

# 情報セキュリティ

## 《学修項目》

- ◎セキュリティの3要素（機密性、可用性、完全性）
- ◎データの暗号化、復号化
- データの盗聴、改ざん、なりすまし
- 電子署名、公開鍵認証基盤（PKI）
- ユーザ認証とアクセス管理
- マルウェアによるリスク（データの消失・漏洩、サービスの停止など）

## 《キーワード》

情報セキュリティの3要素/7要素、機密性、可用性、完全性、機密性の高い情報、マルウェア、情報の格付け、秘密鍵暗号方式、公開鍵暗号方式、電子署名手順、PKI（公開鍵認証基盤）、セキュリティホールと脆弱性、ゼロデイ攻撃、標的型攻撃、2要素認証方式

## 《参考文献、参考書籍》

- [1] 東京大学MIセンター公開教材 「2-6 情報セキュリティ」 《利用条件CC BY-NC-SA》
- [2] マスタリングTCP/IP 情報セキュリティ編(第2版) (オーム社)
- [3] データサイエンスの考え方 社会に役立つAI×データ活用のために (オーム社)
- [4] 数理・データサイエンス・AI公開講座 (放送大学)

## 1. 情報セキュリティとは

あらまし

- 情報セキュリティはなぜ必要か
- 情報セキュリティの3要素/7要素
- 機密性の高い情報
- 情報セキュリティが損なわれた例
- マルウェア（malware）によるリスク
- 情報の格付け

### 1.1 情報セキュリティとは、リスク管理とは

#### 1.1.1 情報セキュリティはなぜ必要か

脅威（加害者の意図）による故意

- 金銭目的、政治目的、好奇心

過失によるもの

- 操作ミス、設定ミス、更新忘れ、故障

対策（危機管理）が必要

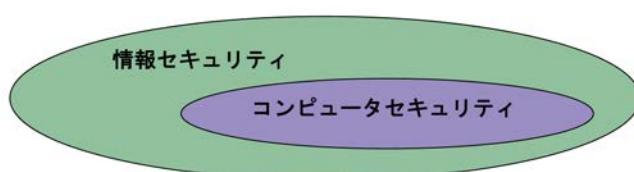
- 情報セキュリティ（ITセキュリティ）が必要

## セキュリティ(security)とは

国語辞書「スーパー大辞林」では以下のように示されています：

- 安全、防犯、安全保障。
- (有価)証券。
- コンピューターセキュリティ  
「コンピューターを利用する上での安全性。コンピューターへの不正アクセスやデータの改竄（かいざん）などの問題を扱う分野」

- 本節のITセキュリティでは、狭義の③コンピュータセキュリティだけではなく、より広義の情報の①安全をあつかいます。  
例えば、個人情報が漏洩するような事象では、データの漏洩経路としてコンピュータネットワーク、印刷物などが考えられます。どちらの経路でも情報セキュリティの問題であることは変わりません。一方、前者はコンピュータセキュリティの問題ですが、後者はそうではありません。
- この資料は、情報セキュリティマネジメント体系（Information Security Management Systems: ISMS）をとりあつかった日本産業規格 JIS Q27000:2019 系列文書に沿った内容としています。



東京大学 数理・情報教育研究センター 小林克志 2021 CC BY-NC-SA

5

## 情報とは

文脈に依存するが、日本産業規格（JIS X 0001-1994）では情報とデータを以下のように定義している。

**情報**：“事実、事象、事物、過程、着想などの対象物に関して知り得たことであって、概念を含み、一定の文脈中で特定の意味をもつもの。”

**データ**：“情報の表現であって、伝達、解釈又は処理に適するように形式化され、再度情報として解釈できるもの。”

情報セキュリティにおける「情報」には上の「データ」の意味も含まれる。さらに ISMS では「文書化した情報」を以下のように定め、適切な取り扱いを求めている。

組織が管理し、維持するよう要求されている情報、及びそれが含まれている媒体。文書化した情報は、あらゆる形式及び媒体の形をとることができ、あらゆる情報源から得ることができる。

注) 管理、維持が要求されている情報であれば該当する。すなわち、意図的に記録、保存していくなくても該当する。



あらゆる形式・媒体

あらゆる情報源

東京大学 数理・情報教育研究センター 小林克志 2021 CC BY-NC-SA

6

### 1.1.2 セキュリティリスクとその対応

## 情報セキュリティリスク

### 情報セキュリティリスク

- 脅威が弱点に付け込み、その結果、組織に損害を与える可能性に伴って生じる。

### 脅威

- 損害を与える可能性がある、望ましくない事態の潜在的な原因。

### リスク

- 目的に対する不確かさによる期待からの乖離（ズレ）のこと。乖離には好ましいものも含まれる。
- 不確かな「事象」の起こりやすさと、その結果の組み合わせとして表現されることも多い。

## リスク対応

リスクを修正する活動（プロセス）のこと、以下の事項が含まれます。

以下の対応例では、管理者権限でおこなうコンピュータ操作におけるリスクに対するものを示しました。  
注) コンピュータに対する操作にはシステムを破壊するものも含まれます。WindowsなどのOSでは危険な操作は管理者権限でのみ実行できるようにし、一般利用者には必要なときのみ管理者権限を与えるようにしています。

リスク対応の事項	対応の例
リスクを生じさせる活動をおこなわないことで、回避する。	データが破壊される可能性があるため、管理者権限での操作を禁止する。
機会を追求するために、リスクをとるまたは増大させる。	業務の効率を上げるために、全ての一般利用者に常時管理者権限を与える。
リスク源を除去する。	危険な操作をおこなうプログラムをすべて消去する。
結果を変える。	いつ破壊されても復元できるようにバックアップの回数を多くする。
他者とリスクを共有する。	管理者権限が必要な操作は他社に外注する。
リスクを評価したうえで、リスクを保有する。	事故が発生しても業務への影響は軽微なので、そのまま様子を見る。

