) 通信路の機密性確保

通信路の機密性確保を実現するために多用される手法としては SSL/TLS による暗号通信を挙げることができる。

### (5) 電子文書の長期保存

電子文書の典型例としては e 文書法の対象となる文書（契約書、設計図、仕様書）があげられる。

電子文書への署名に用いられる鍵の管理については、電子文書のライフサイクルとの対応を考慮する必要がある。管理上の課題としては、暗号鍵の有効期間についての考慮、暗号鍵と署名対象文書との対応付けの記録が上げられる。

### (6)パスワード管理

暗号を用いる上では暗号鍵の代わりにパスワードが用いられる場合も多くある。ユーザにより生成、更新（破棄）される場合には注意が必要である。

### 7.2.3.暗号鍵の分類

一般に暗号の用途とは、暗号の利用により実現される基本的なサービスのことをさす。用途には、機密性、完全性、認証、認可、否認防止がある。

暗号鍵は、暗号アルゴリズムの大分類、利用目的（用途）に応じ、管理内容や利用する期間等が異なる。これらを基にして、暗号鍵を分類することができる。NIST SP800-57part1 に示された暗号鍵の分類のうち、主要なものを下表に示す。[10]

表： 暗号鍵の分類（主要用途に関連するもののみ）

| 分類 | 説明 | 使用目的 |
|---|---|---|
| 署名生成鍵<br>(Private signature key) | デジタル署名の生成に用いられる、公開鍵暗号アルゴリズムの秘密鍵 | 認証<br>データ完全性<br>否認防止 |
| 署名検証鍵<br>(Public signature verification key) | デジタル署名の検証に用いられる、公開鍵暗号アルゴリズムの公開鍵 | 認証<br>データ完全性<br>否認防止 |
| 認証用秘密鍵<br>(Symmetric authentication key) | メッセージやデータの認証に用いられる共通鍵暗号アルゴリズムの秘密鍵 | 認証<br>データ完全性 |
| 認証用秘密鍵<br>(Private authentication key) | データの完全性と作成者についての保証を提供するために用いられる、公開鍵暗号アルゴリズムの秘密鍵 | 認証<br>データ完全性 |
| 認証用公開鍵<br>(Public authentication key) | データの完全性と作成者の確認のために用いられる、公開鍵暗号アルゴリズムの公開鍵 | 認証<br>データ完全性 |
| データ暗号化／復号用秘密鍵<br>（Symmetric data encryption key） | データの機密性を確保するために用いられる共通鍵暗号アルゴリズムの秘密鍵 | 機密性 |

各分類についての詳細を以下に示す。

#### (1)署名生成鍵（Private signature key）

公開鍵暗号アルゴリズムの鍵ペアの秘密鍵。

比較的長期間の有効性を持つデジタル署名の生成に用いられる。

管理を必要とする期間は、設定された鍵の有効期間、または破棄されるまでの間。

一般に、否認不可性を確保するため署名生成鍵のバックアップは作成しない。（ただし、

[10] NIST SP800-57 part1 においては暗号鍵の分類として 19 のカテゴリが示されている。ここでは利用目的に機密性、認証、データの完全性、否認防止が明示される 6 カテゴリを特に重要な暗号鍵とみなして取り上げることとした。

認証局の署名生成鍵のようにバックアップが必要とされる場合もある。署名生成鍵のバックアップを作成する場合には、バックアップは所有者自身の管理下に置くべきである。）

### (2) 署名検証鍵（Public signature verification key）

公開鍵暗号アルゴリズムの鍵ペアの公開鍵。

デジタル署名を検証するために用いられる。

管理を必要とする期間は、署名データの検証の必要が無くなるまでの間。

公開鍵証明書として複数の鍵のバックアップが作成されて用いられる。

### (3) 認証用秘密鍵（Symmetric authentication key）

共通鍵（対称鍵）暗号方式アルゴリズムの秘密鍵。

メッセージ、通信時のセッション、保存データを認証する（完全性および発信元を保証する）ために用いられる。

管理を必要とする期間は、認証が必要となる期間、または破棄されるまでの間。

鍵のバックアップを作成して用いることも可能である。

### (4) 認証用秘密鍵（Private authentication key）

公開鍵暗号アルゴリズムの鍵ペアの秘密鍵。

データの完全性、発信側エンティティ（人あるいはデバイス）の身元、メッセージ／セッション／保存データの出典について保証を提供するために用いられる。

管理を必要とする期間は、設定された鍵の有効期間、または破棄されるまでの間。

アプリケーションで必要とされる場合には鍵のバックアップが作成される。

### (5) 認証用公開鍵（Public authentication key）

公開鍵暗号アルゴリズムの鍵ペアの公開鍵。

データの完全性、発信側エンティティ（人あるいはデバイス）の身元、メッセージ／セッション／保存データの出典について確認するために用いられる。

管理期間は対象データの認証の必要がなくなるまでの間。

公開鍵証明書として複数の鍵のバックアップが作成されて用いられる。

### (6) データ暗号化／復号用秘密鍵（Symmetric data encryption key）

共通鍵暗号アルゴリズムの秘密鍵

データの機密性を確保するために用いられる。

管理期間は、鍵の有効期間またはデータの有効期間のうちいずれか長い間、もしくは鍵が破棄されるまでの間。

バックアップを作成して用いることも可能である。

### 7.2.4.暗号鍵の管理

#### (1)暗号鍵のライフサイクル

暗号鍵の管理段階（状態）およびそのライフサイクルについて一般化したものを下図に示す。

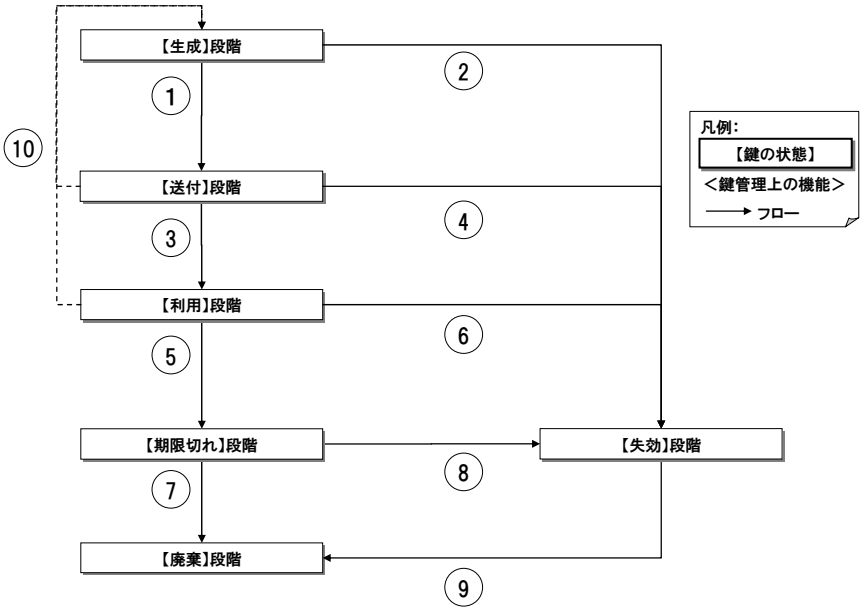



**図 7-1　暗号鍵のライフサイクル**

ここでは暗号鍵の管理段階(状態)は、【生成】、【送付】、【利用】、【期限切れ】、【失効】、【廃棄】からなるものと定義した。

管理段階間の遷移の条件を以下に示す。

① 生成され初期登録が済まされた鍵は、【生成】段階から【送付】段階に移行する。
② 【生成】段階において完全性、機密性が疑わしくなった鍵は【失効】段階に移行する。
③ 実際に利用される地点まで配送され使用可能な状態となった鍵は【送付】段階から【利用】段階に移行する。
④ 【送付】状態において完全性、機密性が疑わしくなった鍵は【失効】段階に移行する。
⑤ 有効期間が経過し、新規の署名生成や暗号化等の処理に利用しなくなった鍵は【利用】段階から【期限切れ】段階に移行する。
⑥ 【利用】段階において完全性、機密性が疑わしくなった鍵は【失効】段階に移行する。

⑦ 期限切れとなった鍵に基づいて作られた全データの利用期限が過ぎ、鍵が不要となった時点で【期限切れ】段階から【廃棄】段階に移行する。
⑧ 【期限切れ】段階において完全性、機密性が疑わしいことが明らかとなった鍵は【失効】段階に移行する。
⑨ 取消された鍵に基づいて作られた全データを利用しなくなり、鍵が不要となった時点で、鍵は【失効】段階から【廃棄】段階に移行する。
⑩ 【送付】段階あるいは【利用】段階において鍵を【失効】した場合、または【利用】段階において鍵が【期限切れ】となった場合には、旧い鍵に変わる新たな鍵を生成する。新たな鍵は【生成】段階からライフサイクルを開始する。

各管理段階における鍵に関する機能については 7.3 に述べる。また、具体的事例として PKI システムを想定した暗号鍵管理について 7.4 に述べる。

### (2) 鍵の有効期間設定

暗号鍵の有効期間とは、特定の鍵について、その鍵を扱う正当なエンティティ（人間やデバイス）による鍵の利用が許されている期間のことを指す。

一般に同一の暗号鍵を長期間にわたって利用するとセキュリティ上のリスクは高まる。以下にそれらのリスクを示す。

- 鍵の利用期間が長いほど、秘密鍵の漏洩の可能性が高まる。ミスや事故による漏えいの機会が増えるだけでなく、物理的・論理的に鍵を保護する機構を不正アクセスや脆弱性を突くことにより破ろうとする機会（時間）も増える。
- 暗号鍵を入手しようとする攻撃の機会が増え、攻撃可能な時間も延びる。
- 同一の鍵を利用し続ければ、鍵が漏洩／解読された場合の影響範囲が拡大する。
- ある暗号鍵に対して解読を試みる時間が長く取れるようになる。また、同じ鍵に基づく情報がより多く作成されるため解読に有用な情報を大量入手し易くなる。

これらを避けるために、暗号鍵には有効期間を設定し、同一の鍵の利用を制限する必要がある。

有効期間の設定には次のような副次的なメリットも期待できる。

- 管理対象とする暗号鍵、暗号処理の対象とするデータ・メッセージの総量に制限を行うこととなり、管理の実効性が高まる。
- 鍵更新の必要性が明確化され、鍵危殆化時の対応や新たな暗号アルゴリズムの切り替えを考慮したシステム構築も容易になる。

有効期間に達した暗号鍵が利用された場合には、鍵、鍵に付随する情報、あるいは鍵から作成された情報を基に、鍵が期限切れで既に無効なものであることを検出可能とする必要がある。

秘密鍵の有効期間をより短く設定すると、漏洩時の影響範囲を抑制することができるが、

反面、鍵の更新を頻繁に行う必要が生じる。現実的には手作業などを通じて鍵が漏洩するリスクが高まる可能性がある。更新頻度を上げる前提として適切な更新プロセスを実現する必要がある。

鍵が復号の対象とする暗号文や、検証対象とする署名が付された情報の有効期間がより長期にわたる場合には、有効期間に達した鍵をアーカイブする場合もある。

鍵により生成された情報が全て破棄され、以後鍵が全く使われないことが明確になった時点で鍵は破棄される。

- 秘密鍵（共通鍵方式における鍵および秘密鍵）が漏えいする可能性は、秘密鍵を用いる期間が長くなるに連れて高まる。
- 秘密鍵が漏洩した際の影響を抑えるためには、鍵の有効期間を短く設定することが望ましい。
  - 同じ秘密鍵で暗号化される情報の総量を抑え、漏えい時の影響範囲を小さくする。また、攻撃者が入手して解読に用いることが可能な情報も少なくする。
  - 物理的・論理的な保護機能に対して有効な攻撃が可能な時間を制限する。
- 秘密鍵の有効期限を短くすると、鍵の更新を頻繁に行う必要が生じる。安全上適切な更新プロセスを実現しなければ、頻繁な更新がセキュリティリスクとなりうる。

### (3) 鍵危殆化の想定

- 秘密鍵、秘密鍵が漏洩した場合には、その鍵で保護されている全ての情報について、漏えい、改ざん、偽造等が起きている可能性がある。
- 速やかに鍵情報を失効させ、影響を受けうる者に通知し、実際の影響範囲の特定と復旧作業（新たな鍵の発行等）を行う。
- 失効させる事態となる可能性が高く、影響範囲が比較的小さい鍵（例：各ユーザの鍵）については、事前に対処の方法（機能やルール）を備えておくことが有効である。
- 影響範囲が広い鍵（例：CA の署名鍵、システムで共有している暗号化や鍵配送のための秘密鍵）については、速やかに通知するとともに、システム全体としての鍵情報の再設定・交換を行う。

## 7.3.暗号鍵ライフサイクル管理

### 7.3.1.鍵の生成

#### (1)共通

- 暗号鍵は適切な暗号モジュールの内部で生成することが望ましい。
- 秘密鍵／秘密鍵は、その値を推定することが至難であるような乱数／擬似乱数処理を通じて生成されることが望ましい。
- 平文の秘密鍵／秘密鍵への直接的なアクセスはできないことが望ましい。

#### (2)公開鍵暗号方式の鍵ペアの場合

- 公開鍵暗号方式の鍵ペアは、その所有者（秘密鍵を利用するエンティティ）、認証局（CA）等の機関、あるいはこれらが協力する処理により生成される。
- 秘密鍵は人間が理解できない形式（可読性がない形式）で生成する。
- 秘密鍵の生成に用いたシードは（鍵回復を考慮しない場合には）生成後にすみやかに消去する。
- 鍵ペアの所有者が、所有者自身の否認防止に鍵ペアを用いる場合には自身で署名用途の鍵ペアを生成する必要がある。
- ユーザが鍵ペアを生成する場合には、真にそのユーザが鍵ペアを所有することを示す情報を認証機関（CA）に提供し、その情報に基づいて CA は検証を行わなければならない。（これを POP: proof of possession という）
- CA 等で集中して生成される署名用途の鍵ペアは、個々のユーザの否認防止には用いることができない。組織の認証局がある組織に属するユーザのために鍵ペアを生成する場合には、その組織としての否認防止は確保される。

#### (3)共通鍵暗号方式の秘密鍵の場合

- 共通鍵暗号方式の秘密鍵は適切な乱数生成、鍵更新、マスタ鍵からの生成により鍵生成が行われなければならない。
- 再現困難な鍵を生成する処理としては、乱数生成／擬似乱数生成があげられる。あるひとつの鍵から再現可能な処理を通じて複数の鍵を生成する処理は鍵変形（key transformation）、鍵導出（key derivation）と呼ばれる。これらの処理は初期鍵が鍵空間において予測不可能な値であれば、以後生成される全ての鍵が予測不可能であるという性質を持つことが望ましい。また、あるこの手法で作られたひとつの鍵が漏洩した際に、他の鍵を逆に導くことが不可能である性質も求められる。

