A04:Cryptographic Failures

今回やること

- 暗号とは
- 共通鍵方式
- 公開鍵方式
- ハイブリッド方式
- ハッシュ関数
- ディジタル署名
- セキュリティプロトコル
- 前回の補足

暗号とは

暗号

暗号とは、暗号化されたデータを見ても特別な知識なしでは意味のある情報として読めないように変換する表記法（アルゴリズム）、もしくは表記（データ）のこと

暗号化アルゴリズムは、共通鍵暗号化方式と、公開鍵暗号化方式の二つに分類される。これらを組み合わせたハイブリッド暗号化方式もある。

ケルクホフスの原理(参考)

ケルクホフスの原理

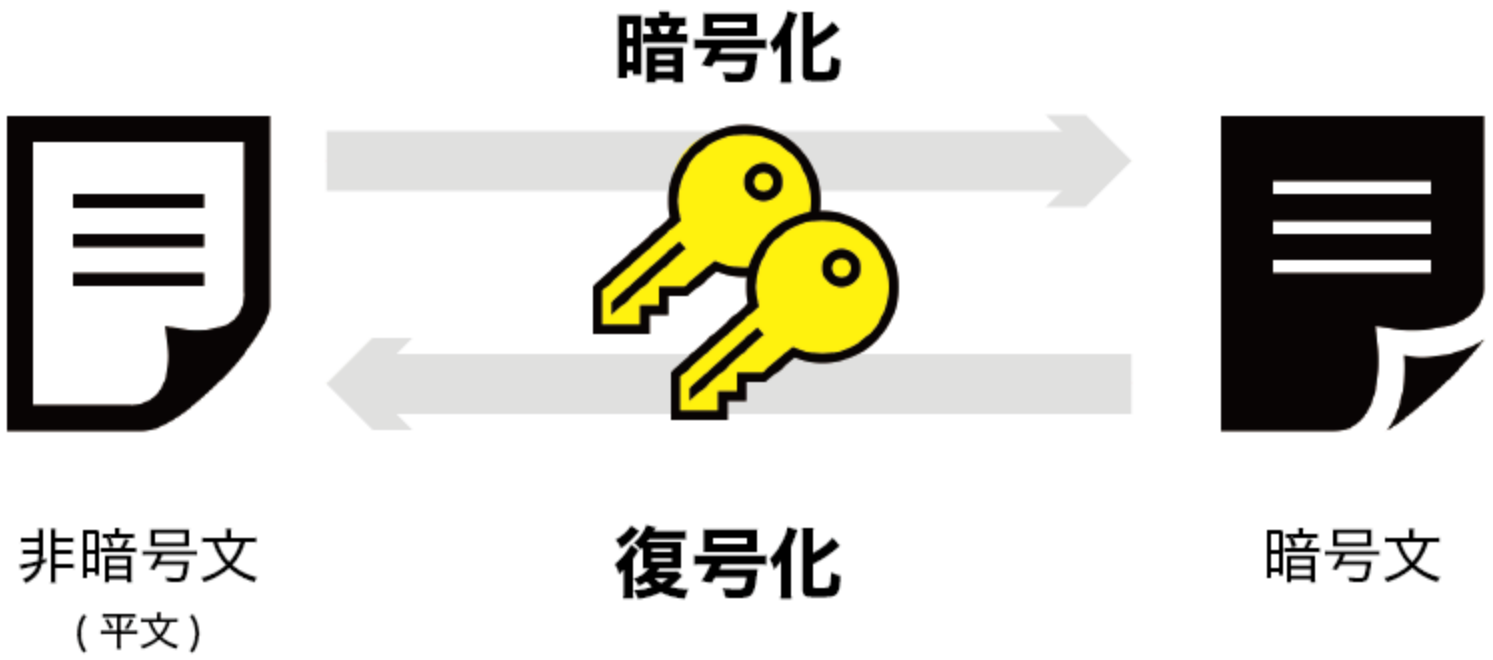
暗号システムは、暗号鍵以外の全てが公知になっていたとして、なお安全であるべきであるという設計理念。

秘匿によるセキュリティと対極にある概念。

共通鍵方式

共通鍵方式

暗号化鍵と復元鍵が同一であるアルゴリズム。



共通鍵暗号方式

- n人でやりとりする場合、n(n-1)/2本の鍵が必要になるため、（1人増えるごとに必要な鍵がn-1本増えるから）暗号化鍵の安全な共有が大きな問題となる。
- ブロック方式とストリーム方式に大別できる。ブロック方式は、鍵生成部とデータ攪拌部により構成され、暗号化をブロック単位で行う。代表的なものとして、DESやAESがある。ストリーム方式は、擬似乱数生成器が生成する乱数を使い、入力された平文をビット単位で暗号化する。代表的なものとして、RC4が存在する。
- 現在は、暗号化アルゴリズムとしてAESの利用が推奨されている。これまで広く利用されてきたDES3やRC4の利用は推奨されていない。