リスク対応があらたなリスクを生じさせたり、既存の別のリスクを増大させることもあります。

## 1.2 情報セキュリティの3要素/7要素

### 情報セキュリティの3要素

- 機密性：権限が無い他者に見られないこと
- 完全性：情報が破壊されていないこと
- 可用性：必要なときに情報を利用できること

### 情報セキュリティの7要素

3要素（機密性、完全性、可用性）に加えられた4つの新たな概念

4. 真正性：アクセスする人や団体が許可を得ている（機密性の一部が独立している）
5. 責任追跡性：アクセスする人や団体の操作を追跡できる
6. 否認防止：あとから覆されない
7. 信頼性：システムが正しく動作する

### 1.2.1 機密性

#### 機密性(Confidentiality)

許可されていない個人、組織、プログラムなどに情報を使用させず、開示しない特性。

- パスワード流出・個人情報漏洩など多く報道されている。企業の製品設計図や、未発表論文のデータ漏洩なども機密性に関わるもの。
- 機密性に関する問題は多岐にわたっている。漏洩経路だけをみても、ネットワークの盗聴・記憶装置の流出といった情報機器を介するだけでなく、内部協力者・廃棄文書といった古典的なものも少なくない。

機密性にかかわる事象と影響の例：

- 試験問題の漏洩によって、試験の公正性・公平性を損なう。
- 新商品の情報が競合他社に漏れ、競争上の不利益になる。
- 個人情報の漏洩によって、当局から多額の制裁金を課される。

#### 情報セキュリティの3要素【機密性】

機密性：権限が無い他者に見られないこと

- 権限とは ⇒ 認証のところで説明する
- 機密性が必要な情報の例
  - 個人情報
  - 住所、メールアドレス
  - パスワード
  - クレジットカード番号
  - 成績情報
  - 検診結果（健康）
- 機密性が高い情報の例（個人情報 JIS Q15001に規定されている）

参考:JQA 一般財団法人日本品質保証機構 | JIS Q 15001（個人情報保護）

機密性の高い情報

- 適切にアクセス権を設定し確認する
- やりとりには暗号化を行う
- 暗号鍵の扱いは厳密にする
- 通信相手やメディアの正当性を確認する（電子署名（後述）などを利用する）

## 1.2.2 完全性

### 完全性(Integrity)

情報の正確さ及び完全さの特性。

- ・ 情報が最新・正確に維持されること。
- ・ データ破壊や悪意を持つもの（内部、第三者）による改ざんも含まれる。
- ・ データ破壊の原因には、機器故障・操作ミス・有害なソフトウェア（マルウェア）によるものに加え、媒体の経年劣化や、通信路のノイズによるものも含まれる。
- ・ データ改ざんは不正行為の隠蔽を目的としておこなわれたこともある。

完全性にかかる事象と影響の例：

- ・ 試験採点結果が誤って記録され、成績評価に影響した。
- ・ 取引先の連絡先情報が古いままで更新されておらず、必要な連絡がとれない。
- ・ 組織ホームページが書き換えられ、信用失墜につながる。

## 情報セキュリティの3要素【完全性】

完全性：情報が破壊されていないこと（データが壊れないようにする）

- ・ 機密性だけを重要視するなら、データは全て破壊しておけば良い
- ・ 実際にはそのような状況にはならない
- ・ ランサムウェア（後述）対策の重要性

## 1.2.3 可用性

### 可用性(Availability)

認可された利用者やプログラムなどが要求したときにアクセスおよび使用が可能である特性。

- ・ 情報を利用しなければならないときに利用できること。  
利用するのはヒトだけではなくコンピュータプログラムも含まれます。
- ・ 廃棄によるデータ（文書）の消失、情報サービスの停止もこの特性に関わる。ここで停止の原因は計画的・事故を問わない。

可用性にかかる事象と影響の例：

- ・ 試験当日に試験問題冊子が紛失し、試験が実施できない。
- ・ 秘伝のレシピを知る唯一の料理人が急病で、開店できない。
- ・ 銀行の ATM が取引中に突然停止した。キャッシュカードも返却されず、必要な買い物ができなかった。

## バックアップ・冗長化

可用性・完全性を高める技術として利用されています。

バックアップ：記憶装置のデータを別の記憶装置に複製（コピー）しておき、障害時には複製からデータを復旧させます。

スマートフォンのデータをインターネット経由でバックアップし、故障時の機器交換の復旧で活用することは広くおこなわれています。また、操作を誤ってデータを破壊した際の復旧にも活用できます。

冗長化：2台以上の機器を用意し一部に障害が発生しても正常な機器で代替します。可用性の維持には効果的ですが、コストも増大します。

コンピュータのハードディスクドライブ（HDD）では古くから利用されてきました。

多地域展開：冗長化の一環で、情報システムを地理的に離れた地域に配置します。停電による電力喪失や災害による建物倒壊といったリスクにも対応できます。従来はコストが高く金融システムなどに限られていましたが、近年ではコストも下がりWebサービス、SNSなどで広く使われています。

バックアップの媒体のみを遠隔地に保管する方法も採られています。

### 情報セキュリティの3要素【可用性】

可用性：必要なときに情報を利用できること

- システムが正しく動いている
- バックアップがあって取り出せる状態にある
- 遅滞なく利用できる

#### 1.2.4 真正性

## 真正性(Authenticity)

利用組織及び人、設備、ソフトウェア及び物理媒体などがそれが主張するところのものであること。

- 情報サービス、媒体の利用者、提供者がなりすましではないこと。

真正性にかかる事象と影響の例：

- 試験をなりすまして受験され、試験の公正性・公平性を損なう。
- 奈良すましの取引先から発行された請求先に送金し、損失を被った。
- 発行元の不明なソフトウェアをコンピュータにインストールしたところ、それはマルウェア<sup>\*</sup>であり、内部情報を搾取されたうえ重要なデータを削除された。