### 7.3.2.鍵の送付

#### (1)共通

- 送付される鍵（および送付中に一時的に保存される鍵）は適切に保護されなければな

らない。

- 保護される対象としては、暗号サービスを確立するために配送中の鍵材料（例；機密性の提供に用いられる鍵の確立)、復旧に備えバックアップまたは記録保存される暗号化情報などがある。
- 鍵の送付は、手動（例：郵送／宅配便事業者等による配達)、自動（例：プロトコルに基づいた電子的通信)、あるいは手動と自動の組み合わせにより行われる。プロトコルによっては保護を提供する場合もある。
- 鍵材料への保護メカニズムの適用は発信側エンティティが行う。保護された状態からの鍵材料の復元・検査は受信者側エンティティが行う。
- 配送後の暗号鍵の可用性は、通信時に伝送誤り、改ざん、破壊の可能性があるため暗号学的手法を適用しても保証はできない。しかしながら経路の多重化、誤り訂正符号等の暗号以外のメカニズムによるサポートは可能である。
- 配送における暗号鍵の完全性は、改ざん防止と改ざん検出の両方に係る。配送中の鍵情報の完全性は、物理的保護が提供されるもとでの手動処理あるいは通信プロトコルに則った電子的な配送処理において、ひとつ以上の保護手法を用いて保護される。情報に対する CRC（手動のみ)、MAC、デジタル署名の適用により検出する手法、あるいは、鍵材料は目的が明確な暗号処理で用いられる際に、受信した情報で目的通りの暗号化ができない場合に鍵材料が壊れている可能性を検出する手法がある。配送後の暗号鍵の完全性に関する不具合が検出された際の対応は、暗号鍵の利用環境により異なる。不具合の処理を適切に行い、攻撃の機会としないことが必要である。この対応についてはセキュリティ方針で定義を行う。
- 配送における暗号鍵の機密性については､鍵材料の暗号化､知識分割(split knowledge)に基づく鍵材料の分割を適用することで確保する。郵送事業者等が提供する適切な物理的・手続き的な保護といった手法を 1 つ以上用いることにより保護される。
- 暗号鍵は、デバイスの変更やすり替え、受け取り先のなり済まし等の不正が行われていないと確認された際にのみ受け渡されるべきである。

### (2) 公開鍵暗号方式の鍵ペアの場合

- 署名生成に用いられる秘密鍵は、その所有者以外のエンティティに配布されてはならない。
- 公開鍵の配布においては、受信者に鍵ペアの所有者が既知であることが保証されるべきである。さらに以下についても保証されるべきである。
    - 鍵の目的／用途が判っていること
    - 公開鍵と関係するパラメタが分かっていること
    - 公開鍵が有効であること
    - 所有者が公開鍵に対応する秘密鍵を所有すること
- CA 等で集中的に鍵ペアを生成する場合には､鍵ペアは暗号モジュール内で生成した後

で、その鍵ペアの所有者だけに対して機密性を保ちつつ鍵を配送しなければならない。

### (3) 共通鍵暗号方式の秘密鍵の場合

- 共通鍵暗号方式の秘密鍵は、以下のいずれかの方法で生成・配送される。
    - 生成に続けた、手動による配送、あるいは、電子的な鍵配送（事前に配布した鍵暗号化鍵による配送等)。
    - 鍵共有方式の適用による鍵の確立
    - 鍵更新処理による鍵の決定
    - マスタ鍵からの導出
- 全ての鍵は適切な保護を与える暗号モジュール内で生成されるべきである。
- 手動での鍵の配送について
    - 鍵は暗号化されるか、適切な物理的セキュリティ手段により保護された上で輸送される。
    - 手動での鍵の配送については以下が保証されなければならない：正式に認められた発信者から鍵が発送されていること。正式に認められた受信者に鍵が受け取られること。鍵を生成するエンティティおよび鍵を受け取るエンティティの両方が信頼するエンティティがから鍵を配送すること。配送時に鍵が適切な方法に従い保護されること。
    - 鍵を暗号化して配送する場合には、鍵は、専用の鍵ラッピング鍵を用いる適切な鍵ラッピングスキーム、あるいは、受信者が持つ鍵配送用公開鍵を用いる鍵配送スキームで暗号化されなければならない。
    - 秘密分散の手法を用いる際には、各鍵コンポーネントは、各個人への輸送のために、暗号化されるか、セキュアな経路を通じて個別に配送されなければならない。機密性を要する情報として個々の鍵コンポーネントに対して適切な物理的セキュリティ手順が用いられなければならない。
- 電子的な手法での鍵の配送について
    - 鍵の配送に先立ち適切に生成され配布された、鍵暗号化鍵または鍵配送用公開鍵が必要となる。
    - 適切な鍵暗号化鍵あるいは鍵配送スキームのみを使わなければならない。これらのスキームは、鍵暗号化鍵および配送される鍵が秘密にされており改ざんされていないこと、適切に保護されていることの保証を与える。加えて、受信者が正しい鍵を得ることを保証する必要がある。
- 平文の鍵は手動で取り扱わないことが望ましい。耐タンパー性を持つハードウェアセキュリティモジュール等の適切な手段で保護した上で配送する。
- 鍵材料については、二重制御（dual control）と知識分割（split knowledge）により処理する。
- 暗号化された鍵は通信路を介して電子的に配送されうる。鍵のすり替え、改ざんに対

する防護が必要となる。

- 鍵を保護するデバイスへの鍵の配送やロードを行う際には適切な手段をお用いる。（例：手動でキーパッド等から鍵材料を直接入力する。デバイス間を有線で直接接続してのロード）

### 7.3.3. 鍵の利用

#### (1) 鍵の利用

- 通常、暗号鍵は有効期間中、常に利用可能な状態におかれる（運用の連続性が確保される）。
- 有効期間中の暗号鍵はストレージ中で適切に保護されなければならない。鍵の保管については 7.3.4 に述べる
- 暗号鍵は運用の連続性を確保するために、有効期間中に紛失や消去等で利用できなくなった場合に復元可能であることが要求される。鍵の復元（Key recovery）はバックアップあるいは鍵導出（Key derivation）に基づく。
- 以下に、鍵の運用の連続性を確保する手段として、鍵の変更および鍵導出について述べる。

#### (2) 鍵の変更

- 暗号鍵の有効期間が終了した後も運用を継承する場合は、古い鍵と新たな鍵を交換する。この交換は、暗号鍵の有効期間の継続性を維持するために、有効期間の終了が近づき、期間が終わる前に行われる。
- 新たな鍵は鍵変更（Key change）の手法により入手される。鍵の変更には鍵再作成（Re-keying）および鍵更新（Key update）の 2 つの手法がある。以下にそれらの手法について述べる。
- 古い鍵は適切に破棄されなければならない。必要でなくなった秘密鍵は漏洩するリスクを最小とするために直ちに破壊されるべきである。鍵の廃棄については 7.3.5 に述べる

#### (3) 鍵再作成

- 鍵再作成（Re-keying）とは、以前の鍵の値とは全く無関係に新しい鍵を生成する手法を指す。
- 鍵の再作成は鍵確立方式を用いて行われる。古い鍵を共有していたエンティティ間で情報の交換が必要となる。
- 鍵再作成は鍵が危殆化した際（ただし鍵確立方式が危殆化していない場合に限られる）あるいは暗号鍵の有効期間の終了が近づいている際に行われる。

#### (4) 鍵更新

- 鍵更新（Key update）とは、古い鍵の値に基づいて、新しい鍵を作る手法である。
- 鍵更新では、古い鍵に不可逆関数を適用して新しい鍵を得る手法が用いられる。古い

鍵を共有していたエンティティ間での情報の交換は不要である。

- 古い鍵が危殆化した場合には鍵更新を用いてはならない。
- 新たに作られた鍵が危殆化した場合でも（不可逆関数が用いられるため）以前の鍵は危殆化せず保護される。

#### (5) 鍵の導出

- 鍵導出（Key derivation）とは、マスタ鍵と呼ばれる秘密の値から秘密鍵（共通鍵暗号アルゴリズムの共通鍵あるいは公開鍵暗号アルゴリズムの秘密鍵）を導出する手法である。導出された秘密鍵は導出鍵と呼ばれる。
- 鍵導出では、秘密値を不可逆関数（導出関数）に入力して導出鍵を生成する。導出関数は他の導出鍵から導出鍵を推定できるものであってはならない。導出鍵の強度は導出に用いられるアルゴリズムやマスタ鍵の強度よりも大きくはならない。

### 7.3.4. 鍵の保管／バックアップ

#### (1) 鍵の保管

- 転送中ではない暗号鍵は何らかのデバイスか記録媒体に保管される（転送中の暗号鍵のコピーの場合も同様である）。保管時には適切な保護を適用しなければならない。
- 暗号鍵は、それを用いるデバイスあるいはモジュール、あるいは即座にアクセス可能な記録媒体に保存される。必要とされた際に暗号鍵がデバイスあるいはモジュールのアクティブなメモリ上に存在しない場合にはアクセス可能な記録媒体から取得される。
- 暗号鍵はアプリケーションで即座に利用可能なように保存されることがある（例：ローカルなハードディスクやサーバ）。典型的な例としては暗号モジュールや即座にアクセス可能な記憶領域（例：ローカルなハードドライブ）に置かれる鍵材料がある。
- 鍵材料は、リムーバブルメディア（例：CD-ROM）上に電子的な形態で記録される場合、リモートアクセス可能な場所に置かれる場合、紙媒体にハードコピーされ安全な場所に保管される場合がある。これらはバックアップやアーカイブの際にしばしば行われる。
- データが暗号鍵を用いて保護されている間は、暗号鍵を即座に利用可能にしておく必要がある。この保護を提供する一般的な手法としては、1 つ以上のコピーを作成し異なる地点に保管しておく方法がある。
- 暗号鍵の有効期間の間は、長期間にわたり可用性を必要とする鍵材料は運用とバックアップの各記憶装置にそれぞれ保管されるべきである。
- 全ての鍵情報には完全性の保護が必要である。改ざんからの保護、改ざんの検知、改ざんからの回復が取られる。これらは物理的手法（例：アクセス制御機能を持つ暗号モジュールや OS、他システムから独立したシステムやメディア、金庫等）、暗号学的手法（例：MAC やデジタル署名、意図した暗号処理実施による検出）、それらの組み合わせにより実現される。

- 改ざんや誤りが検出された際に鍵情報を回復させる場合には、物理的に異なる場所に鍵情報のコピーを作成し保管しておく手法をとるべきである。これらのコピーの完全性についても定期的に確認を行なうべきである。
- 秘密鍵や秘密鍵の機密性を確保するためには、鍵暗号化、暗号モジュール、アクセスが管理されたセキュアな保管庫のいずれかが用いられる。鍵暗号化鍵の回復はその鍵で暗号化された鍵の回復よりも困難とすべきである。

#### (2) 鍵のバックアップ

- 運用に用いられるストレージに置かれている現在利用可能な鍵情報のコピーを保管するためにバックアップは行われる。
- 独立したセキュアな保管用メディアへの鍵材料のバックアップは鍵回復（key recovery）のためのソースとなる。鍵回復については 7.3.6 に後述する。
- 鍵情報が漏洩する危険性を下げるためには厳格なバックアップの運用が求められる。
- 全ての鍵がバックアップを必要とするわけではない。鍵の分類とバックアップの必要性については 7.2.3 に前述した通りである。
- 組織の認証局等が鍵のバックアップを行う場合には、ユーザ個人の否認防止は実現が困難となる点に注意が必要である。

### 7.3.5. 鍵の期限切れ／失効／廃棄

- 鍵の有効期間の終了時、あるいは鍵が危殆化した際には、運用されている鍵は適切な手段で取り除かれる必要がある。
- 以下では、有効ではなくなった鍵が取り除かれる際の流れに沿って、有効期限を過ぎた鍵の扱い、鍵の失効、鍵の廃棄について説明する。

#### (1) 鍵の期限切れ

- 有効期間が過ぎた鍵は使用を停止される。処理・運用が続けられる場合は、古い鍵に変わる新たな鍵が用いられる。
- 暗号鍵のアーカイブが必要となる場合は、鍵の有効期間が終わる前にアーカイブすることが望ましい。

#### (2) 鍵の失効

- 秘密鍵の漏洩による鍵の危殆化や、暗号鍵の利用者が組織から離れることに伴う登録抹消等により、有効期間が終わる前の暗号鍵について以後の利用を停止する必要が生じうる。これを鍵の失効（key revocation）という。
- 暗号鍵の失効は、共通鍵暗号アルゴリズムの共通鍵や公開鍵暗号アルゴリズムの公開鍵が無効であることを明示するために行われる。公開鍵が無効とされた場合には対応する秘密鍵も無効となる。
- 暗号鍵の失効は、その鍵の以後の使用が継続されないことの通知により実現される。これは関係する全エンティティへの通知を送信する機構、あるいはエンティティから

の通知要求に応じる機構を伴う。

- 同じ鍵のコピーが存在する場合、バックアップが存在する場合、鍵がペアで使われる場合等を考慮し、いずれの地点に対しても通知が行われ、失効した鍵の取扱いが適切なものとする必要がある。

#### (3) 鍵の廃棄

- 不要となった暗号鍵の記録は消去される。特に秘密鍵（共通鍵および秘密鍵）は有効期間終了後に確実に消去される仕組みが必要となる。公開鍵は、保持されたままであっても消去されても良い。
- ただし、監査の目的で暗号鍵および暗号鍵に関連する情報が保持され、後に鍵の変更や危殆化を追跡するために用いられる場合もある。
- 鍵のコピーが存在する場合は、鍵危殆化の可能性を最小とするために、鍵が不要となった時点で全てのコピーを消去しなくてはならない。（アーカイブあるいは鍵回復に備えてコピーがとられている場合も同様に考慮が必要となる）
- 暗号化されていない暗号鍵を記録された媒体から消去する際には適切な物理的・電磁気学的な消去方策を講じる必要がある。通常のデータを消去する場合と同様の手法で削除しただけでは情報が完全に抹消されない可能性がある。（例えば長期間暗号鍵が記録されていた磁性体には暗号鍵のビットが焼き付けられている可能性がある）

### 7.3.6. 鍵の回復

- 多くの場合、紛失した古い鍵は失効させて新たな鍵を発行しなおすが、以前暗号化された情報や署名が付された情報の扱いが問題となる。
- 暗号鍵の紛失等により鍵を利用不可能となる事態に対処するために、バックアップを取り、鍵の回復（key recovery）を行う。鍵の回復は、暗号化された情報の復号、データの完全性の検証のために次のような事態になった際に行われる。
  - 暗号鍵の有効期間が過ぎ、ストレージ中に暗号鍵がなくなっている。
  - 暗号鍵が、システムのクラッシュや改ざんにより破損している。
  - 暗号鍵の所有者が所属する組織が、その暗号鍵の利用を求めた際に所有者が求めに応じられない。
- 暗号鍵の回復に際しては、暗号鍵自体をバックアップあるいはアーカイブから取得しデバイス、モジュール等に配送する場合、鍵導出の手法を用いて、改めて暗号鍵を再構築する場合がある。
- 鍵の回復に備えたバックアップやアーカイブの作成の有無についてはシステム個別に決定される。決定にあたっては以下の考慮が必要となる。
  - 暗号鍵の分類に基づく種別
  - 暗号鍵が用いられるアプリケーションの種別
  - 暗号鍵が保持される形態と保持者

    - 暗号鍵を用いた通信におけるエンティティの役割
    - 暗号鍵が用いられるアルゴリズムおよび計算
    - 暗号鍵により保護される情報の価値と鍵喪失時の被害の大きさ
- 暗号鍵の回復を実現するためには鍵回復に関する方針に従い、セキュアな鍵復元のシステムを確立する必要がある。このシステムは、暗号鍵の保存・復元に関する技術と設備、システム管理の手順、システムオペレータで構成される。
- 暗号鍵の回復については運用方針を事前に策定しておく必要がある。以下に関して考慮し明確化しておく。
    - 保管しておくべき暗号鍵
    - 暗号鍵を保管する手法および場所
    - 鍵回復に関連する情報の取扱い責任者
    - 鍵回復を要請可能な者および回復の条件
    - 鍵回復の方針を変更する条件および変更可能な者
    - 適切に鍵回復が行われたことを示すための監査機能
    - 長期間を経た暗号鍵とその破壊に関する対処
    - 暗号鍵が回復された際に通知を行う対象者および条件
    - 鍵回復のためのデータが危殆化された際に従うべき手順