AES

AES

AES（Advanced Encryption Standardの略）は、2001年にFIPS 197として策定されたブロック暗号化方式。

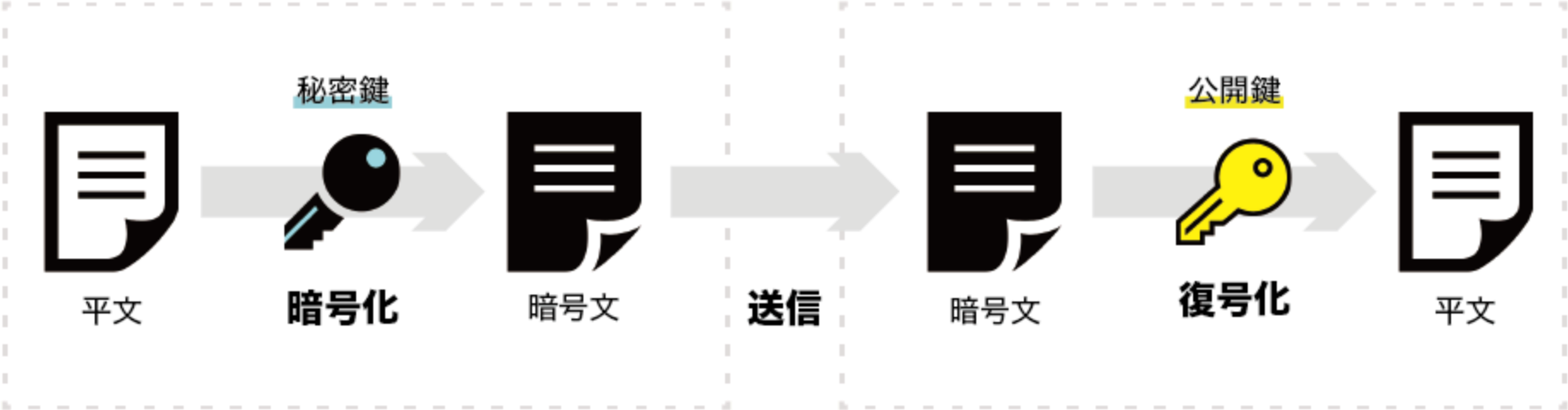
- FIPS: NISTによって発行された標準及び指針。

インテル社製CPUにはデータ撹拌の一部分がAES-NIとしてハードウェア化されており、高速に処理できる。

公開鍵方式

公開鍵方式

暗号化鍵と復号鍵を別のものとする暗号化方式。代表的なものとしてRSAがある。



RSA

RSA

RSAでは、まず対となる公開鍵と秘密鍵を生成し、公開鍵を配布する。公開鍵を用いて平文を暗号化し、秘密鍵を用いることで復号を行うことができる。秘密鍵を知らずに暗号文を復号することは困難であるため、暗号化鍵を公開することができる。

- RSAの安全性はコンピュータの計算性能や素因数分解の困難性に依存している。
- RSAでは、共通鍵方式に比べて鍵のやりとりを厳密に行う必要がないというメリットがある。一方で、鍵長（データの大きさ）が大きいほど処理速度が遅いというデメリットもあるため、これらの方式を組み合わせたハイブリッド方式も存在する。

RSAの具体例

1. 正数p,qを決定する。
2. n = (p-1)(q-1)と互いに素な正数eを決定する。
3. ed ≡ 1 (mod(p-1)(q-1))となるdを決定する。
4. n,eを公開鍵とし、dを秘密鍵とする。p,qは破棄する。
5. 平文をMとすると、以下のようにして暗号化と復号を行える。
    - 暗号化 $C = M^e \quad (mod n)$
    - 復号 $M = C^d \quad (mod n)$

RSAの演算の例

1. 鍵の生成
    - p=5, q=11とする。
    - n=55となり、これと互いに素な正数3を選ぶ。
    - ed=1(mod 4*10)となる正数dとして27を選ぶ。
    - 公開鍵をしてe=3とn=55, 秘密鍵として27する。
2. 平文として７とする。
    - 暗号化 $7^3 \equiv 13 \quad (mod 55)$
    - 復号　$13^{27} \equiv 7 \quad (mod 55)$
3. このように、平文の７が暗号文から復元できる。