(\*)不正かつ有害な動作をするコンピュータソフトウェア

マルウェア（malware）によるリスク

## malware (malicious software)

- コンピュータウィルス
- ワーム
- スパイウェア, キーロガー
- アドウェア
- ランサムウェア

情報セキュリティの3要素を全て破壊していく脅威である

例：ランサムウェア

ランサム=人質 という意味

- 利用者のPCなどのデータを勝手に暗号化し、復号鍵と引き換えに身代金を（ビットコインなどで）要求する ⇒ 完全性・可用性への攻撃
- 最近、身代金を払わないと、データをインターネット上へ公開してしまう攻撃も増えてきている ⇒ 機密性への攻撃
- 金銭目的の脅威は増大している

### 1.2.5 責任追跡性

#### 責任追跡性(Accountability)

- 誰が情報に対する操作をおこなったかを後から追跡できること。  
作成・読み・更新・削除といった操作に加え、権限付与も含まれる。
- 情報機器のアクセスだけではなく、建造物・部屋などへの入退室記録も責任追及性に関わる。

責任追跡性にかかる事象と影響の例：

- 学生からの申告で試験成績が改ざんされたことは明らかとなったが、誰が・いつ改ざんをおこなったかが不明。
- システム管理者（複数）には業務上の都合からあらゆる操作を許す特権的アクセスが与えられていた。特権操作による事故が発生したがどの管理者が操作したかあきらかにできない。

### 1.2.6 否認防止

## 否認防止(non-repudiation)

事象又は処置の発生、及びそれらを引き起こしたエンティティ（実体）を証明する能力。

- いわゆるしらばっくれを検出できる特性のこと。合意した内容を後から否定することができない。

否認防止にかかる事象と影響の例：

- 価格・納期を決めて売買契約を締結したが、後で契約を否定され損害を被る。

### 1.2.7 信頼性

## 信頼性(Reliability)

意図する行動と結果とが一貫しているという特性。

- バグによるシステム誤動作がこの特性に関係している。
- 情報システムだけではなく人や組織が想定した結果を出せないことも含まれる。

信頼性にかかる事象と影響の例：

- ソフトウェアをバージョンアップしたが新旧バージョンで動作が異なる、旧バージョンに依存していた業務が滞る。
- 通販サイトで発注したものとは違ったものが届く、納品遅れ・返品などでコストがかかる。

### 1.3 機密性の高い情報、情報の格付け

#### 機密性の高い情報

- 適切にアクセス権を設定し確認する
- やりとりには暗号化を行う
- 暗号鍵の扱いは厳密にする
- 通信相手やメディアの正当性を確認する（電子署名（後述）などを利用する）

## 情報の分類

すべての情報に対しセキュリティ特性（3 or 7要素）を最高レベルで維持することは現実的ではない。一方で個別の情報の内容ごとに異なる対応をとることも現実的ではない。

- ISMS では情報を格付け（分類）し、必要なレベルで管理する方法が示されている。例えば、機密性については以下のような分類が考えられる。

- 開示されても損害が生じない。
- 開示された場合に、軽微な不具合が生じる。
- 開示された場合に、重要な短期的影響が及ぶ。
- 開示された場合に、組織の存続が危機にさらされる。

情報ごとに求められる特性は異なるため、分類にあたってはそれぞれの特性も考慮する必要がある。

- 一部のみ重要：商品のマニュアルは開示されても損害が生じないが内容の正確性は求められる。すなわち、機密性【低】・完全性【高】といった分類がされる。
- 時間によって格付けが変わる：新製品の情報は発表前は機密性【高】と分類されるが、発表後は公知となり機密性【低】となる。。

### 情報の格付け

#### 機密性

- 機密性3情報：秘密文書
- 機密性2情報：やや秘密文書
- 機密性1情報：公開可能

#### 完全性

- 完全性2情報：破損すると業務継続性が困難になる
- 完全性1情報：上記以外の情報

#### 可用性

- 可用性2情報：紛失すると業務に重大な障害をもたらす
- 可用性1情報：上記以外の情報

## 2. 暗号方式（公開鍵暗号、電子署名、PKI）

### 2.1 暗号に関連するターム

## 暗号で使われる用語

暗号化：秘匿化のためにデータを権限を与えていない者にとって意味のわからないデータに変換する操作

平文：変換前のデータ

暗号文：変換後のデータ

復号化：暗号から平文に変換する操作

アルゴリズム：暗号化・複合化の手続き

鍵：アルゴリズムの結果を決定する情報

解読：第三者が暗号から平文に変換する操作

メッセージ：交換される情報\*

古典暗号：アルゴリズム・鍵の両方を秘匿する。

現代暗号：アルゴリズムは公開し、鍵だけを秘匿する。

アルゴリズムはプログラムなどで利用者に配布する必要があり、秘匿することは事実上不可能です。

一方、公開によって第三者の安全性評価が可能となり信頼性が向上するという利点があります。

(\*) 暗号は情報交換を前提に設計されることからこの用語がよく使われる。



平文：誰でも内容（意味）を解釈できるデータ 暗号文：権限がない人・団体・装置に内容を読み取られないように加工されたデータ 暗号化：平文を暗号にする 復号化：暗号文を平文にする

暗号鍵：暗号化するときの鍵（情報） 復号鍵：復号化するときの鍵（情報） ※対にしないでバラバラになっていることもある

送信者：平文を暗号化して送信する者（ノード） 受信者：暗号文を受信する者（ノード）

攻撃者：送受信者間の通信を傍受して、暗号文を平文に戻すことを試みる者（ノード）

## 2.2 共通鍵暗号方式（古典暗号）

### 共通鍵暗号方式

暗号化と復号化で同じ鍵を使用します。前のシーザー暗号や最初の現代暗号の DES\*を含めて、多くの方式が考案されています。

秘匿性の維持には鍵は第三者に知られないことが必要です。個人のスマホデータの暗号化では鍵は当人で管理すればよくあまり問題は生じません。しかし、暗号文を電子メールなどで交換するには送受信者で鍵を安全に、すなわち第三者に知られないように共有する必要があります。