## 7.4. PKI システムにおける暗号鍵ライフサイクル管理

本章では、ライフサイクルを考慮した暗号鍵管理の実際を具体的に示す一例として PKI における公開鍵証明書の管理とセキュリティ対策について示す。

想定する PKI の用途モデルを下の 2 つの図に示す。この例ではユーザ証明書を利用者における署名生成と検証に用いることを想定し、ユーザ証明書に係る鍵ペア、認証局（CA）の自己署名証明書に係る鍵ペアを中心に証明書の利用を示している。このモデルはあるひとつの構築手法を例示するものであり推奨例ではない。

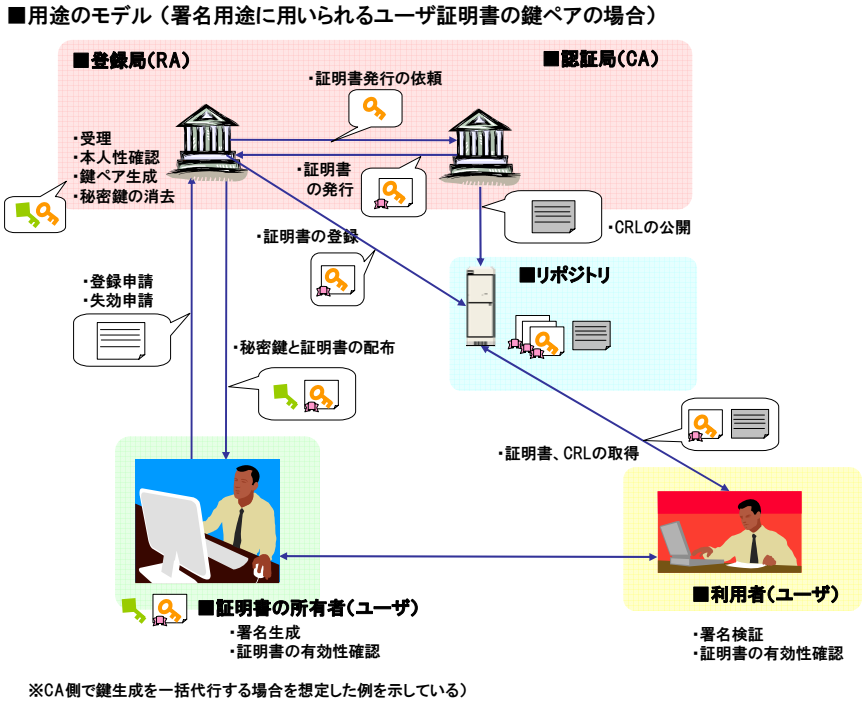



図 7-2　用途モデル（署名用途に用いられるユーザ証明書の鍵ペア）

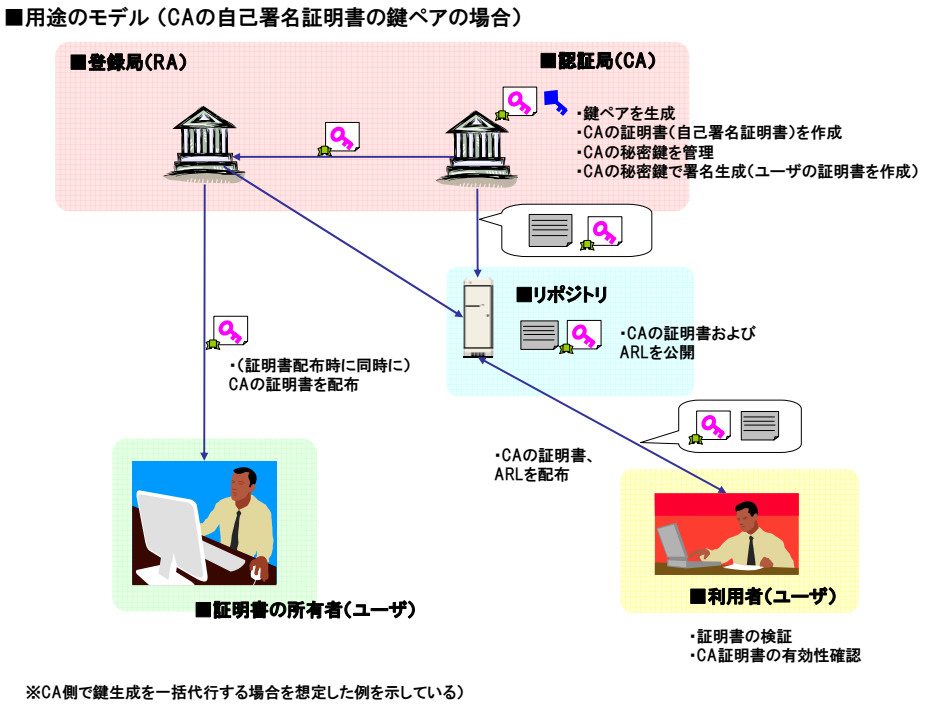



図 7-3　用途モデル（CA の自己署名証明書の鍵ペア）

以下では、このモデルに基づいて、認証局（CA）の鍵ペア（秘密鍵、公開鍵）に係る鍵管理、利用者の鍵ペアに係る鍵管理を段階毎に 4 つに分けて記述する。

### 7.4.1. 利用者の秘密鍵の管理

以下では、利用者の秘密鍵の管理について、脅威と対策の方向性をライフサイクルに沿って記述する。下図に利用者の秘密鍵のライフサイクルを示す。

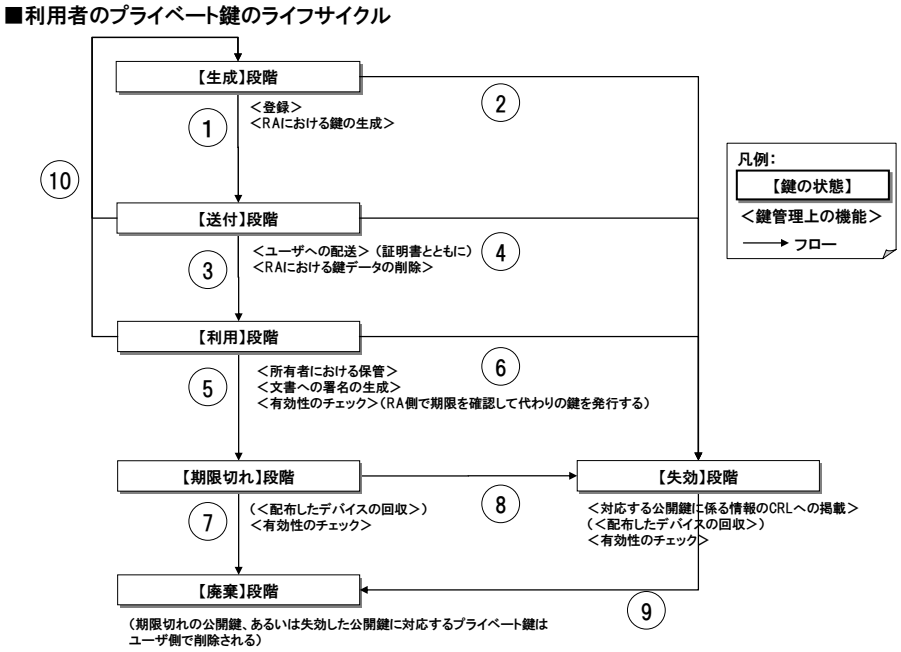



図 7-4　利用者の秘密鍵のライフサイクル

### (1) 生成段階

### (ア) 発行申請とエンティティ登録

#### (i)処理の内容

利用者は登録局（RA）に赴いて登録申請を行う。RA は登録申請を受理し、申請内容に基づいて利用者の資格および本人性の審査（確認）を行う。

| 関連する情報資産 | 種別 | ロケーション |
|---|---|---|
| 登録申請、本人確認のための情報（登録申請情報） | データ | RA |
| 証明書に記載する利用者情報（利用者情報） | データ | RA |
| 登録申請／本人確認ツール | ソフトウェア | RA |
| RA のオペレータ | エンティティ | RA |
| 利用者 | エンティティ | 利用者 |

#### (ii)想定される脅威

| 脅威（故意） | 種別 |
|---|---|
| 登録後に登録申請情報を不正に入手される | 情報の漏洩 |
| 登録後に登録申請情報が改ざんされる（あるいは削除される） | 情報の改ざん |
| 登録後に利用者情報が第三者に漏洩する | 情報の漏洩 |
| 登録後に利用者情報が改ざんされる（あるいは削除される） | 情報の改ざん |
| 登録申請／本人確認ツールが改ざんされる（例：情報を漏洩する機能の追加） | ソフトウェアの改ざん・破壊 |
| 利用者になりすまされ虚偽の申請をされる | なりすまし |

| 脅威（過失） | 種別 |
|---|---|
| 登録時に登録申請情報が誤入力される | 情報の誤入力 |
| 登録後に登録申請情報が消失する | 情報の亡失 |
| 登録後に登録申請情報を漏洩する | 情報の漏洩 |

#### (iii)対策の方針

- 厳密な本人確認を行う。

#### (iv)対策手法の例

- 本人との対面を含む手続きで登録を行う。
- HSM（Hardware Security Module）内で鍵ペアを生成し保持する。

### （イ）登録局（RA）における鍵の生成

#### (i)処理の内容

このモデルでは認証局サイドの登録局（RA）で利用者の鍵ペアを生成するものとする。

| 関連する情報資産 | 種別 | ロケーション |
|---|---|---|
| 鍵ペア | データ | RA |
| 鍵のシード | データ | 鍵ペア生成用ツール内 |
| 鍵ペア生成用ツール | ソフトウェア | RA |
| 鍵ペア/証明書保管ツール | ソフトウェア | RA |
| RA のオペレータ | エンティティ | RA |

#### (ii)想定される脅威

| 脅威（故意） | 種別 |
|---|---|
| 鍵ペアを不正に入手される | 情報の漏洩 |
| 鍵ペアが別の鍵ペアにすりかえられる | 情報の改ざん |
| 鍵ペア生成ツールが改ざんされる<br>（例：別の鍵ペアへのすりかえ、鍵ペアを漏洩する機能の追加） | ソフトウェアの改ざん・破壊 |
| 鍵シードがすりかえられる | 情報の改ざん |

| 脅威（過失） | 種別 |
|---|---|
| 鍵ペアが消失する | 情報の亡失 |
| 鍵ペアを漏洩する | 情報の漏洩 |
| 問題のある鍵シードが使用される | デザイン・運用上のミス |

#### (iii)対策の方針

- 鍵の生成に関するオペレータの不正を監視する
- 暗号学的な強度の観点からみて適切な鍵シードを用いる。

#### (iv)対策手法の例

- 複数のオペレータで協同作業をしなければ秘密鍵が発行できない
- 鍵のシードとして物理乱数生成装置を用いる
- HSM（Hardware Security Module）内で鍵ペアを生成し保持する。
- 鍵ペアの生成に関するツールを HSM 内に保持し、鍵ペアの生成を HSM 内で実行する（このツールには鍵生成に用いるシードが含まれる）。
- 鍵の生成と一時的な保存に関する一連の処理を自動化する。

### (2) 送付段階

#### (i)処理の内容

登録局（RA）は利用者に、利用者の秘密鍵を証明書とともに配送する。

利用者は登録局（RA）より得た秘密鍵を、利用者の環境において保管する。

| 関連する情報資産 | 種別 | ロケーション |
|---|---|---|
| 秘密鍵 | データ | RA、RA―利用者間、利用者 |
| 利用者の証明書 | データ | RA、RA―利用者間、利用者 |
| IC カード検証ツール | ソフトウェア | RA |
| 一時保管用ツール | ソフトウェア | RA |
| 利用者 | エンティティ | 利用者 |
| RA のオペレータ | エンティティ | RA |
| RA―IC カード間 | 通信路 | ― |
| RA―利用者間 | 通信路 | ― |

#### (ii)想定される脅威

| 脅威（故意） | 種別 |
|---|---|
| 配送の途中で秘密鍵が不正に入手される | 情報の漏洩 |
| 異なる秘密鍵と証明書、あるいは偽の秘密鍵と証明書が IC カードに格納される。 | 情報の改ざん |
| 検証ツールが改ざんされ、すりかえられた不正な IC カードに秘密鍵、証明書が格納される。 | ソフトウェアの改ざん |

| 脅威（過失） | 種別 |
|---|---|
| 秘密鍵を消失する | 情報の亡失 |
| 秘密鍵が誤入力される | 情報の誤入力 |
| 秘密鍵を漏洩する | 情報の漏洩 |

#### (iii)対策の方針

- 秘密鍵を可読性のあるデータとしては扱わない。
- 秘密鍵を誤って紛失、消去しないように適切な媒体上に記録する

#### (iv)対策手法の例

- 秘密鍵を IC カード内に保管し、IC カードの外部に直接出力できないようにする。

### (3) 利用段階

### (ウ) 利用者による秘密鍵の保管

#### (i)処理の内容

利用者は秘密鍵を保管する。

| 関連する情報資産 | 種別 | ロケーション |
|---|---|---|
| 秘密鍵 | データ | 利用者 |
| 利用者 | エンティティ | 利用者 |

#### (ii)想定される脅威

| 脅威（故意） | 種別 |
|---|---|
| 秘密鍵を他者に知られる | 情報の漏洩 |
| 秘密鍵を他者に改ざんされる | 情報の改ざん |
| 署名アプリケーションを改ざんされ処理を妨害される（例：偽の秘密鍵で署名をつける。誤った署名を出力する） | ソフトウェアの改ざん |

| 脅威（過失） | 種別 |
|---|---|
| 利用者が秘密鍵を漏洩する（例：秘密鍵を記録したディスクを不用意に他者に渡してしまう） | 情報の漏洩 |
| 利用者が秘密鍵を紛失する（例：暗号化して保存している秘密鍵のパスワードを忘れてしまう。） | 情報の亡失 |
| 利用者が秘密鍵を誤って消去する（例：ハードディスクを壊し秘密鍵を読み出せない） | 情報の亡失 |

#### (iii)対策の方針

- 秘密鍵を利用者以外にはアクセスできないよう管理可能な媒体に記録する
- 利用者が秘密鍵を誤って紛失、消去しないような媒体上に記録する

#### (iv)対策手法の例

- 秘密鍵を IC カード内に保管し、IC カードの外部には出力しない。

### （エ）署名の作成

#### (i)処理の内容

利用者は秘密鍵を用いてデジタル署名を作成する。

期限切れあるいは失効している秘密鍵では署名を行わない。

| 関連する情報資産 | 種別 | ロケーション |
|---|---|---|
| 秘密鍵 | データ | 利用者 |
| 署名アプリケーション | ソフトウェア | 利用者 |
| 利用者 | エンティティ | 利用者 |

#### (ii)想定される脅威

| 脅威（故意） | 種別 |
|---|---|
| 秘密鍵を他者に知られる | 情報の漏洩 |
| 秘密鍵を他者に改ざんされる | 情報の改ざん |
| 署名アプリケーションを改ざんされ処理を妨害される（例：偽の秘密鍵で署名をつける。誤った署名を出力する） | ソフトウェアの改ざん |

| 脅威（過失） | 種別 |
|---|---|
| 利用者が秘密鍵を漏洩する（例：秘密鍵を記録したディスクを不用意に他者に渡してしまう） | 情報の漏洩 |
| 利用者が秘密鍵を紛失する（例：暗号化して保存している秘密鍵のパスワードを忘れてしまう。） | 情報の亡失 |
| 利用者が秘密鍵を誤って消去する（例：ハードディスクを壊し秘密鍵を読み出せない） | 情報の亡失 |