ハイブリッド暗号

ハイブリッド暗号

RSAは鍵長が大きいほど処理速度が遅くなるため、平文全てをRSAで暗号化することはしない。一般的には、平文をAESで暗号化し、その暗号鍵のみをRSAで暗号化する。このような暗号方式をハイブリッド暗号と呼ぶ。この時、一時的に用意する暗号鍵をセッション鍵と呼ぶ。

- SSL/TLSやIPsecなどのセキュリティプロトコルで用いられる鍵交換では、このハイブリット暗号が採用されていたが、PFS(Pefect Forward Securityの略。前方秘匿性(FS)が完全(Pefect)であるということ)が確保できないとの理由から、TLS1.3からはRSAを用いた鍵交換は利用できなくなった。これを解決する鍵交換アルゴリズムとしてDiffie-Hellmanアルゴリズム(DHE)が存在する。

前方秘匿性

前方秘匿性(FS)

前方秘匿性(Forward Security)とは、ある時点の秘密鍵や共通鍵が知られたとしても、その共通鍵を使った通信内容以外は解読されないようにする性質。
ハイブリッド暗号による鍵交換ではFSが実現されないため、DHEやECDHEがあり、TLS1.3における暗号スイートに採用された。(ECDHEは楕円曲線DHを用いた動的な鍵共有方式)

ハッシュ関数

ハッシュ関数

ハッシュ関数とは、任意の長さのデータを固定長のデータに圧縮する、次の性質をもつ関数$hash$のこと。

- 1方向性(現像計算困難性): 出力値から入力値を発見することが困難であること。すなわち、hが与えられた時、$h = hash(m)$となるようなmを見つけるのが困難でなければならない。
- 第2原像計算困難性: ある入力値と同じハッシュ値となるような別のハッシュ値を求めることが困難であること。すなわち、mが与えられた時、$hash(m) = hash(m')$となるようなm'を求めるのが困難でなければならない。
- 衝突困難性: 同じ出力値を生成する2つの入力値を発見することが困難であること。すなわち、$hash(m1) = hash(m2)$となるような$m1, m2$を見つけるのが困難でなければならない。

ハッシュ関数への攻撃

現像攻撃

現像攻撃: 与えられたハッシュ値から元の入力値を探索すること。一般に、nビットのハッシュ関数に対して、$2^n/2$の計算量が必要になる。

衝突攻撃

衝突攻撃（誕生日攻撃）: 与えられた入力値$m1$に対して、$hash(m2) = hash(m1)$となるような別の入力値$m2$を探索すること。誕生日のパラドクスにより、$2^{n/2}$の計算量が必要になる。(誕生日のパラドクス：あるグループにおいて、自分と同じ誕生日の人間がいる確率よりも、グループ内に同じ誕生日の人間がいる確率はずっと大きいという直感に反しているという意味でのパラドクス。)

衝突耐性

現像攻撃への耐性を耐性を弱衝突耐性、衝突攻撃への耐性を強衝突耐性という。

ハッシュ関数の具体例

- MD4,MD5,SHA,SHA-1,SHA-2,SHA-3などが存在する。
- MD4,MD5,SHA,SHA-1はすでに安全性が疑問視されており、NISTのガイドラインによって利用を避けることになっている。SHA-3は、次世代の012年10月に次世代のハッシュ関数に選定された。2023/03/23の記事では、Windows11がSHA-3に対応していると発表された。
  (元記事)