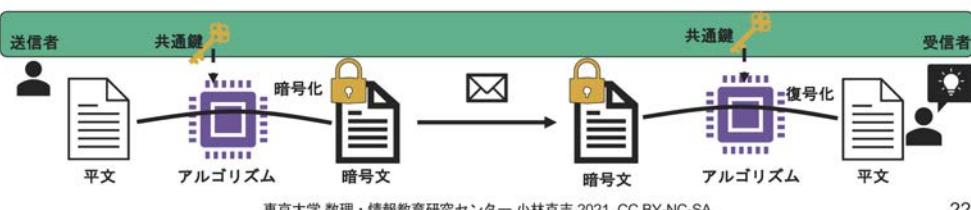
課題：この方式では送受信者が増えると鍵の共有が問題になります。共有方針には大きく分けて以下が考えられます。

1. すべての相手で同じ鍵を共有
2. 相手ごとに異なる鍵を共有

1.は送受信者が増えるに従って鍵が漏洩する可能性が高くなります。

2.では送受信者を N とすると必要な鍵の数は  $N C_2 = \frac{N(N-1)}{2}$  となり、N が大きくなると鍵の数が膨大（爆発）となり実用には耐えられません。次に紹介する公開鍵方式を用いることで実用的な鍵の共有ができるようになります。

(\*) Data Encryption Standard の略、1976 年に合衆国で採用された暗号標準



## 共通鍵暗号方式（古典暗号）

- 送信者と受信者が共通の鍵を取り交わし、それを秘密鍵とする
- 送信者が平文を秘密鍵で暗号化して送信し、受信者は受信した暗号文を秘密鍵で復号化する
- 密密鍵のセキュアな共有の手段が困難となる

## Python PyCryptodomeライブラリを用いたAES暗号化・復号化処理の例

```
In [1]: # PyCryptodomeライブラリをインストールする
!pip install pycryptodome
```

```
In [2]: # AESと乱数生成用のパッケージをインポート
from Crypto.Cipher import AES
from Crypto.Random import get_random_bytes

key = get_random_bytes(16) # 鍵生成(128bit)
print(key) # 共通鍵を表示してみる

cipher = AES.new(key, AES.MODE_EAX) # 暗号器を生成する
cipher_dec = AES.new(key, AES.MODE_EAX, cipher.nonce) # 復号器を生成する(nonceを

b'\xab\x04\xdf\r\xad\xb5\x1c\x51\x066L'
```

```
In [3]: data = b"Hello, AES World!" # 平文 暗号化する文字列
ciphertext, tag = cipher.encrypt_and_digest(data) # 暗号化処理

# 暗号文 ciphertext を確認する
print(ciphertext)
##print(tag)
##print(cipher.nonce)

dec_data = cipher_dec.decrypt_and_verify(ciphertext, tag) # 復号処理
print(dec_data) # 復号処理後の平文を確認

b'\x8e\x9b\x9f\xe9[\x8b\x92U\x12H\xd2?J8\x13c'
b'Hello, AES World!'
```

## 2.3 公開鍵暗号方式（現代暗号）

## 公開鍵暗号方式

暗号化にだれにでも公開してよい公開鍵を、復号化で所有者以外に知られない秘密鍵の鍵ペアを使用する暗号方式で、RSA暗号<sup>\*</sup>がよく知られています。秘匿性の維持だけではなく、完全性・真正性の維持にも利用されます。

公開鍵から秘密鍵の割り出しや暗号文の解読が事実上不可能なアルゴリズム・鍵ペアが利用されます。

公開鍵暗号方式の処理は遅いので動画のような大きなメッセージの暗号化では効率が悪くなります。このため、メッセージ本体の暗号化は共通鍵暗号方式でおこない、比較的サイズの小さい共通鍵のみを公開鍵で暗号化して送受信者で共有する方式が利用されています。

必要な鍵の数：公開鍵暗号方式では送受信者がそれぞれの鍵ペアを作成し、公開鍵を全員に公開するだけで秘匿性が維持できます。すなわち、N人のメッセージ交換に必要な公開鍵の総数はN個となり、共通鍵方式で課題となつた鍵の数の爆発は抑えられます。

課題：この方式では公開鍵の真正性が問題になります。すなわち、使用する公開鍵が確実に所有者から提供されている必要があります。

\* 1977年にRivest-Shamir-Adlemanによって発表された公開鍵暗号方式、広く利用されています。



### 公開鍵暗号方式（現代暗号）

- 受信者が、暗号鍵を予め公開しておく
- 鍵の管理が容易となるが、暗号鍵から復号鍵が容易に推定できないようにする
- 受信者の復号鍵は秘密にしておく

#### 2.3.1 RSA公開鍵暗号方式

- Rivest, Shamir, Adlemanが発明したので頭文字を取ってRSAと呼ばれている。
- 暗号鍵から復号鍵が容易に推定できないようになるため、(巨大な)素数の素因数分解の推定困難性を利用している（計算量的安全性）
- 理論基盤としては、フェルマーの小定理、中国風剩余定理が用いられている

#### 受信者の鍵生成の手順

- 受信者が2つの素数  $p, q$  を選ぶ
- $n = p \times q$  とする
- $(p-1)(q-1)$  と互いに素な数のうち、1つを  $e$  とおく
- $e \times d = (p-1)(q-1) \times k + 1$  を満たすような整数  $d, k$  のペアを求めておく

#### 2.3.2 RSA公開鍵暗号方式による暗号・復号手順

##### 受信者の公開鍵の情報

- ここまで挙っている情報は 素数  $p, q$  と 整数  $n, e, d, k$  の6つとなるが、公開するのは  $\langle n, e \rangle$  の対のみとする。
- $n$ だけの情報では因数  $p, q$  を求めるのが困難で、かつ  $e$ だけの情報では最後のステップ4.で計算した  $\langle n, d \rangle$  の対の推定が困難である。

##### 送信者の暗号文生成手順

- $x$ を平文とする

2.  $y = x^e \bmod n$  を求める (公開鍵情報 $<n, e>$ に基づいた暗号化)
3.  $y$ を送信する

受信者の復号手順

1.  $y$ を受信する
2.  $x = y^d \bmod n$  を求める (秘密鍵情報 $<n, d>$ に基づいた復号化)
3.  $x$ が平文として得られた