#### (iii)対策の方針

- 秘密鍵を利用者以外にはアクセスできないよう管理可能な媒体に記録する

#### (iv)対策手法の例

- 秘密鍵を IC カード内に保管し、IC カードの外部には出力しない。
- 署名アプリケーションは、署名生成時に IC カード内の機能を呼び出す。

### （オ）失効状況の確認

#### (i)処理の内容

期限切れあるいは失効している秘密鍵を用いた署名は行わない。

期限のチェックについては、署名の際に秘密鍵の有効期間（＝証明書の有効期間）をチェックする

失効状況のチェックについては、署名の際に秘密鍵の失効状況（＝証明書の失効状況）を CRL に照らし合わせてチェックする。

| 関連する情報資産 | 種別 | ロケーション |
|---|---|---|
| 秘密鍵 | データ | 利用者 |
| 署名アプリケーション | ソフトウェア | 利用者 |
| 利用者自身の証明書 | データ | 利用者 |
| CA の証明書 | データ | 利用者 |
| 現在の時刻に関する情報 | データ | 利用者 |
| CRL | データ | リポジトリ、リポジトリ―利用者間、利用者 |
| 証明書有効性検証ツール | ソフトウェア | 利用者 |
| ディレクトリ・ソフトウェア | ソフトウェア | リポジトリ |
| 利用者 | エンティティ | 利用者 |
| 利用者―リポジトリ間 | 通信路 | ― |

#### (ii)想定される脅威

| 脅威（故意） | 種別 |
|---|---|
| 秘密鍵を他者に知られる | 情報の漏洩 |
| 秘密鍵を他者に改ざんされる | 情報の改ざん |
| 署名アプリケーションを改ざんされ処理を妨害される（例：偽の秘密鍵で署名をつける。誤った署名を出力する） | ソフトウェアの改ざん |
| 現在の時刻に関する情報を改ざんする | 情報の改ざん |
| 証明書有効性検証ツールを改ざんし、証明書の有効・無効について誤った判断をさせる（例：改ざんされた署名をパスさせる、期限切れの証明書をパスさせる） | ソフトウェアの改ざん |

| 脅威（過失） | 種別 |
|---|---|
| 利用者が秘密鍵を漏洩する（例：秘密鍵を記録したディスクを不用意に他者に渡してしまう） | 情報の漏洩 |
| 利用者が秘密鍵を紛失する（例：暗号化して保存している秘密鍵のパスワードを忘れてしまう。） | 情報の亡失 |
| 利用者が秘密鍵を誤って消去する（例：ハードディスクを壊し秘密鍵を読み出せない） | 情報の亡失 |
| CRL を誤って消去する | 情報の亡失 |
| 本来有効な証明書を誤って CRL に載せる | 情報の誤入力・誤更新 |

### (iii)対策の方針

- 時刻情報の改ざんを防止する
- 古い証明書の有効期限が切れる前に証明書の更新を申請する。(利用期間が重複していても構わない。)
- 証明書を利用する際にソフトウェアで有効期限を確認し利用可能かを自動的に判断する。

### (4) 期限切れ段階

利用期限が切れた秘密鍵は新たな署名を作成するために使われることはない。秘密鍵はすみやかに破棄される。

#### (i)処理の内容

| 関連する情報資産 | 種別 | ロケーション |
|---|---|---|
| 秘密鍵 | データ | 利用者 |
| 現在の時刻に関する情報 | データ | 利用者 |
| 署名アプリケーション | ソフトウェア | 利用者 |
| 利用者 | エンティティ | 利用者 |

#### (ii)想定される脅威

| 脅威（故意） | 種別 |
|---|---|
| 期限切れの秘密鍵を用いて新たな署名を作成する | その他 |
| 期限切れの秘密鍵を他者に知られる | 情報の漏洩 |

| 脅威（過失） | 種別 |
|---|---|
| 期限切れの秘密鍵を用いて新たな署名が作成される | その他 |

#### (iii)対策の方針

- 有効期間が過ぎた秘密鍵については更新を期間が切れる前に行い、すみやかに破棄する。
- 有効期間について確認をした上で有効な鍵のみで署名を行う。

#### (iv)対策手法の例

- 署名アプリケーションにおける有効期間確認および鍵更新機能の実装
- 登録局側から鍵更新を促すような手順の実装

### (5) 取消し段階

失効した秘密鍵は新たな署名を作成するために使われることはない。秘密鍵はすみやかに破棄される。

#### (i)処理の内容

| 関連する情報資産 | 種別 | ロケーション |
|---|---|---|
| 秘密鍵 | データ | 利用者 |
| 利用者自身の証明書 | データ | 利用者 |
| CRL | データ | リポジトリ、リポジトリ―利用者間、利用者 |
| 署名アプリケーション | ソフトウェア | 利用者 |
| 証明書有効性検証ツール | ソフトウェア | 利用者 |
| ディレクトリ・ソフトウェア | ソフトウェア | リポジトリ |
| 利用者 | エンティティ | 利用者 |
| 利用者―リポジトリ間 | 通信路 | ― |

#### (ii)想定される脅威

| 脅威（故意） | 種別 |
|---|---|
| 失効した秘密鍵を用いて新たな署名を作成する | その他 |
| 失効した秘密鍵を他者に知られる | 情報の漏洩 |

| 脅威（過失） | 種別 |
|---|---|
| 失効した秘密鍵を用いて新たな署名が作成される | その他 |

#### (iii)対策の方針

- 失効した秘密鍵については更新を期間が切れる前に行い、すみやかに破棄する。
- 失効状況について確認をした上で有効な鍵のみで署名を行う。

#### (iv)対策手法の例

- 署名アプリケーションにおける失効状況の確認および鍵更新機能の実装
- 登録局側から失効後に鍵更新を促すような手順の実装

### (6) 破棄段階

有効期間が過ぎた秘密鍵および失効した秘密鍵は記録した媒体から消去される。

#### (i)処理の内容

| 関連する情報資産 | 種別 | ロケーション |
|---|---|---|
| 秘密鍵 | データ | 利用者 |
| 利用者 | エンティティ | 利用者 |

#### (ii)想定される脅威

| 脅威（故意） | 種別 |
|---|---|
| 古い秘密鍵を用いて新たな署名を作成する | その他 |
| 古い秘密鍵を他者に知られる（例：秘密鍵の消去が不十分で他者が入手してしまう） | 情報の漏洩 |

| 脅威（過失） | 種別 |
|---|---|
| 古い秘密鍵を用いて新たな署名が作成される | その他 |

#### (iii)対策の方針

- 使用しなくなった古い秘密鍵はコピーを含めて確実に破棄する。

#### (iv)対策手法の例

- 鍵データを論理的・電磁気学的に確実に消去する。（例：記録媒体上に焼付けられた場合を考慮し、0、1、ランダムなビットの上書きを複数回繰り返す。

### 7.4.2. ユーザおよび認証局の鍵ペアに係るその他の鍵の管理

#### (1) 認証局の秘密鍵の管理

認証局の秘密鍵の管理について、鍵ライフサイクルを下図に示す。

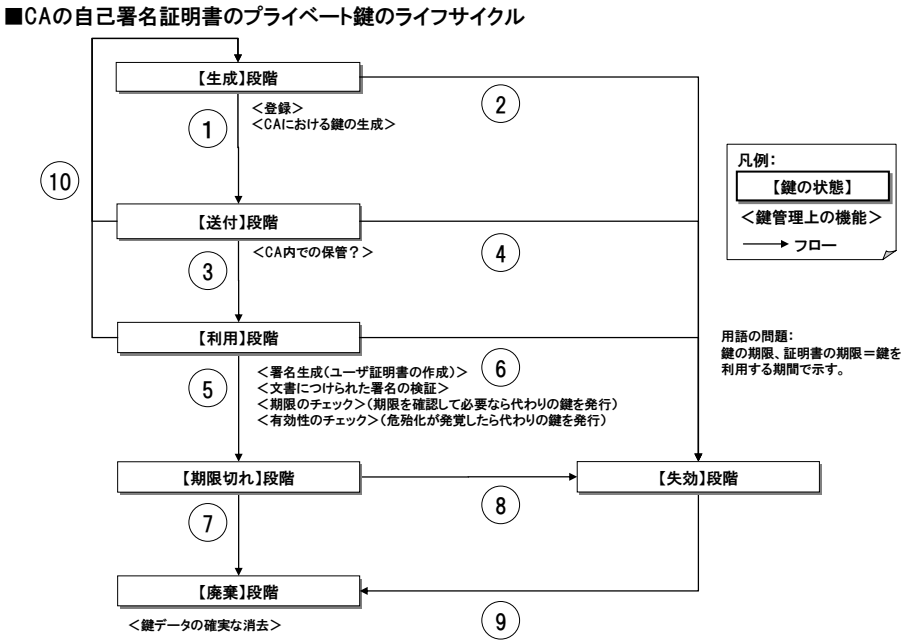



図 7-5　CA の自己署名証明書に係る秘密鍵のライフサイクル

### (2) 利用者の公開鍵の管理

利用者の公開鍵の管理について、鍵のライフサイクルを下図に示す。

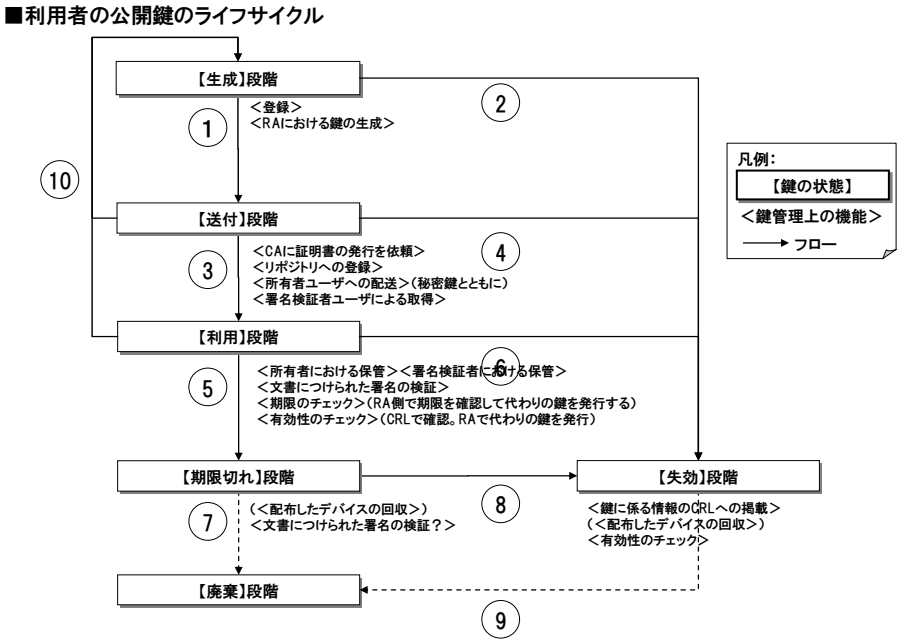



図 7-6　利用者の公開鍵のライフサイクル

### (3) 認証局の公開鍵の管理

認証局の公開鍵の管理について、鍵ライフサイクルを下図に示す。

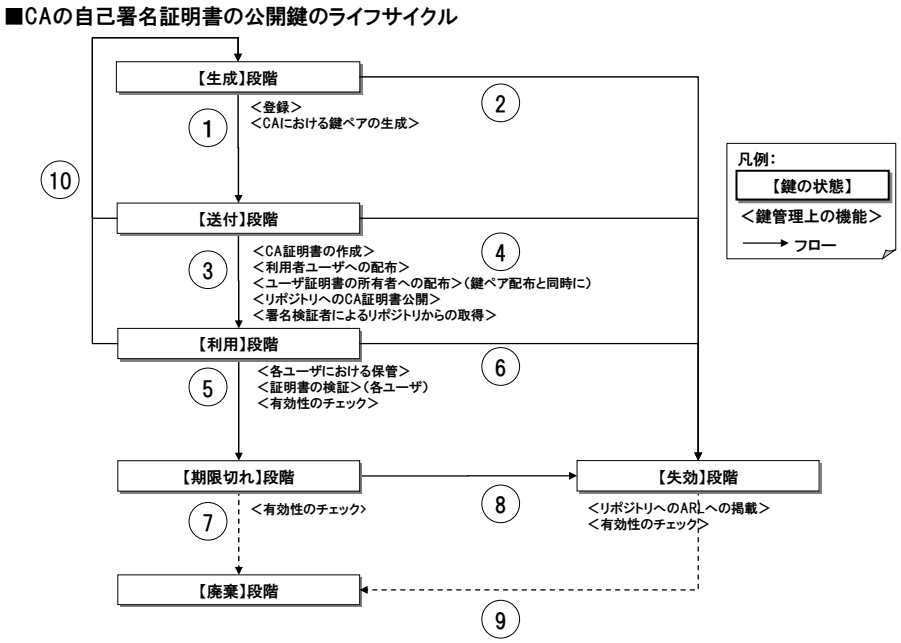



図 7-7　CA の自己署名証明書に係る公開鍵のライフサイクル

# 8. 暗号利用モード

## 8. 1. 暗号利用モードの概説

ブロック暗号は、暗号学的プリミティブの一つであり暗号化関数と復号関数により構成される。暗号化関数は、固定長の入力と鍵を受け入れ、入力を入力と同じ長さの出力に攪拌する。また、暗号化の際に利用した鍵と復号関数を用いることで出力を元の入力に戻すことができる。一般に、ブロック暗号は、入出力ペアから鍵情報を推定すること、および、鍵を知らずに出力から入力（あるいは入力から出力）を推測することが困難となるように設計されている。

しかしながら、ブロック暗号は固定長の入力しか変換できないため、長い平文を処理する際には、その利用方法を工夫する必要がある。その利用方法がブロック暗号利用モードであり、ブロック暗号利用モードには、秘匿性のみを提供するモードやメッセージ認証機能を提供するモード、更にはその両方を提供するモードなどがある。

1.1.節では、秘匿のためのブロック暗号利用モードについて概説することとする。

暗号利用モードはブロック暗号を用いて、n ビットより長いメッセージを暗号化する。送信者と受信者が秘密鍵 K を共有しており、送信者は暗号化アルゴリズム E を用いて、平文 M と鍵 K から暗号文 C = EK(M)を計算し、C を受信者に送る。M の長さは n ビットよりも長くてよい。受信者は復号アルゴリズム D を用いて、暗号文 C と鍵 K から平文 M = DK(C)を計算する。例として、ECB、CBC、OFB、CFB、CTR などがある。これらについては、下記で述べる。

歴史的に標準を紹介すると、NIST は DES を FIPS 掲載してから間もなく、DES の利用方法を定める DES 利用モードを FIPS 81 で定義した(1980 年 12 月 2 日)。

また､仕様書の誤植の変更として 1981 年 11 月 20 日に Change Notice が発行された｡FIPS 81 では、ECB、CBC、k-CFB、k-OFB の 4 つのモードが定義されている。ただし、この文書に対する Change Notice 2(1996 年 5 月 31 日)において、k-OFB に関しては k<64 では使うべきでなくこれを以降サポートしないことが記載された。Change Notice 3 は 64-bit OFB のテストベクトルのみ記載されている。

NIST は次に AES のための利用モードを定義するが、ここでは FIPS ではなく Special Publication としての発行されている。2003 年 11 月時点では、5 つの 秘匿に関する利用モードが SP 800-38A として定義されている(2001 年 12 月版)。