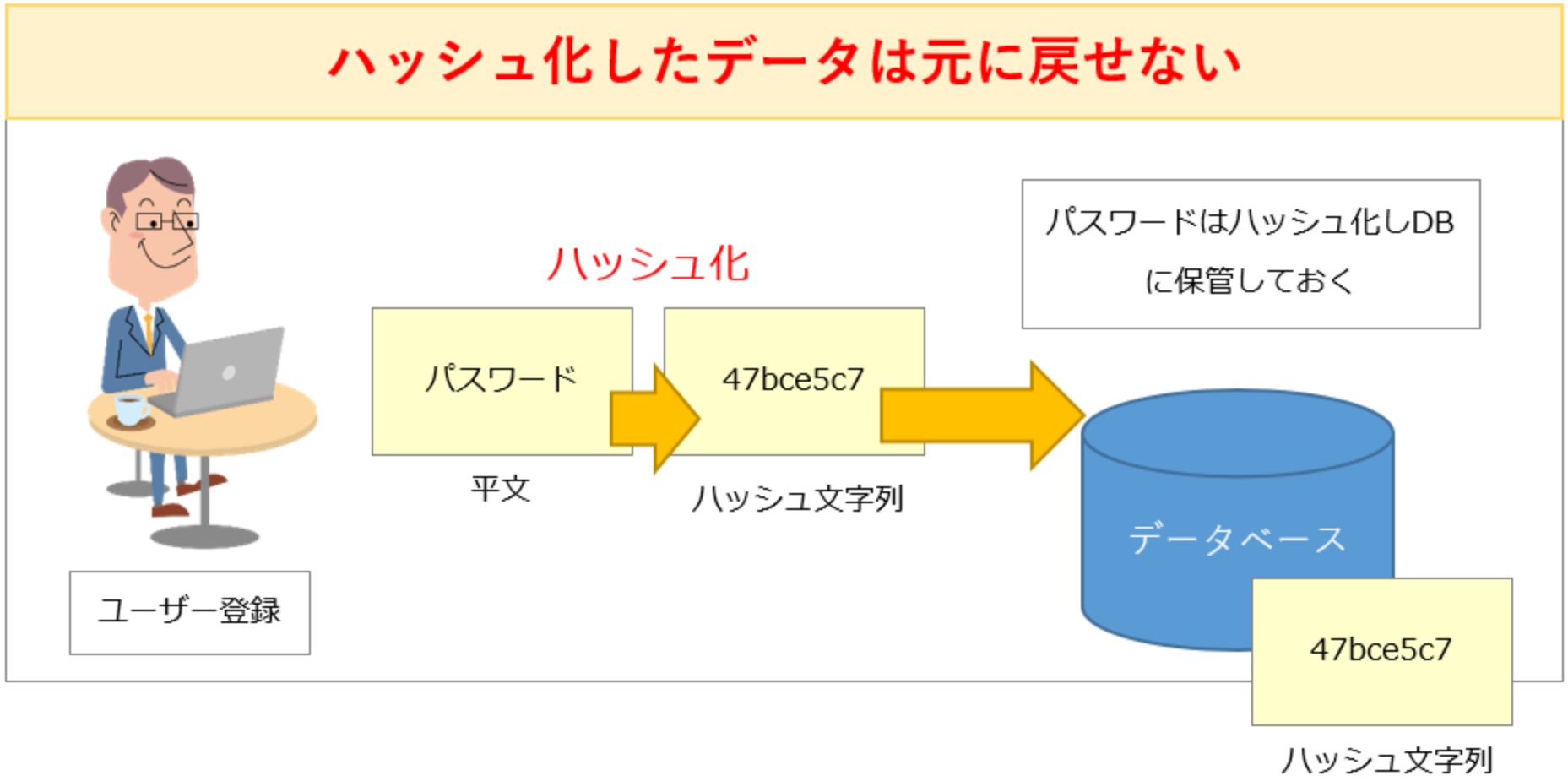
ハッシュ関数の脆弱性

- MD4,MD5は強衝突耐性は容易に突破されることが報告されている。これは、理論上は、あらかじめ同じハッシュ値になるメッセージを二つ用意しておけば、途中で署名されたとしても、すり替えることが可能であることを意味する。
- 2005年2月に、SHA-1に対して効率的に総当たり攻撃を仕掛けることができるとの記事が公開された。(元記事1)これは本来の総当たり攻撃より、約2000倍も高速な攻撃方法であった。これを受け、NISTは2022年にSHA-1の廃止を宣言した。(元記事2)