### 2.3.3 計算例

鍵生成

1.  $p=3, q=11$  の場合,  $p \times q = 33$ .  $(p-1)(q-1) = 20$  と互いに素な数として  $e=7$  を取る.
2.  $7 \times d = (p-1)(q-1) \times k * 1$  を満たす  $d=3, k=1$  を取る.
3.  $n=33, e=7$  を公開する

暗号化 4. 平文  $x=6$  とすると,  $y = x^e \bmod n = 6^7 \bmod 33 = 279936 \bmod 33 = 30$  (これが暗号文)

復号化 5. 暗号文  $y=30$ を受信し,  $x = y^d \bmod n = 30^3 \bmod 33 = 27000 \bmod 33 = 6$  (平文が得られた)

量子コンピュータの出現と計算量的安全性の関係について

- 量子コンピュータを使うと, 巨大な素数の因数分解が高速にできる可能性があるので, RSA暗号化方式が危殆化してしまう.
- 量子耐性を持つような暗号化方式 (楕円曲線暗号など) が研究されている

## 2.4 ハッシュ (一方向性関数)

### ハッシュ値\*

ハッシュ関数は特定のアルゴリズムを用いて任意のメッセージを短い固定長のデータ（ハッシュ値）に変換します。二つの情報のハッシュ値を比較することで、元のメッセージの同一性（内容が同じであること）が検証できます。情報セキュリティでは改ざん検出など完全性の検証に利用されており、SHA-256などが広く使われています。

ハッシュ関数に求められる代表的な性質を以下に示します。

- 同じメッセージに対して同一のハッシュ値が得られる（決定性）
- 同一のハッシュ値が得られる（意味のある）メッセージを作成することが不可能（耐衝突性）

以下に類似した情報（文字列）の SHA-256 ハッシュ関数のハッシュ値を示しました。それぞれハッシュ値は大きく異なり、元の情報が異なることを容易に検出できます。

元データ	SHA-256 ハッシュ値
"Hello world"	1894a19c85ba153acbf743ac4e43fc004c891604b26f8c69e1e83ea2afc7c48f
"Hello world" 最後に空白あり	1bece7cd4104d3cba6e58a41e6aaef6474b6a5e1fa45adb97739b8623af0f2c4a
"Hello World" W が大文字	d2a84f4b8b650937ec8f73cd8be2c74add5a911ba64df27458ed8229da804a26

大きなデータの比較には時間を要しますが、データの小さなハッシュ値の活用で効率的に同一性を検証できます。

- 過去のデータが失われてもハッシュ値さえ保存されていれば改変を検出できます。
- 膨大な数のファイルについて改変場所を特定が必要な場合でもハッシュ値が異なるファイルのみを取り出し比較するといった方法で、作業が効率化できます。

(\* )ハッシュ関数は暗号ではありませんが、暗号技術と組み合わせて利用されることが多いためここで取り上げます。



## 2.5 電子署名（デジタル署名）

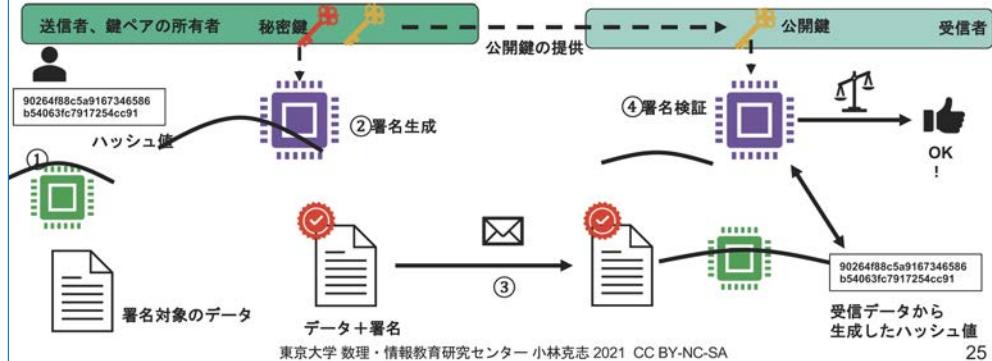
### デジタル署名

受信者がメッセージの出所を確認できるように付加されるデータを指し、真正性、完全性を維持します。署名生成に所有者以外に知らない秘密鍵を、署名検証に誰にでも公開してよい公開鍵を利用します。公開鍵で検証に成功する署名生成には秘密鍵が必要なため、署名者のはりすましを検出できます。マルウェアによるリスクを下げる目的でソフトウェアの配布元の検証にも用いられています。

デジタル署名処理の流れは以下のようになっています。

1. 署名対象のメッセージをハッシュ関数に与えハッシュ値を求めます。
2. ハッシュ値から秘密鍵を利用して署名を生成します。
3. メッセージと署名を併せて送ります。
4. メッセージから生成したハッシュ値、署名、公開鍵を利用して署名を検証します。  
検証に成功すれば、メッセージの出所は公開鍵の所有者であることを確認できます。

課題：この方式でも公開鍵の真正性が問題になります。公開鍵が本人になりましたものから提供されていないことを検証する必要があります。



### RSAによる電子署名手順

- RSA暗号化方式の場合には平文を公開鍵で暗号化していた
- 電子証明を行う場合は、暗号鍵と復号鍵の役割を逆にして用いる。つまり、暗号鍵<n, d>の方を公開する

### 発信者の電子署名手順

1. xを平文とする
2.  $y = x^e \text{ mod } n$  を求める（秘密鍵情報<n, e>に基づいた暗号化）
3. x, y の対を送信する（xが書類、yが書類xに対して公開暗号鍵を用いた署名）

### 受信者の署名の検証手順

1. x, y の対を受信する
2.  $y^d \text{ mod } n$  を求めて（公開鍵情報<n, d>に基づいた復号化）これがxと一致するか検証する
3. 一致ていれば署名が正しい（送信者が正しい）ことがわかる