これには、FIPS 81 で定義した 4 つのモードに加えて、CTR モードが追加されている。また OFB モードは安全性の観点からパラメータ k はブロックサイズのみとし、末端処理の定義を付け加えている。これら方式は、FIPS 認定の任意のブロック暗号アルゴリズムに適用できると記載されている。

また、DES を定義する FIPS 46-3 でも ANSI X9.52 で定義される 7 つの利用モードの利

用を認めている。7 つとは、すべて TDEA 用であって 4 つは ECB、CBC、CFB、OFB であり、残りは ANSI X9.52 版 CBC、CFB、OFB モードである(これらはインターリービング、すなわちパイプライン処理系にも適用できるような仕様変更がなされている)。
暗号利用モードについては、いくつかの安全性定義が存在するが、ランダムビット列からの識別不能性が一般的である。暗号文 C か、もしくは C と同じ長さのランダムビット列 R が与えられ、有意な確率でこの 2 つを見分けることができないとき、安全であるという。
暗号利用モードの安全性の議論は 1990 年代から多く議論されるようになった。その大きな話題のひとつが、証明可能安全性に関する議論である。これは、内部で用いるブロック暗号を擬似ランダム関数としてモデル化しながら、暗号利用モードが提供する機能を数学的に証明するものである。以下では、利用するブロック暗号のブロックサイズが n ビットであるとして解説を行う。

## 8.2. 個別の暗号利用モード

### 8.2.1. ECB

#### (1) 技術概要

ECB(Electronic CodeBook、電子辞書)モードは、平文長が n の倍数であるような平文に対して暗号化を行なう利用モードである。
手法は、平文を n ビット毎のブロックに分割し(それぞれを Mi とする)、それぞれ独立にブロック暗号の暗号化関数の入力とする。その結果得られた出力が暗号文ブロック(Ci)となり、暗号文はそれらを接続したものである。
Ci = EncK(Mi)
この利用モードには初期値がない。平文と鍵のみから暗号文が生成される。復号はその逆関数である。
Mi = DecK(Ci)
ここで EncK(Mi)はブロック暗号の暗号化関数、DecK(Ci)は復号関数である。

#### (2) 安全性

ECB モードには以下のような欠点があるため、その特性が必要でない限り利用すべきではない。具体的には、平文がオールゼロなど、ある文字列を繰り返すものを想定すると、暗号文もあるパターンを繰り返すことになる。一般化して、同じ平文パターンは同じ暗号文パターンとして再現されるため、暗号文からそのような情報が漏洩する。
この欠点を補う方法としては、平文ブロックが衝突しない(同じ値にならない)ように圧縮を掛けたり、平文としてエントロピの高いデータを用いることなどが挙げられる。しかしながら、これらの対策も万全ではないため、できる限り他の暗号利用モードを使うべきである。

### (3)効率

平文長ｔｘｎビットに対して、ブロック暗号をｔ回呼び出すのみであり、処理効率はよい。

### (4)エラー伝播

暗号文を伝送するなどした時に発生した１ビットのエラーは該当ブロックｎビットに影響を及ぼす可能性がある。

同期ずれについては(同期のずれる幅がブロック単位である特殊な場合を除いて)、別途再同期のメカニズムが必要である。

### (5)並列処理性など

暗号化、復号ともに並列処理性と、処理順序不問性(Out-of-order)性がある。すなわち、ブロック単位でデータが入れ替わったとしても、その順序入れ替えをすることなく、到着した順序に復号処理に渡すことができる。もちろん、復号結果は、適切な順序に並べかえねば正しい平文には戻らない。

### (6)復号

復号時にも､暗号化処理と同様に並列処理性やOut-of-orderの利点がある｡復号処理には、ブロック暗号の復号関数を使う。

### (7)標準化動向

ISO/IEC 10116:2006 Information technology -- Security techniques -- Modes of operation for an n-bit block cipher

NIST SP 800-38A 800-38A、Recommendation for Block Cipher Modes of Operation - Methods and Techniques

## 8.2.2.CBC

### (1)技術概要

CBC(Cipher Block Chaining、暗号文ブロック連鎖)モードは、平文長がｎの倍数であるような平文に対して暗号化を行なう利用モードである。手法は、平文をｎビット毎のブロックに分割し(それぞれをMiとする)、中間値

Hi = Mi　Ci-1, C0 = IV

を生成したあと、それをブロック暗号の暗号化関数の入力とする。その結果得られた出力が暗号文ブロック(Ci)となり、暗号文はそれらを接続したものである。

Ci = EncK(M_i　Ci-1)

復号はその逆関数である.

Mi = DecK(Ci)　Ci-1

### (2)安全性

内部ブロック暗号をランダム関数モデルに置き換えた場合、Left-or-Right（左右平文暗号文識別）の観点から安全性が示されている。また、内部ブロック暗号を擬似ランダム関数モデルに置き換えた場合の安全性が示されている。また、詳しくは、CRYPTREC Report 2005 の付録 5 を参照のこと。

### (3)効率

平文長 t x n ビットに対して、ブロック暗号を t 回呼び出すのみであり、処理効率はよい。

### (4)エラー伝播

暗号文を伝送するなどした時に発生した 1 ビットのエラーは該当ブロック n ビットに影響を及ぼす可能性があり、次のブロックの該当部分 1 ビットが確実に反転する。

### (5)並列処理性など

暗号化には、まったくの並列処理性がない。一方、復号では、ブロック暗号処理に関する並列処理は可能である。しかしながら、平文データを復元するためには前ブロックの暗号文ブロックが必要であることを注意しなければならない。

また、ECB モードほど小さな単位では実現できないが、ある程度ブロックがまとまれば、Out-of-order 的な復号処理も可能な場合がある。すなわち、t ブロック単位でデータが入れ替わったとしても、その順序いれかえをすることなく、到着した順序に復号処理に渡すことができ、その場合、最初のブロックを除いた t-1 ブロックは正常に復号可能である。

ただし、例外的に並列処理が可能な運用もある。ANSI X3.106 や ISO/IEC 10116 では、CBC モードをインターリーブすることにより、ある程度の並列度を持たせることができる暗号方式を記載している。具体的には、独立な CBC モードを並列度数だけ飛ばしながらメッセージストリームを処理する仕様である。この場合、初期値も並列度数だけ用意せねばならず、それぞれ独立かつランダムに選択する必要がある。

### (6)復号

復号時に関する特別な注意事項はない。復号処理には、ブロック暗号の復号関数を使う。

### (7)CTS

CTS(CipherText Stealing、暗号文窃盗)モードは、IETF RFC 2040 で提案された、CBC モード向けの端数処理モードである。RFC ではバイト単位の端数処理のみが定義されているが、単純に一般化することで n ビット以上の任意のビット数のメッセージに対して処理

可能となる。
このモードはほとんどの処理が CBC モードであるので、安全性以外の主な特徴は CBC モードに準じる。
安全性については特別に議論された技術文書は見当たらないが、次のように考えることで秘匿に関する安全性は保持できていると考える。
CBC+という新しい利用モードを考える。従来の CBC モードを暗号化する場合に、n ビットの 0 パディングを行なってから CBC モードを処理する。CBC+と同様に安全であると考えられる。
ここで m x n ビット長の CBC+モードでの暗号化結果と(m-1) x n + t (1 ≦ t < n)ビット長の CTS モードでの暗号化の強度は、後者、すなわち CTS のほうが強力である。なぜならば、CTS における任意の攻撃者の振舞いは、すべて前者に対する攻撃者として再現できるからである。よって CTS は CBC と同程度に強力であると考えられる。

#### (8)標準化動向

ISO/IEC 10116:2006 Information technology -- Security techniques -- Modes of operation for an n-bit block cipher
NIST SP 800-38A 800-38A、Recommendation for Block Cipher Modes of Operation - Methods and Techniques

### 8.2.3.k-CFB

#### (1)技術概要

CFB(Cipher FeedBack、暗号文フィードバック)モードは、パラメータ k を持つブロック暗号利用モードである。平文長が k の倍数であるような平文に対して暗号化を行なう利用モードであることから、バイト単位のデータなど、データ単位長がブロック長の倍数でないような場合に用いられていた。便宜的に内部レジスタ R を考えながら処理を説明する。
k ビットの倍数長のメッセージ M は、k ビット毎のブロックに分割する(それぞれを Mi とする)。初期値 IV は R の初期値 R0 である。各ブロックでのブロック処理は、まず中間値
Hi = EncK(Ri)
を生成することから始まり、このうち上位 k ビットの値 i を、平文ブロック Mi と排他的論理和をとることで暗号文ブロック
Ci = Mi　　i
を得る。最後に R を更新する。R を上位へ k ビットシフトし、シフトの際、0 が埋められた下位 k ビットに Ci を埋め込む。よって、k=n の場合は、Ri = Ci となる。
Ci = Mi　　msbk(EncK(Ri-1))、Ri = ((Ri-1) << k)　　Ci
復号はその逆関数である.
Mi = Ci　　msbk(EncK(Ri-1))、Ri = ((Ri-1) << k)　　Ci

### (2)安全性

Left-or-Right（左右平文暗号文識別）の観点から安全性が示されている。また、内部ブロック暗号を擬似ランダム関数モデルに置き換えた場合の安全性が示されている。また、詳しくは、CRYPTREC Report 2005 の付録 5 を参照のこと。

ただし、特に k が小さい場合には、初期値に注意する必要がある。例えば、0 ばかり続く平文（もしくは 1 ばかり続く平文）を初期値 IV = 0n や IV = 1n の 1-CFB で暗号化した場合、約半分の鍵に対しては内部レジスタの更新がまったくおこなわれないため安全性に問題が生じることとなる。

### (3)効率

CFB モードは，パラメータの値に応じて処理効率が変化し、場合よっては、他のモードよりも極端に非効率的となる。

具体的には m x k ビットのメッセージを暗号化するためには m 回のブロック暗号の呼び出しを必要とする。k = n の場合、ECB や CBC と同じ程度の効率であるが、それ以外の場合、約 n/k 倍の処理量となる。

### (4)エラー伝播

1 ビットの暗号文におけるエラーにより、まず該当の平文ビットの反転が起こる。さらに該当エラーがレジスタに残る限り、平文回復ができないので、その間はエラーがおき続ける可能性がある。これは最悪、 ブロック分、エラーが起こる可能性がある。

### (5)並列処理性など

CBC モードと同様、暗号化には並列処理性がない。復号では、該当ブロックのブロック暗号処理結果は次のブロック暗号処理に直接影響しない。よって構成上はパイプライニングなど並列処理性はある。しかし、該当ブロックを処理するためには、該当ブロック以前の暗号文ブロックが必要であるので、各々のブロック暗号エンジンでこれらをバッファリングするメカニズムが必要である。

これらバッファは左右にずれているだけであるので、(並列度に応じた長い)バッファを共有することでも実現可能である。

また CBC モードと同様に、例外的に並列処理が可能な運用もある。ANSI X3.106 や ISO/IEC 10116 では、CFB をインターリーブすることにより、ある程度の並列度を持たせることができる暗号方式を記載している。具体的には、独立な CFB モードを並列度数だけ飛ばしながらメッセージストリームを処理する仕様である。この場合、初期値も並列度数だけ用意せねばならず、それぞれが安全であるためには、ランダムな nonce にする必要がある。

### (6)復号

CFB モードでは、ブロック暗号の復号関数を利用しない。よって、CFB 暗号、CFB 復号の両方の機能を実装する場合には、その実装コストは、CBC や ECB に比較して軽いことが期待できる。

### (7)自己同期性

CFB の大きな特徴として、自己同期性がある。これはブロック単位でのデータの欠損や挿入については、ある程度のエラーブロックをひきずりながらも、その後には、復号処理が回復するものである。

この機能は CBC にも一応あてはめることはできる(ただし同期パケットの境界がブロック暗号のブロック長)が、比較的大きいためこの機能が現実的に便利であることはあまりない。

CFB の場合、ブロック長を任意に設定することができるため、例えばバイト単位や、極端な例ではビット単位の同期ずれやデータ欠損挿入などにも回復する強みがある。ただし、注意を要するのは、ビット単位やバイト単位など短いデータ境界での自己同期を期待すればするほど、その処理負荷が大きくなる。これを解決したのが，OFB モードである。詳細は OFB モード参照のこと。

### (8)標準化動向

ISO/IEC 10116:2006 Information technology -- Security techniques -- Modes of operation for an n-bit block cipher

NIST SP 800-38A 800-38A、Recommendation for Block Cipher Modes of Operation - Methods and Techniques

## 8.2.4.OFB

### (1)技術概要

OFB(Output FeedBack、出力フィードバック)モードは、初期値のみに依存し逐次的に擬似乱数を生成しながら暗号化を行なう方法であり、任意のビット長の平文を処理できる。

まず、平文を n ビット毎のブロックに分割(それぞれを Mi とする)し、最後の端数の部分は端数ブロックとして扱う。初期値 IV を内部レジスタの初期値 H0 とする。Hi-1 をブロック暗号入力とし、暗号化処理の結果を Hi とする(すなわち次のブロックの内部レジスタの値にもなる)。これより暗号文ブロック Ci = Mi　 Hi を生成する。

Hi = EncK(Hi-1)、Ci = Mi　 Hi

この利用モードには初期値がない。平文と鍵のみから暗号文が生成される。復号はその逆関数である。

Hi = EncK(Hi-1)、Mi = Ci　 Hi

### (2)安全性

OFB モードに関するきちんとした安全性の証明は知られていない。しかし、ブロック暗号出力全体をそのまま入力に戻すことで内部のブロック暗号が理想的である場合、周期が約 2n-1 になることが知られている。この周期の中では乱数性の高い鍵ストリームとして利用できるため、高い安全性が期待できる。

### (3)効率

OFB は ECB や CBC と同等の処理効率で暗号化、復号処理を行なうことができる。

### (4)エラー伝播

暗号文における 1 ビットのエラーは，対応する平文ビットの反転を起こす。しかし、それ以降のエラー伝播などの影響はない。

ただし、同期ずれについては(ECB と同様、ブロック長単位の同期ずれでない限り)耐性がなく、同期ずれが起こるような場合には、別途再同期のメカニズムが必要である。

### (5)並列処理性など

暗号化、復号処理ともに、並列処理性はまったくない。

しかし、インターリービングによる並列処理が可能な運用方法が知られ、ANSI X3.106 や ISO/IEC 10116 などで記載されている。具体的には、独立な OFB モードを並列度数だけ飛ばしながらメッセージストリームを処理するものである。この場合、初期値も並列度数だけ用意せねばならず、それぞれが安全であるためには、ランダムかあるいは攻撃者に選択されないような nonce にする必要がある。

### (6)復号

OFB モードでは、ブロック暗号の復号関数を利用しない。よって、OFB 暗号化、OFB 復号の両方の機能を実装する場合には、その実装コストは、CBC や ECB に比較して軽いことが期待できる。

### (7)k-OFB

NIST FIPS PUB 81 など古い利用モードの標準化では、OFB モードを k ブロック単位で行なうことも指摘されていた。これは、k-CFB と同様、n ビット単位でないデータを扱う場合への適用を考えたものであった。しかし、k<n の場合、安全性の観点から大きな問題があることを理由に、OFB は k=n として用いるべきとなった。よってこの古い仕様は今後使われるべきでない。

安全性の懸念には二通りある。第一には、脆弱な初期値の存在である。初期値に n ビット

の 0 を与えて 1-OFB を実行すると約半数の鍵において 0 ばかり続く鍵系列が出力され、初期値に n ビット値 1 を与えて 1-OFB を実行すると約半数の鍵において 1 ばかり続く鍵系列が出力される。また、第二の安全性の懸念として、k<n の場合には、内部レジスタの更新関数が単射性を保証できなくなり、これが理由で周期の平均が 2n/2 ブロック程度となることが挙げられる。

#### (8)標準化動向

ISO/IEC 10116:2006 Information technology -- Security techniques -- Modes of operation for an n-bit block cipher

NIST SP 800-38A 800-38A、Recommendation for Block Cipher Modes of Operation - Methods and Techniques

### 8.2.5.CTR

#### (1)技術概要

CTR(カウンタ)モードは、初期値のみに依存し逐次的に擬似乱数を生成しながら暗号化を行なう方法であり、任意のビット長の平文を処理できる。まず、平文を n ビット毎のブロックに分割(それぞれを Mi とする)し、最後の端数の部分は端数ブロックとして扱う。開始値 SV を内部レジスタの初期値 R1 とする。Ri をブロック暗号入力とし、暗号化処理の結果を Hi とする。これより暗号文ブロック Ci=Mi　Hi を生成する。次のブロックでは、内部レジスタ R を整数カウンタとして 1 数えあげる。

Ci = Mi　EncK(Ri)、Ri+1 = Ri + 1

復号はその逆関数である。

Mi = Ci　EncK(Ri)、Ri+1 = Ri + 1

ここで開始値とは、特殊な運用が必要な初期値である。CTR が安全な処理モードであるために、同一の鍵が用いられている間は常に異なるブロック暗号入力を与える必要がある。CTR モードでは、内部状態の更新がカウンタであるため、システム要件から、カウンタの更新回数の限度などを知ることができる場合がある。このような情報を使いながら、うまく開始値を定義して、(同じ鍵のもとで)複数の平文を安全に暗号化できるようにする。

具体的には、ひとつのメッセージ長が 32 ブロック未満で定義されるシステムでは、下位 5 ビットをカウンタ動作部分としてリザーブしておき、残り上位 n-5 ビットをメッセージ ID として固有な数字を埋め込む。こうすることにより、最大 2n-5 個のメッセージを安全に処理できる 。

#### (2)安全性

CTR モードの安全性については、開始値が乱数の場合と、カウンタの場合との 2 種類について検討されている。

どちらの場合についても、Left-or-Right（左右平文暗号文識別）の観点からと内部ブロック暗号を擬似ランダム関数モデルに置き換えた場合の安全性が示されている。また、詳しくは、CRYPTREC Report 2005 の付録 5 を参照のこと。

### (3)効率

CTR は ECB や CBC モードとほぼ同程度に効率的である。

### (4)エラー伝播

暗号文における 1 ビットのエラーは、対応する平文ビットの反転を起こす。しかし、それ以降のエラー伝播などの影響はない。
ただし、同期ずれについては(ECB と同様、ブロック長単位の同期ずれでない限り)耐性がなく、同期ずれが起こるような場合には、別途再同期のメカニズムが必要である。

### (5)並列処理性など

暗号化復号ともに並列処理性が実現可能である。しかし、このためには、処理しているブロックが平文(もしくは暗号文)の何ブロック目であるかという情報を処理系が知っている必要がある。従って、パイプライニングなどのようなメカニズムで、メッセージ(もしくは暗号文)を最初のブロックから処理する場合には問題とはならない。
同様に、何ブロック目のデータであるかがわかれば、処理順序不問性(Out-of-order)性も達成できる。すなわち、ブロック単位でデータが入れ替わったとしても、その順序いれかえをすることなく、到着した順序に復号処理に渡すことができる。もちろん、復号結果は、到着順序に応じて並べかえねば正しい平文には戻らない。

### (6)復号

CTR モードでは、ブロック暗号の復号関数を利用しない。よって、CTR 暗号化、CTR 復号の両方の機能を実装する場合には、その実装コストは、CBC や ECB に比較して軽いことが期待できる。

### (7)標準化動向

ISO/IEC 10116:2006 Information technology -- Security techniques -- Modes of operation for an n-bit block cipher
NIST SP 800-38A 800-38A、Recommendation for Block Cipher Modes of Operation - Methods and Techniques

# 9.鍵導出関数（Key Derivation Function, KDF）

## 9.1.鍵導出関数（Key Derivation Function, KDF）

### 9.1.1.KDF 関数の概説

Diffie-Hellman 鍵共有等により 2 者の間で共有されたデータがあるとする。このデータをここでは Source Key Material（略して、SKM）と呼ぶことにする。KDF 関数は、SKM を主な入力とし、Key Material(以下、略して KM)を出力する。

KDF 関数の一般形については、Extractor と Expander とに区分してブロック化された一般形が Krawczyk 氏により提唱されている。Extractor は、暗号学的に安全ではない SKM から情報を抽出(エントロピーを濃縮)しつつ、ある特定の長さのデータ K を生成する。Expander は K を基に、任意の長さの KM を生成する。Extractor を持たない KDF 関数が多いが、Extractor を持つ場合は、Hash 関数や HMAC が使われることが多い。Expander の構造は様々であるが、PRF(Pseudo Random Function)を並べた構造がその基本である。

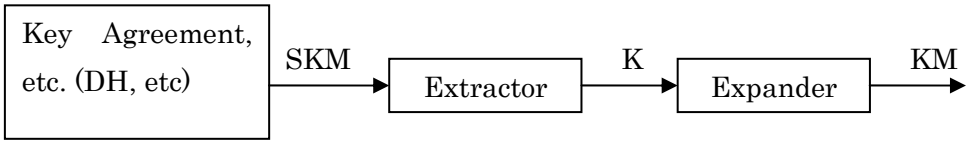



図 1 KDF 関数の入出力

9.1.1.1. KDF の構造と分類

KDF 関数の構造上の分類ポイントを以下に述べる。

#### (1) Extractor の構造

Hash 関数か、HMAC 等の Hash 関数から派生した関数を単独で適用するか、あるいは、Extractor は必要なしとしてそれを持たない KDF 関数や、Expander 中の PRF として Hash 関数等を採用することで Extractor と Expander の区別がつかない（Expander の中に Extractor の機能をするものが埋め込まれている）KDF 関数も考えられる。

実際、ANSI や NIST の仕様書において定義されている KDF 関数のほとんどは Extractor を持たない形をしている。あるいは、PRF を組み合わせた複雑な構造を持つものもある。

(a) Extractor の種類

i. (明示的な)Extractor を持たない、つまり、K = SKM

ii. Hash 関数の単独適用、つまり、K = Hash(SKM)

iii. HMAC 等、Hash 関数の派生関数の単独適用、つまり、K = HMAC(SKM,K')

iv. PRF の組合せによるより複雑な構成(例えば IKE(RFC2409))

ここで K'は鍵つきハッシュ関数の鍵とする。

### (2) Expander の構造

PRF としては、擬似乱数の性質をもつ関数であり、かつ、出力の長さが一定である必要がある。鍵を共有しようとする 2 者の間の事前ネゴシエーションにより、そのような関数であれば任意の関数が利用可能である、とする仕様もある。しかし、一般には、ベースとなる関数を一定の構造に従って組み合わせたものを PRF として使用する。

(a) PRF のベース関数

i. Hash 関数

ii. CBC 利用モード等のブロック暗号

iii. その他

(b) PRF の構造(ベース関数をどの構造に組み合わせ PRF とするか)

i. そのまま適用、例えばベース関数が Hash 関数なら PRF(x,y)=Hash(x ‖ y)など

ii. HMAC 構造

iii. NMAC 構造

iv. その他

### (3) Expander への入力

Expander への入力として、K 以外に、鍵交換の状況等をコード化した OtherInfo が一般に入力される。Extractor を迂回させた SKM や一種のパスワードを Expander へ与える場合もある。

(a) Expander への入力

i. K と OtherInfo

ii. K と SKM と OtherInfo

iii. K とパスワード pw と OtherInfo

### (4) Expander の構造

PRF をどのように組み合わせているかで分類すると以下のようになる。

(a) PRF の並列と、出力の concatenation

PRF を並列に並べ、各 PRF には K+α と counter を入力する構造。

出力を concatenation し、必要なら、上位数 octet か下位数 octet を切り落として指定の長さの KM とする

(b) PRF の feedback と、出力の concatenation

PRF の出力を feedback していく構造。

出力を concatenation し、必要なら、上位数 octet か下位数 octet を切り

落として指定の長さの KM とする。

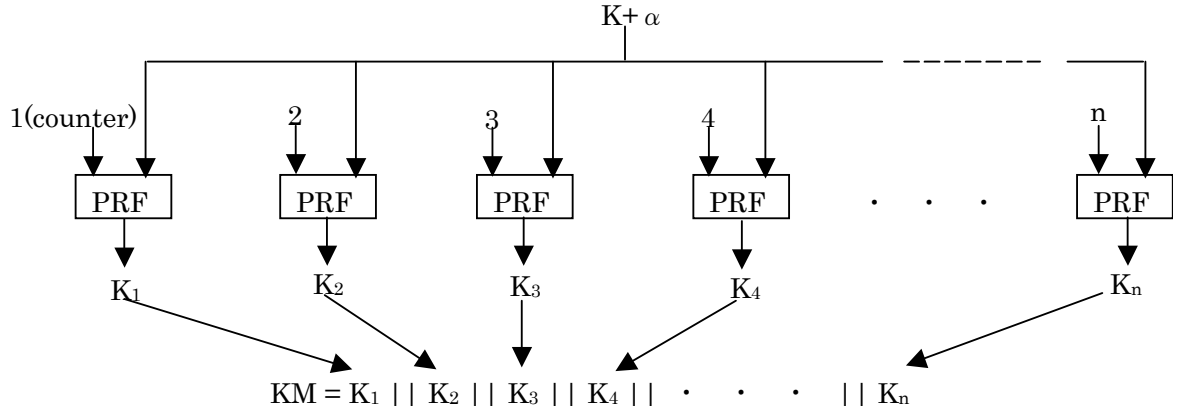



図 2.2　Expander の構造---PRF の並列構造

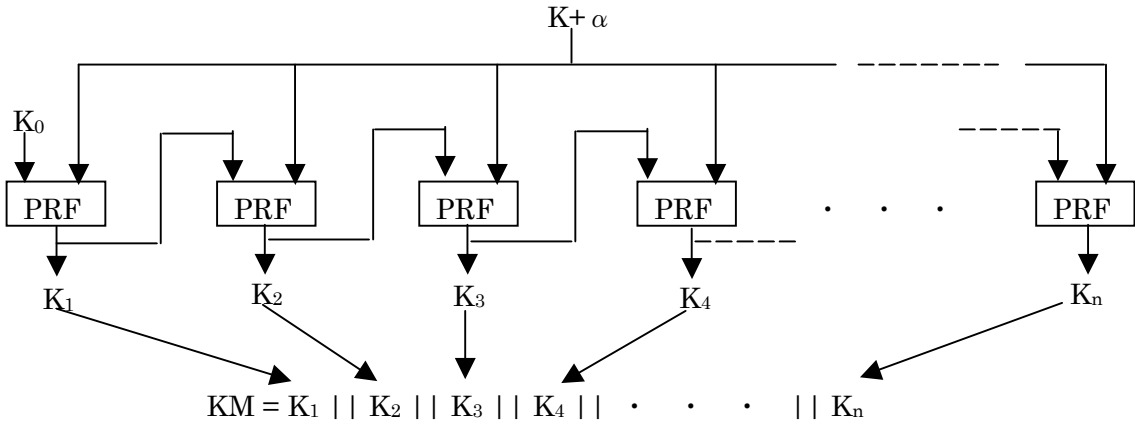



図 2.3　Expander の構造---PRF の feedback 構造

### (5) PRF への入力の構成、および OtherInfo の構成

上のブロック図において矢印で示されている(PRF への)入力項目達を使って、実際に PRF への入力を構成する際にも、様々なやり方がある。理論的に可能な構成の種類は極めて多くなるが、ここでは、比較的良くみられる構成に限って示す。尚、counter は図 2.2 における counter 値の入力、Ki は図 2.3 における feedback 値の入力を表す。

(a)　　PRF が関数として 1 引数の関数である場合で、引数を(K ‖ counter ‖ α)あるいは

(K ‖ Ki ‖ α)とする場合

(b)　同じく 1 引数の関数である場合で、引数を(counter ‖ K ‖ α) あるいは(Ki ‖ K ‖ α)とする場合

(c)　PRF が関数として 2 引数の関数である場合で、引数を (K, Ki ‖ α)とする場合

上において、1 と 2 の間に安全性上の差異は無いと考えられるが、鍵交換の 2 者の間では 1 か 2 に揃っている必要があるので、KDF 関数の実装上、重要な KDF 関数分類ポイントである。

OtherInfo は、一般に、鍵交換に関する状況や選択肢を表す文字列を連結したものであり、その内容・構成方法は仕様により様々である。内容・構成方法を定めていない仕様や、OtherInfo をオプションとしている仕様も存在する。OtherInfo は、鍵交換を行う両者の Expander 中の各 PRF 関数で共通の文字列であり、OtherInfo の内容・構成方法が安全性に影響を与えるとは考えられないが、OtherInfo の内容・構成方法が鍵交換の 2 者の間で異なれば鍵交換は当然不可能なので、KDF 関数の実装上、重要な KDF 関数分類ポイントである。

## 9.2. 個別の KDF 関数

### 9.2.1. ハッシュ関数ベースの KDF 関数

### (1) HMAC ベースの KDF 関数

| PRF のベース関数 | PRF の構造（ベース関数をどう適用するか） | KDF 仕様規定具体例（仕様名と KDF 通称）<br><br>（ー は具体例無し） | 左具体例における Expander への K 以外の入力（左図中の $\alpha$） | 左具体例における各 PRF への入力の構成 |
|---|---|---|---|---|
| ・Hash 関数 | ・そのまま適用 | ・SECG SEC 1-v.1.0<br>・ANSI X9.63-2001<br>・ANSI X9.42-2001 concatenation<br>・ANSI X9.42-2001 ASN.1<br>・ISO/IEC 18033-2 KDF1、KDF2 | ・OtherInfo | ・(K ‖ counter ‖ $\alpha$) |
| | | ・NIST SP800-56A concatenation<br>・NIST SP800-56A ASN.1 | | ・(counter ‖ K ‖ $\alpha$) |
| | | ー | ・パスワード pw と OtherInfo | |
| | ・HMAC 構造 | ー | | |
| | ・NMAC 構造 | ー | | |

| PRF のベース関数 | PRF の構造（ベース関数をどう適用するか） | KDF 仕様規定具体例（仕様名と KDF 通称）<br><br>（ー は具体例無し） | 左具体例における Expander への K 以外の入力（左図中の $\alpha$） | 左具体例における各 PRF への入力の構成 |
|---|---|---|---|---|
| ・Hash 関数 | ・そのまま適用 | ー | | |
| | ・HMAC 構造 | ・IKE(IETF RFC2409) | ・SKM と OtherInfo | ・(K, $K_i$ ‖ $\alpha$) |
| | ・NMAC 構造 | ー | | |

### 9.2.2.KDF 関数の安全性

KDF 関数への攻撃は、セッション鍵など秘密鍵を攻撃者が入手した後で行われる点に留意する必要がある。NIST SP 800-56A、ANSI X9.42、SECG SEC1 で使用される KDF 関数の安全性はハッシュ関数の原像計算困難性に帰着できる。2008 年現在では SHA-1, SHA-256, SHA-384, SHA-512 に対する原像計算困難性については全数探索以外に有効な手段は発見されていない。SHA-1 を使った HMAC が利用されている場合は、SHA-1 の擬似衝突の発見が攻撃に繋がる可能性がある。80 段中 70 段の擬似衝突などが発見されているが、それらを利用した HMAC の攻撃への適用例は報告されておらず、SHA-1 を使った HMAC への攻撃も全数探索以外に有効な手段は発見されていない。