## 2.6 公開鍵基盤(Public Key Infrastructure:PKI)

## 公開鍵基盤(Public Key Infrastructure:PKI)

公開鍵と所有者の真正性を保護する仕組みとして広く利用されています。公開鍵暗号方式やデジタル署名で課題とされた公開鍵がほんとうの所有者から提供されたことを証明書によって検証します。

### PKIの用語と役割：

証明書：公開鍵と所有者情報（名前、会社名など）が含まれており、認証局がその情報を信頼できることをデジタル署名によって証明したデータ

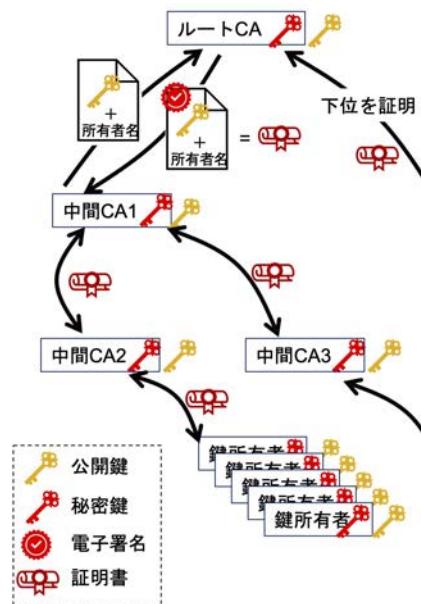
認証局（Certification Authority : CA）：公開鍵と所有者情報を信頼できることを確認し、証明書を発行します。CA自身の証明書を他のCAから発行してもらうこともあります。この場合、発行元を上位CAとする中間CAと呼ばれます。

ルートCA：他のCAから証明書を発行されない特別なCA、信用点（トラストアンカー）とも呼ばれる。証明書はルートCA自身が署名し、自己署名証明書と呼ばれます。

CAの信頼関係を木構造（右図）で構成し、最上位のルートCAだけ信頼できれば、信頼がつながっている下位の証明書、すなわち公開鍵と所有者情報を信頼してよい。

ルートCAの証明書はOS、Webブラウザなどにあらかじめ組み込まれておらず、組み込まれたルートCAにつながる中間証明書であれば誰から提供されたものでも信頼して良い。このため、利用者はPKIの知識や、ルートCAの導入といった手順なしにPKIが提供する仕組みを享受できる。

(\*) 信頼性の確認にはさまざまな方法でおこなわれています。登記簿・電話確認が必要な厳格なものや、オンラインだけで完了する簡易なものまであります。



東京大学 数理・情報教育研究センター 小林克志 2021 CC BY-NC-SA

26

## PKI：電子署名の公開鍵に対する認証局（Authority）の存在

発信者のなりすましを防ぐために、発信者の実在性を担保した公開鍵の認証を行う「認証局」と、発信者・受信者とのプロトコルが必要になる。これはPKI（公開鍵認証基盤）としてインターネットに整備されている。

### 発信者の公開鍵登録

1. 認証局へ、自分の公開鍵（復号鍵）と、ID情報（通常はドメイン）を登録する

認証局の承認 2. 発信者の登録情報、実在性をチェックした上で、申請を認める

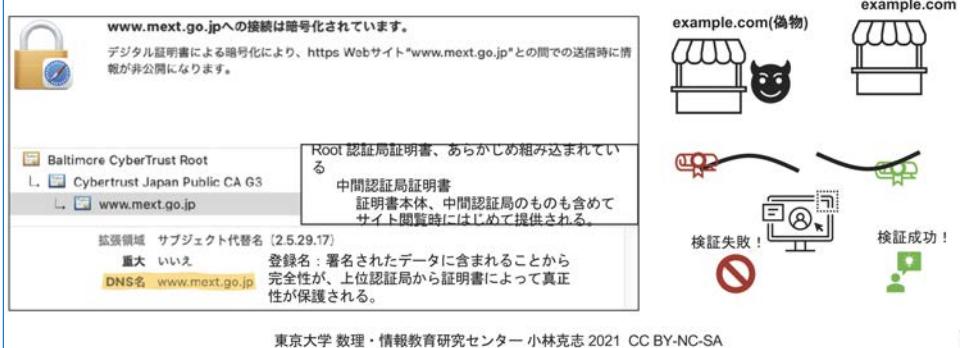
受信者の署名の確認 3. 認証局の正しさは、WebブラウザやOSなどの基盤ソフトウェアに予め書き込まれている。4. 受信した署名の正しさの検証を行う際、公開鍵の正しさを認証局へ問い合わせせる。

## PKIの応用：Web サーバの真正性、通信の秘匿性

- Web サーバがアクセスした URL(\*) の登録名のものかどうかの検証に利用されています。
- Web サーバが提供する証明書について Root 認証局からの信頼関係が確認できない場合、や有効期限が切れている場合は、ユーザーに警告し、閲覧への注意を促します。
  - 証明書の公開鍵を利用してその後の通信は暗号化され、通信内容は秘匿されます。

左下の図は文部科学省トップページ (<https://www.mext.go.jp>) の証明書の内容と信頼関係を表示させています。

(\*) URL(Uniform Resource Locator) : Web ブラウザなどで利用されるインターネット上の資源を特定する <https://example.com/sample.html> といった形式の文字列のこと。登録名は example.com が該当する。