以上より、SHA-1 の危殆化及び移行計画などに注意が必要であるが、SHA-1, SHA-256, SHA-384, SHA-512 に関する限り、NIST SP 800-56A、ANSI X9.42、SECG SEC1 で使用される KDF 関数の安全性が直ちに安全性が脅かされる状況ではないと考えられる。

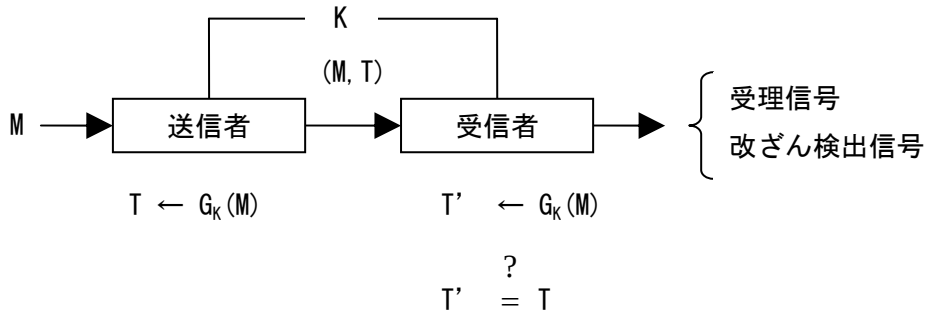
### 9.2.3.標準化動向

個別の KDF 関数の節を参照のこと。

# 10.メッセージ認証コード

## 10.1.メッセージ認証コードの概説

メッセージ認証コードはブロック暗号やハッシュ関数を用いて，メッセージが偽造，改ざんされることを防ぐ技術である．送信者と受信者が秘密鍵 K を共有しており，送信者はタグ生成アルゴリズム G を用いて，平文 M と鍵 K からタグ $T=G_K(M)$ を計算し，メッセージ，タグのペア (M,T) を受信者に送る．ここでタグの長さを固定長の n ビットとする。またメッセージの長さはｎよりも長くてよい.
タグのペア (M,T) を受け取った受信者は確認アルゴリズム V を用いて，受理信号，もしくは改ざん検出信号を出力する．V は，受け取ったメッセージに対し，$T'=G_K(M)$ を計算し，T’=T なら受理信号を，そうでなければ改ざん検出信号を出力する.




図：メッセージ認証コードのモデル

EMAC, XCBC, CMAC は CBC MAC の変形であり,ブロック暗号を用いて構成されているのに対して，HMAC はハッシュ関数を用いて構成される方式である.

## 10.2.個別のメッセージ認証コード

### 10.2.1.CBC MAC

CBC MAC には，パディングの方法や，最終ブロックの処理など，いくつもの仕様がある．次に述べる仕様は，最も単純なものである.

#### (1)技術概要

CBC MAC はブロック暗号 Enc, タグ長 $\tau$, (メッセージ長を規定する) 定数 m をパラメータとする. ブロック長 n のブロック暗号 $Enc_K$: $\{0,1\}^n \rightarrow \{0,1\}^n$ を用いた場合は, $\tau \leqq n$ でなくてはならない.

$CBC\text{-}G_K$: $\{0,1\}^{mn} \rightarrow \{0,1\}^{\tau}$は、鍵 K とメッセージ $M \in \{0,1\}^m$ を入力として、タグ $T = CBC\text{-}G_K(M)$を出力し、$CBC\text{-}V_K$: $\{0,1\}^{mn} \times \{0,1\}^{\tau} \rightarrow$ accept or reject は、鍵 K, メッセージ $M \in \{0,1\}^m$ とタグ $T \in \{0,1\}^{\tau}$ を入力として、accept or reject = $CBC\text{-}V_K(M,T)$ を出力する。

タグ生成アルゴリズム CBC-G, 確認アルゴリズム CBC-V はそれぞれ以下のように動作する.

CBC MAC のタグ生成アルゴリズム $CBC\text{-}G_K(\cdot)$

```
Algorithm CBC-G_K(M)
Y[0] ← 0^n
Partition M into M[1]…M[m]
for i ← 1 to m do
   X[i] ← M[i] ⊕ Y[i-1]
   Y[i] ← Enc_K(X[i])
T ← the left most τ bits of Y[m]
return T
```

CBC MAC の確認アルゴリズム $CBC\text{-}V_K(\cdot,\cdot)$

```
Algorithm CBC-V_K(M,T)
T' ← CBC-G_K(M)
if T=T' then return accept else return reject
```

### (2)安全性

Bellare, Kilian, Rogaway と Bellare, Pietrzak, Rogaway とにより安全性が解析されている。

- M. Bellare, J. Kilian, and P. Rogaway. The security of the cipher block chaining message authentication code, Journal of Computer and System Science (JCSS), Vol. 61, No. 3, pp. 362--399, 2000.
- M. Bellare, K. Pietrzak, and P. Rogaway. Improved Security Analyses for CBC MACs. Advances in Cryptology - CRYPTO 2005, Lecture Notes in Computer Science vol.3621, pp.527-545, Springer-Verlag, 2005.

ブロック暗号 Enc が安全な擬似ランダム置換族であれば, CBC MAC は, 偽造不可能性の意味で安全な MAC であることが示されている. ただし、メッセージ空間がある定数 m に対

し、 $\{0,1\}^{mn}$ となっていない場合、特に可変長のメッセージ空間（たとえば、$(\{0,1\}^{mn})^{+}$）に対しては、CBC MAC は安全な MAC ではなくなる。

(3)効率

CBC MAC の効率は，以下のようにまとめられる.

ブロック暗号の鍵 K は一つのみである。
ブロック暗号の鍵スケジューリングの呼び出し回数：1 回である.
メッセージ M に対するタグを生成するのにかかるブロック暗号の呼び出し回数：(|M|/n) 回の呼び出しである.
事前計算するべきブロック暗号の呼び出し回数：必要ない.
並列処理性：並列処理はできない.

(4)標準化状況

ISO/IEC 9797-1, ISO 16609, NIST FIPS PUB 113 に含まれている.

- ISO/IEC 9797-1:1999, Information technology -- Security techniques -- Message Authentication Codes (MACs) -- Part 1: Mechanisms using a block cipher.
- ISO 16609:2004, Banking -- Requirements for message authentication using symmetric techniques.
- National Institute of Standards and Technology, Federal Information Processing Standards Publication 113, Computer data authentication, 1994.

なお，メッセージ長の問題を解決するために，パディング，最終出力の前に暗号化を施すなど，いくつかの変形がある．正確な仕様については，各標準の文書を参照されたい.

10.2.2.EMAC

(1)技術概要

EMAC はブロック暗号 Enc とタグ長 $\tau$ をパラメータとする．ブロック長 n のブロック暗号 $Enc_K$: $\{0,1\}^n \rightarrow \{0,1\}^n$ を用いた場合は，$\tau \leqq n$ でなくてはならない.
$EMAC\text{-}G_K$: $\{0,1\}^{mn} \rightarrow \{0,1\}^{\tau}$ は、鍵 K1,K2 とメッセージ $M \in (\{0,1\}^m)^+$ を入力として、タグ $T = EMAC\text{-}G_K(M)$ を出力し、$EMAC\text{-}V_K$: $(\{0,1\}^{mn})^{+} \times \{0,1\}^{\tau} \rightarrow$ accept or reject は、鍵 K1,K2,

メッセージ M ∈ $(\{0,1\}^m)^+$ とタグ T ∈ $\{0,1\}^\tau$ を入力として、accept or reject = $EMAC\text{-}V_K(M,T)$ を出力する。
タグ生成アルゴリズム EMAC-G, 確認アルゴリズム EMAC-V はそれぞれ以下のように動作する.

EMAC のタグ生成アルゴリズム $EMAC\text{-}G_{K1,K2}(\cdot)$

```
Algorithm EMAC-G_{K1,K2}(M)
Y[0] ← 0^n
Partition M into M[1]…M[m]
for i ← 1 to m do
   X[i] ← M[i] ⊕ Y[i-1]
   Y[i] ← Enc_{K1}(X[i])
T ← the left most τ bits of Enc_{K2}(Y[m])
return T
```

EMAC の確認アルゴリズム $EMAC\text{-}V_{K1,K2}(\cdot,\cdot)$

```
Algorithm EMAC-V_{K1,K2}(M,T)
T' ← EMAC-G_{K1,K2}(M)
if T=T' then return accept else return reject
```

### (2)安全性

Petrank, Rackoff、Bellare, Pietrzak,Rogaway により安全性が解析されている。

- E. Petrank and C. Rackoff. CBC MAC for real-time data sources. Journal of Cryptology, Vol. 13, No. 3, pp. 315--338, Springer-Verlag, 2000.
- M. Bellare, K. Pietrzak, and P. Rogaway. Improved Security Analyses for CBC MACs. Advances in Cryptology - CRYPTO 2005, Lecture Notes in Computer Science vol.3621, pp.527-545, Springer-Verlag, 2005.
- K. Pietrzak: A Tight Bound for EMAC. ICALP (2) 2006: 168-179 Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II. Lecture Notes in Computer Science 4052 Springer 2006

ブロック暗号 E が安全な擬似ランダム置換族であれば, EMAC は, 偽造不可能性の意味で安全な MAC であることが示されている.

### (3)効率

EMAC の効率は，以下のようにまとめられる.

ブロック暗号の鍵 K1,K2 の二つである.
ブロック暗号鍵スケジューリングの呼び出し回数：2 回である.
メッセージ M に対するタグを生成するのにかかるブロック暗号の呼び出し回数：(|M|/n)+1 回の呼び出しである.
事前計算するべきブロック暗号の呼び出し回数：必要ない.
並列処理性：並列処理はできない.

### (4)標準化動向

ISO/IEC 9797-1 に含まれている。

- ISO/IEC 9797-1:1999, Information technology - Security techniques - Message Authentication Codes (MACs) - Part 1: Mechanisms using a block cipher.

## 10.2.3.XCBC

### (1)技術概要

XCBC はブロック暗号 Enc とタグ長 $\tau$ をパラメータとする. ブロック長 n のブロック暗号 $Enc_K$: $\{0,1\}^n \rightarrow \{0,1\}^n$ を用いた場合は，$\tau \leqq n$ でなくてはならない.
$XCBC\text{-}G_K$: $\{0,1\}^* \rightarrow \{0,1\}^\tau$は、鍵 K1,K2,K3 とメッセージ M $\in \{0,1\}^*$を入力として、タグ T = $XCBC\text{-}G_K$(M)を出力し、$XCBC\text{-}V_K$: $\{0,1\}^* \times \{0,1\}^\tau \rightarrow$ accept or reject は、鍵 K1,K2,K3, メッセージ M $\in \{0,1\}^*$ とタグ T $\in \{0,1\}^\tau$ を入力として、accept or reject = $XCBC\text{-}V_K$(M,T) を出力する。
タグ生成アルゴリズム XCBC-G, 確認アルゴリズム XCBC-V はそれぞれ以下のように動作する.

$XCBC\text{-}G_{K1,K2,K3}$(・) のタグ生成アルゴリズム

```
Algorithm XCBC-G_{K1,K2,K3}(M)
Y[0] ← 0^n
Partition M into M[1]…M[m]
```

```
for i ← 1 to m-1 do
    X[i] ← M[i] ⊕ Y[i-1]
    Y[i] ← Enc_K1(X[i])
X[m] ← pad_n(M[m]) ⊕ Y[m-1]
if |M[m]|=n then X[m] ← X[m] ⊕ K2 else X[m] ← X[m] ⊕ K3
T ← the left most τ bits of Enc_K1(X[m])
return T
```

XCBC は M の長さが n の倍数でなくてもよい．3 行目において，M=M[1]M[2]…M[m-1]M[m] は，|M[1]|=|M[2]|=…=|M[m-1]| かつ 1 ≦ |M[m]| ≦ n となるように分割される．M=$\varepsilon$（空文字列、ヌルストリング）のときは例外である．この場合，|M[m]|=0 となる．

また，7 行目の関数 $pad_n$: $\{0,1\}^{\leqq n} \rightarrow \{0,1\}^n$ は以下のように定義される．
a を長さが高々 n ビットのビット列とする（a=$\varepsilon$ でもよい）．このとき，

$$pad_n(a) = a10^{n-|a|-1} \text{ if } |a|<n$$
$$a \text{ if } |a|=n$$

XCBC の確認アルゴリズム $XCBC\text{-}V_{K1,K2,K3}(\cdot,\cdot)$

```
Algorithm XCBC-V_K1,K2,K3(M,T)
T' ← XCBC-G_K1,K2,K3(M)
if T=T' then return accept else return reject
```

### (2)安全性

Black, Rogaway、Iwata, Kurosawa、Minematsu, Matusshima により安全性が解析されている。

- J. Black and P. Rogaway, CBC MACs for Arbitrary-Length Messages:The Three-Key Constructions, Advances in Cryptology - CRYPTO 2000, Lecture Notes in Computer Science vol.1880, pp.197-215, Springer-Verlag, 2000.
- T. Iwata and K. Kurosawa, Stronger security bounds for OMAC, TMAC and XCBC, Progress in Cryptology - INDOCRYPT 2003, Lecture Notes in Computer Science vol.2904, pp.402-415, 2003.
- K. Minematsu and T. Matsushima, New Bounds for PMAC, TMAC, and XCBC. Fast Software Encryption-, FSE'07, LNCS 4593, pp.434-451, 2007.

ブロック暗号 Enc が安全な擬似ランダム置換族であれば，XCBC は，偽造不可能性の意味

で安全な MAC であることが示されている.

(3)効率

XCBC の効率は, 以下のようにまとめられる.

ブロック暗号の鍵 K1 と n ビットの鍵 K2,K3 の計 3 つが必要である.
ブロック暗号鍵スケジューリングの呼び出し回数：1 回である.
メッセージ M に対するタグを生成するのにかかるブロック暗号の呼び出し回数：$\max\{1, \lceil |M|/n \rceil\}$ 回の呼び出しである.
事前計算するべきブロック暗号の呼び出し回数：必要ない.
並列処理性：並列処理はできない.

(4)標準化状況

NISTの利用モードの公募へ提案されていたが、RFC 3566(AES-XCBC-MAC-96)やRFC 3664(AES-XCBC-PRF-128、擬似乱数生成器)という派生物となった。

- S. Frankel, H. Herbert, The AES-XCBC-MAC-96 Algorithm and Its Use With IPsec, RFC 3566, The Internet Society (2003), available at http://www.ietf.org/rfc/rfc3566.txt
- P. Hoffman, The AES-XCBC-PRF-128 Algorithm for the Internet Key Exchange Protocol (IKE), RFC 3664, The Internet Society (2004), available at http://www.ietf.org/rfc/rfc3664.txt

## 10.2.4.CMAC

(1)技術概要

CMAC はブロック暗号 Enc とタグ長 $\tau$ をパラメータとする. ブロック長 n のブロック暗号 $Enc_K$: $\{0,1\}^n \rightarrow \{0,1\}^n$ を用いた場合は, $\tau \leqq n$ でなくてはならない.
$CMAC\text{-}G_K$: $\{0,1\}^* \rightarrow \{0,1\}^\tau$は、鍵 K とメッセージ M ∈ {0,1}*を入力として、タグ T = $CMAC\text{-}G_K$(M)を出力し、$CMAC\text{-}V_K$: $\{0,1\}^* \times \{0,1\}^\tau \rightarrow$ accept or reject は、鍵 K, メッセージ M ∈ {0,1}* とタグ T ∈ $\{0,1\}^\tau$ を入力として、accept or reject = $CMAC\text{-}V_K$(M,T) を出力する。
生成アルゴリズム CMAC-G, 確認アルゴリズム CMAC-V はそれぞれ以下のように動作する.

CMAC の生成アルゴリズム CMAC-$G_K$(・)

Algorithm CMAC-$G_K$(M)
$L \leftarrow Enc_K(0^n)$
$Y[0] \leftarrow 0^n$
Partition M into M[1]…M[m]
for i ← 1 to m-1 do
    $X[i] \leftarrow M[i] \oplus Y[i-1]$
    $Y[i] \leftarrow Enc_K(X[i])$
$X[m] \leftarrow pad_n(M[m]) \oplus Y[m-1]$
if |M[m]|=n then $X[m] \leftarrow X[m] \oplus (L \cdot u)$
else $X[m] \leftarrow X[m] \oplus (L \cdot u^2)$
T ← the left most τ bits of $Enc_K(X[m])$
return T

CMAC も XCBC と同様，M の長さが n の倍数でなくてもよい．3 行目において，M=M[1]M[2]…M[m-1]M[m]は，|M[1]|=|M[2]|=…=|M[m-1]| かつ 1 ≦ |M[m]| ≦ n となるように分割される．M=ε（空文字列、ヌルストリング）のときは例外で，この場合，|M[m]|=0 となる．

$a \in \{0,1\}^n$ に対し，a ・ u は、GF($2^n$)上の乗算によって定義される。a を GF(2)係数の多項式 $a(u)=a_{n-1}u^{n-1}+\cdots+a_1u+a_0$ とみなして、u を掛けたものとして定義され、a ・ u は一般に、

$a \cdot u = a \ll 1$ if msb(a)=0
$a \ll 1 \oplus Cst_n$ otherwise

となる．a ・ $u^2$ は (a ・ u) ・ u として得られる．
a ≪ 1 は a の左 1 ビットシフトを表し，$a=a_{n-1}a_{n-2}\cdots a_1a_0$ と a をビット表現した場合，a ≪ $1=a_{n-2}a_{n-3}\cdots a_00$ となる．すなわち，最上位ビットはなくなり，最下位ビットに 0 が補充される．
また， msb(a) は a の最上位ビットを表し， $Cst_n$ は GF($2^n$)を定義する既約多項式に付随する n ビットの定数である．たとえば，$Cst_{128}=0^{120}10000111$ であり，$Cst_{64}=0^{59}11011$ である．

また， 7 行目の関数 $pad_n$: $\{0,1\}^{\leqq n} \rightarrow \{0,1\}^n$ は XCBC の場合と同じように定義される．

CMAC の確認アルゴリズム CMAC-$V_K$(・, ・)

```
Algorithm CMAC-VK(M,T)
T' ← CMAC-G_K(M)
if T=T' then return accept else return reject
```

### (2)安全性

Iwata, Kurosawa により安全性が解析されている。

- T. Iwata and K. Kurosawa. OMAC: One-Key CBC MAC. Fast Software Encryption, FSE 2003, Lecture Notes in Computer Science vol.2887, pp.129-153, Springer-Verlag, 2003.
- T. Iwata and K. Kurosawa. Stronger security bounds for OMAC, TMAC and XCBC. Progress in Cryptology - INDOCRYPT 2003, Lecture Notes in Computer Science vol.2904, pp.402-415, 2003.

ブロック暗号 Enc が安全な擬似ランダム置換族であれば，CMAC は，偽造不可能性の意味で安全な MAC であることが示されている.

### (3)効率

CMAC の効率は，以下のようにまとめられる.

- ブロック暗号の鍵 K の一つのみである.
- ブロック暗号鍵スケジューリングの呼び出し回数：1 回である.
- メッセージ M に対するタグを生成するのにかかるブロック暗号の呼び出し回数：$\max\{1, \lceil |M|/n \rceil\}$ 回の呼び出しである.
- 事前計算するべきブロック暗号の呼び出し回数：$L=Enc_K(0^n)$ を計算するのに 1 回必要である.
- 並列処理性：並列処理はできない.

### (4)標準化状況

NIST は 2005 年 5 月に利用モードの公募に応募されていた OMAC1 を推奨方式として採用した. SP 800-38B では，OMAC1 は CMAC (Cipher-based MAC) と呼ばれている.

- National Institute of Standards and Technology, Special Publication 800-38B, Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication, 2005.

## 10.2.5.HMAC

### (1)方式

HMAC (Hash-based MAC) は反復型ハッシュ関数 F とタグ長 $\tau$ をパラメータとする. 出力長 n の反復型ハッシュ関数 F:{0,1}*→{0,1}$^n$ を用いた場合は, $\tau \leqq n$ でなければならない.
HMAC-G のタグ生成アルゴリズム HMAC-G:{0,1}$^n$x{0,1}*→{0,1}$^\tau$ は, 鍵 K∈{0,1}$^n$ とメッセージ M∈{0,1}*を入力とし, タグ T = HMAC-GK(M)∈{0,1}$^\tau$ を出力する.
HMAC-V:{0,1}$^n$x{0,1}*x{0,1}$^\tau$→accept or reject は, 鍵 K∈{0,1}$^n$, メッセージ M∈{0,1}*, タグ T∈{0,1}$^\tau$ を入力とし, accept or reject=HMAC-VK(M,T)を出力する.
タグ生成アルゴリズム HMAC-G, 確認アルゴリズム HMAC-V はそれぞれ以下のように動作する.

HMAC のタグ生成アルゴリズム HMAC-$G_K$(・)

```
Algorithm HMAC-G_K(M)
T ← the left most τ bits of F(K̄ ⊕ opad, F(K̄ ⊕ ipad, M))
Return T
```

ハッシュ関数 F:{0,1}*→{0,1}n の圧縮関数を f:{0,1}$^n$x{0,1}$^b$→{0,1}$^n$ とする. ここで, b は圧縮関数に入力されるメッセージのブロック長である. $\bar{K}$ は, 長さが b ビットになるまで K に 0 を付加して得られる系列である. opad は, 00110110 を長さが b ビットになるまで繰り返して得られる系列である. ipad は, 01011100 を長さが b ビットになるまで繰り返して得られる系列である.

HMAC の確認アルゴリズム HMAC-VK(・, ・)

```
Algorithm HMAC-VK(M,T)
T' ← HMAC-GK(M)
if T=T' then return accept else return reject
```

### (2)安全性

Bellare, Canetti, Krawczyk と Bellare により安全性が解析されている。

- M. Bellare, R. Canetti, and H. Krawczyk, Keying Hash Functions for Message Authentication, Advances in Cryptology - CRYPTO '96, Lecture Notes in Computer Science vol.1109, pp.1--15, Springer-Verlag, 1996.
- Bellare, New Proofs for NMAC and HMAC: Security Without Collision-Resistance, Advances in Cryptology - CRYPTO 2006,

Lecture Notes in Computer Science vol.4117, pp.602-619, Springer-Verlag, 2006.

ベースとなる圧縮関数が安全な擬似ランダム関数であれば，HMAC は，偽造不可能性の意味で安全な MAC であることが示されている.

### (3)効率

HMAC の効率は以下のようにまとめられる.

- ハッシュ関数の入力の一部として使用される鍵 K の一つである．この鍵の長さはハッシュ関数 F の出力長と等しい.
- メッセージ M に対するタグを生成するのにかかるハッシュ関数の圧縮関数の呼び出し回数：一カブロック $\bar{K} \oplus$ opad と $\bar{K} \oplus$ ipad の処理のため，NMAC での呼び出し回数より 2 回多い．ただし，K を変更しない限り，K1 = f(IV, $\bar{K} \oplus$ opad)，K2 = f(IV, $\bar{K} \oplus$ ipad) を計算して保持しておけば，NMAC と同じ回数となる.
- 並列処理性：並列処理はできない.

### (4)標準化状況

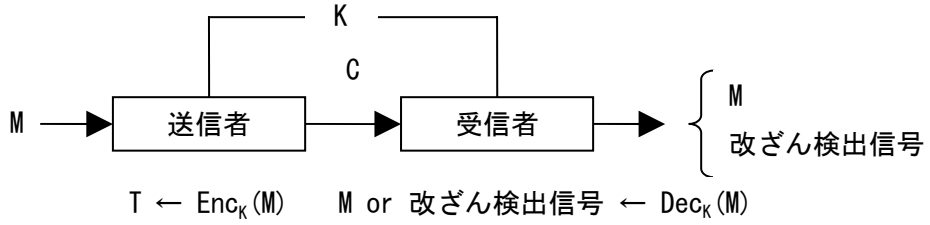

- ISO/IEC 9797-2:2002 Information technology - Security techniques - Message Authentication Codes (MACs) - Part 2: Mechanisms using a dedicated hash-function.
- National Institute of Standards and Technology, Federal Information Processing Standards Publication 198, The Keyed-Hash Message Authentication Code (HMAC), 2002.
- S. Kent, R. Atkinson, IP Authentication Header, RFC 2402, The Internet Society (1998), available at http://www.ietf.org/rfc/rfc2402.txt


IPSec では，任意の認証アルゴリズムの実装が許されているが，最低限 MD5 と SHA-1 による HMAC の実装がなされなければならない.

# 11. メッセージ認証付き暗号利用モード

## 11.1. メッセージ認証付き暗号利用モードの概説

メッセージ認証付き暗号利用モードはブロック暗号を用いて，暗号化及びメッセージの偽造・改ざんを防ぐ機能を併せ持つ技術である．送信者と受信者が秘密鍵 K を共有しており，送信者は暗号化アルゴリズム Enc を用いて，平文 M と鍵 K から暗号文 $C = Enc_K(M)$ を計算し，C を受信者に送る．受信者は復号アルゴリズム Dec を用いて，暗号文 C と鍵 K から平文 $M = Dec_K(C)$ を，もしくは改ざん検出信号を出力する．




図：メッセージ認証付き暗号利用モードのモデル

## 11.2. 個別のメッセージ認証付き暗号利用モード

### 11.2.1. CCM

#### (1) 技術概要

CCM (Counter with CBC-MAC)は,CTR モードによる暗号化と CBC-MAC を組み合わせたメッセージ認証付き暗号利用モードである.具体的には,メッセージ及び認証データに対して,CBC-MAC による MAC を生成し,MAC 処理で生成されたタグとメッセージを CTR モードで暗号化する,CTR,CBC-MAC の両方に同じ秘密鍵を用いているので鍵のセットアップは１回である.
暗号化処理や MAC 生成には,いくつかの詳細なパディングが記載されており,その一部には長さ情報が含まれる。

#### (2) 安全性

Jonsson により安全性評価が与えれている.

- J. Jonsson, On the Security of CTR + CBC-MAC, Selected Areas in

Cryptography, 9th Annual Workshop, SAC 2002, St. John's, Newfoundland, Canada, Aug. 2002, Revised Papers, ed. K. Nyberg and H. Heys, p.76-93, Lecture Notes in Computer Science vol.2595, Springer-Verlag, 2002.

詳しくは、CRYPTREC Report 2005 の付録 5 を参照のこと。

### (3) 効率

CCM は ECB や CBC モードの 2 倍のブロック暗号呼び出しを行なうため, 処理量も ECB のそれに比べて約 2 倍である.

ただし, 実質的に CTR モードと CBC-MAC の組合せであり, データサイズが(処理系が扱えるメモリサイズに対して)大きい場合には注意が必要である. 例えば, ストリーミングデータなどへの処理には, CCM として, 内部で呼び出す CTR の処理と CBC-MAC の処理, 両方を交互に行なうような実装を行なわないと, 処理が不可能となる. この場合, 中間データの保持のためにいくらかの必要レジスタサイズの増加が考えられる.

### (4) 並列処理性

まず, CCM 処理中の CTR 処理と, CBC-MAC 処理は並列処理が可能である. 従って適切な実装により 2 並列度までは簡単に達成できる. しかし, CBC-MAC には並列処理機能がないため, それ以上の並列処理は CTR のみに適用可能となる. これは復号処理についても同じことがいえる.

### (5) 復号

CCM モードでは, ブロック暗号の復号関数を利用しない. よって, CCM 暗号化, CCM 復号の両方の機能を実装する場合には, その実装コストは, CBC や ECB に比較して軽いことが期待できる.

### (6) 標準化動向

- D. Whiting, R. Housley, and N. Ferguson, AES Encryption & Authentication Using CTR Mode & CBC-MAC, IEEE P802.11 doc 02/001r2, May 2002.
- NIST SP 800-38C 800-38C、Recommendation for Block Cipher Modes of Operation : The CCM Mode for Authentication and Confidentiality, May 2004.

### 11.2.2.GCM

#### (1)技術概要

GCM (Galois/Counter Mode)モードは、CTR モードの暗号化と universal hash 関数による MAC 生成を組み合わせたメッセージ認証つき暗号化方式である.
具体的には,メッセージに対して CTR モードで暗号文を生成し, その暗号文と, 暗号化はされない認証対象のメッセージに関して, GF($2^{128}$)上で定義された universal hash 関数により認証タグを計算し, 暗号文に付加する.

#### (2)安全性

McGrew, Viega により安全性評価が与えれている.

- D. McGrew, J. Viega, The Security and Performance of the Galois/Counter Mode (GCM) of Operation, Progress in Cryptology - INDOCRYPT 2004, 5th International Conference on Cryptology in India, Chennai, India, December 20-22, 2004, Proceedings. p.343-355, Lecture Notes in Computer Science vol.3348, Springer-Verlag, 2004.

詳しくは、CRYPTREC Report 2005 の付録 5 を参照のこと。

#### (3)効率

GCM のソフトウェア実装では, GF($2^{128}$)上の乗算を 256B, 4KB, 64KB のテーブル参照を用いて実装しており, テーブルサイズが大きいほど高速である.
また、ハードウェア実装においても GF($2^{128}$)上の乗算は小型かつ高速に実装することができ, GCM はソフトウェア実装, ハードウェア実装ともに優位性をもっている.

#### (4)並列処理性

universal hash 関数の積和演算は基本的に逐次処理であるが, GF($2^{128}$)上の乗算は AES の暗号化処理に対して非常に軽いため, CTR モードの高速性を犠牲にすることはほとんどない.

#### (5)復号

GCM モードでは，ブロック暗号の復号関数を利用しない．よって，GCM 暗号化，GCM

復号の両方の機能を実装する場合には(もちろん，GF($2^{128}$)上乗算のコストが新たに必要だが AES の暗号化処理に対して非常に軽いため)，CBC や ECB に比較して軽いことが期待できる．

### (6) 標準化動向

(i) NIST SP 800-38C 800-38D、Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC, November 2007.

発行日　2008年5月30日　第1版 第1刷
　　2008年7月28日　第2版
発行者
・〒184-8795
東京都小金井市貫井北四丁目2番1号
独立行政法人 情報通信研究機構
（情報通信セキュリティ研究センター セキュリティ基盤グループ）
NATIONAL INSTITUTE OF
INFORMATION AND COMMUNICATIONS TECHNOLOGY
4-2-1 NUKUI-KITAMACHI,KOGANEI
TOKYO,184-8795 JAPAN

・〒113-6591
東京都文京区本駒込二丁目28番8号
独立行政法人 情報処理推進機構
（セキュリティセンター 暗号グループ）
INFORMATION-TECHNOLOGY PROMOTION AGENCY JAPAN
2-28-8 HONKOMAGOME,BUNKYO-KU