27

## Python PyCryptodomeライブラリを用いたRSA暗号化・復号化処理の例

PKCS #7 による暗号化・署名処理

参考：6 章 セキュリティプロトコル - 電子情報通信学会知識ベース

SSL/TLSインターネットコネクションの暗号トンネル方式

参考：GMO | SSLのバージョンについて

In [7]: `## (step 1) 密密鍵・公開鍵の作成【本来は受信側の準備プロセス】`

```
# RSAパッケージをインポート
from Crypto.PublicKey import RSA

# 密密鍵の生成とファイルへのエクスポート
key = RSA.generate(2048) # 鍵長 n が 2048bit
private_key = key.export_key()
file_out = open("private.pem", "wb")
file_out.write(private_key)
file_out.close()

# 公開鍵の生成とファイルへのエクスポート
public_key = key.publickey().export_key()
file_out = open("public.pem", "wb")
file_out.write(public_key)
file_out.close()

# 作業用フォルダに、確かに private/public.pem ファイルが保存されているか確認する(中身も見てみ
# 実際のサーバSSL環境では,public.pemを認証局へ登録する作業が伴う
##!ls -al
##!cat private.pem
!cat public.pem
```

```
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEAAQ8AMIIBCgKCAQEAgEtd0sSu8mIjMIA8YX/Y
ABbWpV9yTNE13FizRFdq7yUxsnK/4pbk0eviyDqXbW22ZMaXSuoPF9NqSjM00FM0
50462i9LnC0jZjPI/LazvjIqvjFKqBjEMs17yAbkLJQ7WP6SDKZNrspDR2F0l8oI
eU5Xtm6FNoFmbAv0qEclIRic15Jo2m1HVL17bc10kCCfZBbpqrdrQKUfBGlm2prFd
gXb0biK7cqQFJGuxAFyiQUX50ldheMYHQ7LPeAf4FicZUNH3ynUBoBAfBBzvXUht
GJ7Iy06vP45IlJv0fy7fVdoiL7FQzMzTdctFEjBewDXh8ZTfVoxBXo4/4ke2Ix0L
6wIDAQAB
-----END PUBLIC KEY-----
```

```
In [5]: ## (step 2) 公開鍵を使ったデータの暗号化【本来は送信側のプロセス】
```

```
# RSAと乱数生成,RSAのパディング PKCS/0AEP, データ暗号化のためのAESパッケージをインポート
# 参考:https://qiita.com/kj1/items/aebbb73a034f36d73e40
from Crypto.PublicKey import RSA
from Crypto.Random import get_random_bytes
from Crypto.Cipher import AES, PKCS1_0AEP

data = b"Hello, RSA World!" # 平文 暗号化する文字列
file_out = open("encrypted_data.txt", "wb") # 暗号化文を格納するファイル(本来は送信側)

recipient_key = RSA.import_key(open("public.pem").read()) # 公開鍵の設定(実際は受信側)
session_key = get_random_bytes(16) # セッション鍵を生成

# セッションキーをRSA公開鍵で暗号化する
cipher_rsa = PKCS1_0AEP.new(recipient_key)
enc_session_key = cipher_rsa.encrypt(session_key)

# データをAESセッションキーで暗号化
cipher_aes = AES.new(session_key, AES.MODE_EAX)
ciphertext, tag = cipher_aes.encrypt_and_digest(data)

# RSA公開鍵暗号済セッションキー, AES共通鍵暗号のnonce/タグ値, および AESセッションキーで暗号化したデータを格納する
[file_out.write(x) for x in (enc_session_key, cipher_aes.nonce, tag, ciphertext)]
file_out.close()
```

```
In [ ]: ## (step 3) 密鑑を使ったデータの復号化【本来は受信側のプロセス】
```

```
# RSA, RSAのパディング PKCS/0AEP, データ復号化のためのAESパッケージをインポート
# 参考:https://qiita.com/kj1/items/aebbb73a034f36d73e40
from Crypto.PublicKey import RSA
from Crypto.Cipher import AES, PKCS1_0AEP

private_key = RSA.import_key(open("private.pem").read()) # 密鑑の設定(本来は受信側)

file_in = open("encrypted_data.txt", "rb") # 暗号化文を読み出すファイル(本来は受信側)
# RSA公開鍵暗号済セッションキー, AES共通鍵暗号のnonce/タグ値, および AESセッションキーで暗号化したデータを格納する
enc_session_key, nonce, tag, ciphertext = \
    [file_in.read(x) for x in (private_key.size_in_bytes(), 16, 16, -1)]

# セッションキーをRSA密鑑で復号する
cipher_rsa = PKCS1_0AEP.new(private_key)
session_key = cipher_rsa.decrypt(enc_session_key)

# データをAESセッションキーで復号する
cipher_aes = AES.new(session_key, AES.MODE_EAX, nonce)
data = cipher_aes.decrypt_and_verify(ciphertext, tag)

print(data)
## 暗号文 encrypted_data.txt を適当に変更(メッセージ改竄)すると、正常に復号できなくなることを確認
## また、同じ公開鍵・密鑑であっても、別のセッションキーでAES暗号化したメッセージは、元のセッションキーで復号されない
```

### 3. ユーザ認証とアクセス管理

#### 3.1 アクセス制御

##### アクセス制御

情報および情報処理施設へのアクセス（接触・接続）を認可及び制限する手段を提供します。秘匿性・完全性・可用性すべての特性に関わります。

情報システムのアクセス制御によって実現される機能の例を示します。

1. 他人を利用者登録できない。（完全性）
2. 他人のファイルを読めない。（秘匿性）
3. 他人のパスワードを変更できない。（完全性）
4. 他人のファイルを削除できない。（完全性、可用性）  
（）は直接維持できる特性を示しますが、行為が許されると他の特性に影響がおよびます。例えば、「3. 他人のパスワード変更」できると、他人へのなりすましによって秘匿性・可用性に影響する「2. ファイル閲覧」や「4. ファイル削除」も可能になります。

アクセス制御の例：

- WindowsなどのOSではファイルごとに操作の許可/禁止を設定できます。例えば、他のユーザに読み出しが許可、書き込みが禁止といったアクセス制御を設定できます。
- ICカードや写真つきIDカードを用いた入構・入室管理もアクセス制御に含まれます。

利用者ごとに制御を変える場合は、利用者ごとに真正性を確認する認証も必要となります。

##### 情報セキュリティの7要素

3要素（機密性、完全性、可用性）に加えられた4つの新たな概念

- 信頼性：システムが正しく動作する
- 真正性：アクセスする人や団体が許可を得ている（機密性の一部が独立している）
- 責任追跡性：アクセスする人や団体の操作を追跡できる
- 否認防止：あとから覆されない

#### 3.2 ユーザ認証

##### ユーザ認証

主体が正しく権限を持っているかを確認する

- 主体（エンティティ）：人、組織、機械

機密性の一部である「真正性」の確保を目的としている

##### ユーザ認証の3つの方法

- 知識認証（ID/パスワード、秘密の合言葉など）
- 物理認証（Felicaカードなど）
- 生体認証（指紋認証、静脈認証など）

## 認証 1/2

相手の真正性を検証する。例えば、ネット通販事業者は購買者が登録された利用者本人かどうかを検証し、なりすましによる未払いなどの被害を防ぐ必要があります。

利用者認証で代表的なものを挙げます。

- ID / Password : 利用者が入力する ID(識別子, Identifier) とパスワードが、登録されたものと一致するかを検証する。ID は重複を防ぐため電子メールアドレスなどが使われる。
- SMS, 電話, メール認証 : 利用者が ID を入力すると、登録された連絡先にランダムに生成した情報が送られる。利用者がその情報を正しく入力できるかどうかを検証する。パスワード再設定用の URL が送られる方式も含まれる。
- 証明書 : 利用者が署名した電子署名を(PKI で説明した)証明書で検証する。秘密鍵・証明書は IC カードで保存される。秘密鍵の利用時には持ち主に暗証番号(文字列)を入力させる。この暗証番号は Personal Identification Code(PIN) コードとも呼ばれる。
- 生体認証 : 人間の身体的な特徴や行動的な特徴を利用する認証方式のこと。指紋・静脈・虹彩を利用する方式が使われている。
- 認証器 : 利用者が持つデバイスから提供される一定間隔で更新される文字列を入力させる。認証器はトークンと呼ばれる物理デバイスやスマートフォンアプリケーションとして提供される。ネット銀行の振り込みなどで利用されている。

## 認証 2/2

ID / Password は複数のサービスで使いまわされることが多く、なりすましのリスクが高い。実際なりすましによる被害も大きくなっている。最近では、最初の認証に成功したのち、さらに別の認証を求める 2 段階認証も使われ始めている。

第一段階で ID/Password 認証を要求し、2 段階目で認証器による認証をもとめる方式が使われ始めている。

### 知識認証（いわゆるID/パスワード方式）

システム内部にはパスワードの設定値が保存されない

- 設定値は、一方向性関数でハッシュ化されて保存されることが多い
- 一方向性が考慮されていないシステムは、パスワード情報をハッシュ化しないでデータベースに保存している

⇒非常に危険で、大規模なパスワード漏洩事故を起こすと甚大な被害が発生する

### 利用者認証の手順

1. 利用者が入力したパスワード文字列を、ハッシュ化する
2. システム側で保存してある設定値と同じであるなら、パスワードが正しいものとして認証する

## 2段階認証方式 (2FA)

### 2段階1要素認証方式

- 第1段階：知識認証（ID/パスワード）
- 第2段階：知識認証（秘密の合言葉など）※重要な操作、定期的に実施する

### 2段階2要素認証方式

- 第1段階：知識認証（ID/パスワード）
- 第2段階：物理認証（当人が持っているデバイス（スマホなど）のSMS/メールに認証コードを送って入力させる）

2要素認証を組み合わることによって、なりすましのリスクは下げることができる。

- 物理認証 × 知識認証（ICカードをかざして、次にパスワード・パスコード入力）※クレカなどによくある方式
- 生体認証 × 知識認証（静脈認証を通過した後、次にパスワード・パスコード入力）※銀行ATMによくある方式
- 物理認証 × 生体認証（ICカードをかざして、その後に指紋リーダーを通過）※機密性の高いビルの入退出管理によくある方式

## 4. セキュリティホールと脆弱性への対応

### 4.1 セキュリティホールと脆弱性

#### セキュリティホール

- コンピュータの動作上、情報セキュリティを損なう可能性がある(OSなど基盤ソフトウェア上の)動作

脆弱性は、より広い概念となる

- セキュリティホール
- 人（組織）の問題
- ソフトウェアバグ

### 4.2 セキュリティホールと脆弱性への対応

#### ソフトウェアのアップデート

- ソフトウェアは人間が作成したもので、完全ではない
- 不完全なところ、セキュリティホールは発見され次第、逐次更新される（セキュリティ・アップデート）
- 大規模な更新が行われる場合、古いバージョンのソフトウェアは更新されなくなる（メジャー・アップデート）

## セキュリティホールと脆弱性への対応

- Windows 8, 8.1 は、2023.1.10にセキュリティ延長サポートが終了する
- Windows 10 は 2025.10.14にセキュリティ延長サポートが終了する

## セキュリティ・アップデートを行わなかった場合

- 発見されたセキュリティホールを攻撃される
- 通常はセキュリティホールが発見されても、広く知られることがない
- OSやアプリの製作者がアップデートを公開するまで、セキュリティホールが知られないことが多い（ゼロデイ攻撃）

## 4.3 ゼロデイ攻撃、標的型攻撃

### ゼロデイ攻撃とは

ソフトウェアなどのセキュリティホールが発見されてから、その情報公開や対策が講じられる前に、そのセキュリティホールを狙う攻撃のこと。脆弱性発見から日にちを空けない攻撃が名前の由来である。

[参考:IPA情報セキュリティ10大脅威にみるセキュリティリスクー内在する脆弱性を悪用したゼロデイ攻撃とはー](#)

### 標的型攻撃とは

特定の組織や人を狙って行われる標的型サイバー攻撃は、近年大きな脅威となっています。ソーシャルエンジニアリング手法を駆使した標的型攻撃メールや、セキュリティソフト等による検知を回避し侵入の痕跡を巧妙に隠蔽しながら活動するマルウェアなど、手口や技術も年々高度化しています。

> IPAでは、標的型サイバー攻撃による被害拡大防止のため、標的型サイバー攻撃に関する情報やガイド、企業・組織向けの相談窓口や対応支援、情報共有の仕組みの提供など、様々な取組みを行っています。

[参考:IPA情報処理推進機構 | 標的型サイバー攻撃対策](#)

**memo**