ハッシュ関数の利用

パスワード認証におけるハッシュ関数の利用

サーバーやOSのパスワード認証では、サーバーやOSに平文のパスワードを保存するのではなく、ハッシュ値で比較することで、安全性を高めることができる。



ハッシュ関数の利用

ハッシュ関数による改竄検知

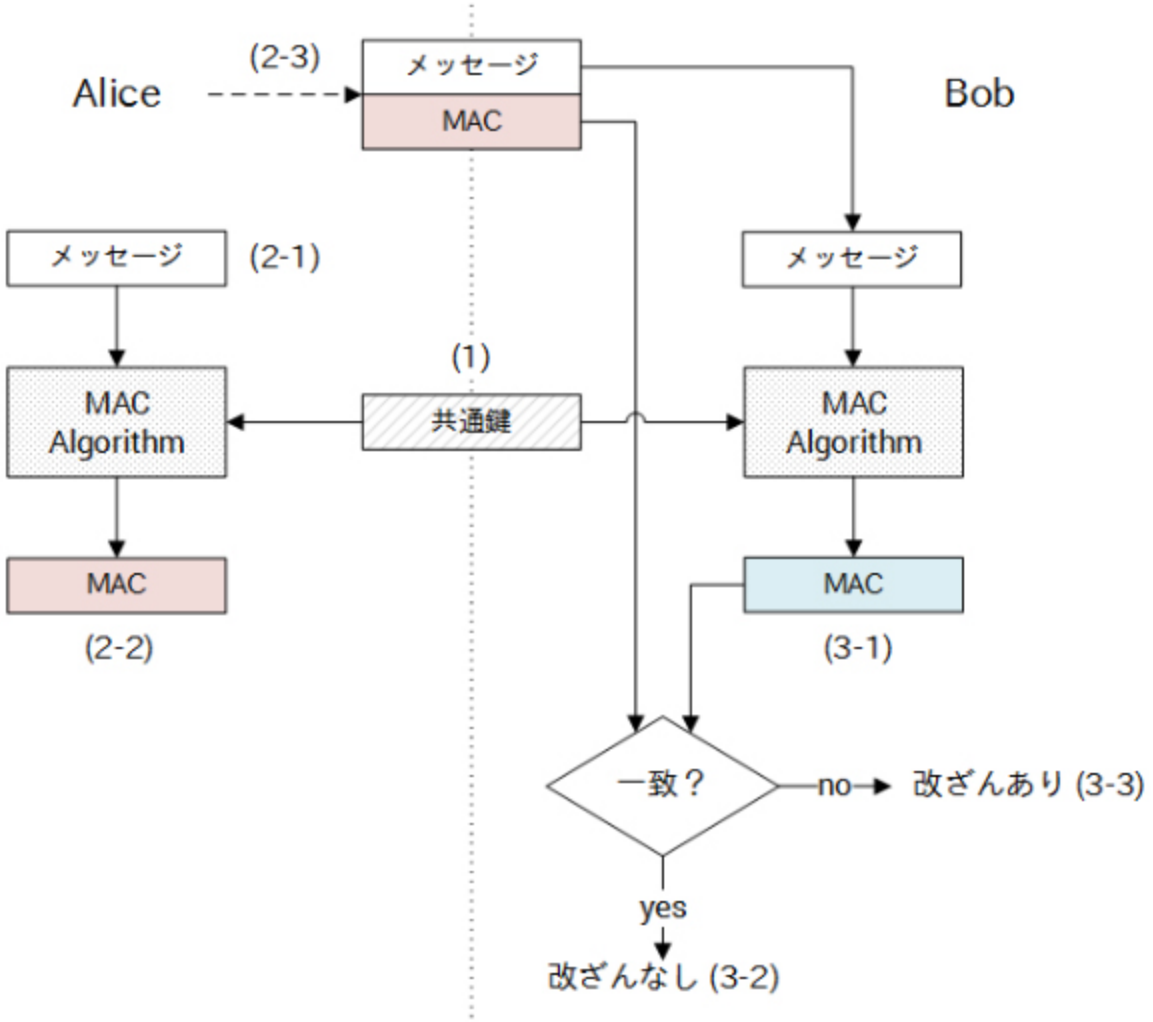
改竄の検知をしたいファイル群のハッシュ値をあらかじめ計算しておき、それを安全に保管しておく。ファイルの改竄があるか否かは、元のハッシュ値を一致するかを調べることで確認できる。

ハッシュ関数の利用

メッセージの改竄検知

ハッシュ関数は、MACというメッセージの真正性を保証するための仕組みにも用いられている。これは、送信するメッセージmと共通鍵暗号化方式の暗号鍵Kとにハッシュ関数$hash$を適用して作るMAC値と、メッセージmをあわせて送信することにより、メッセージmの真正性（認証された通信相手や暗号鍵を共有している通信相手からのメッセージであること）や、完全性（ビット反転攻撃などにより暗号文自体が改竄されている時に、それを検知できること）を保証するものである。（ビット反転攻撃とは、暗号文のバイナリデータの一部の値をビット反転させ、平文を改竄すること。）

ハッシュ関数の利用


Alice
(2-3)
メッセージ
MAC
Bob
メッセージ
(2-1)
メッセージ
(1)
MAC
Algorithm
共通鍵
MAC
Algorithm
MAC
MAC
(2-2)
(3-1)
一致？
no
改ざんあり (3-3)
yes
改ざんなし (3-2)

ディジタル署名

ディジタル署名

ディジタル署名とは、メッセージに対するハッシュ値を秘密鍵で暗号化することで、メッセージの改竄検知、及び否認防止（発信したのに発信していないと否定すること）を実現する仕組みのこと。

ディジタル署名の仕組み

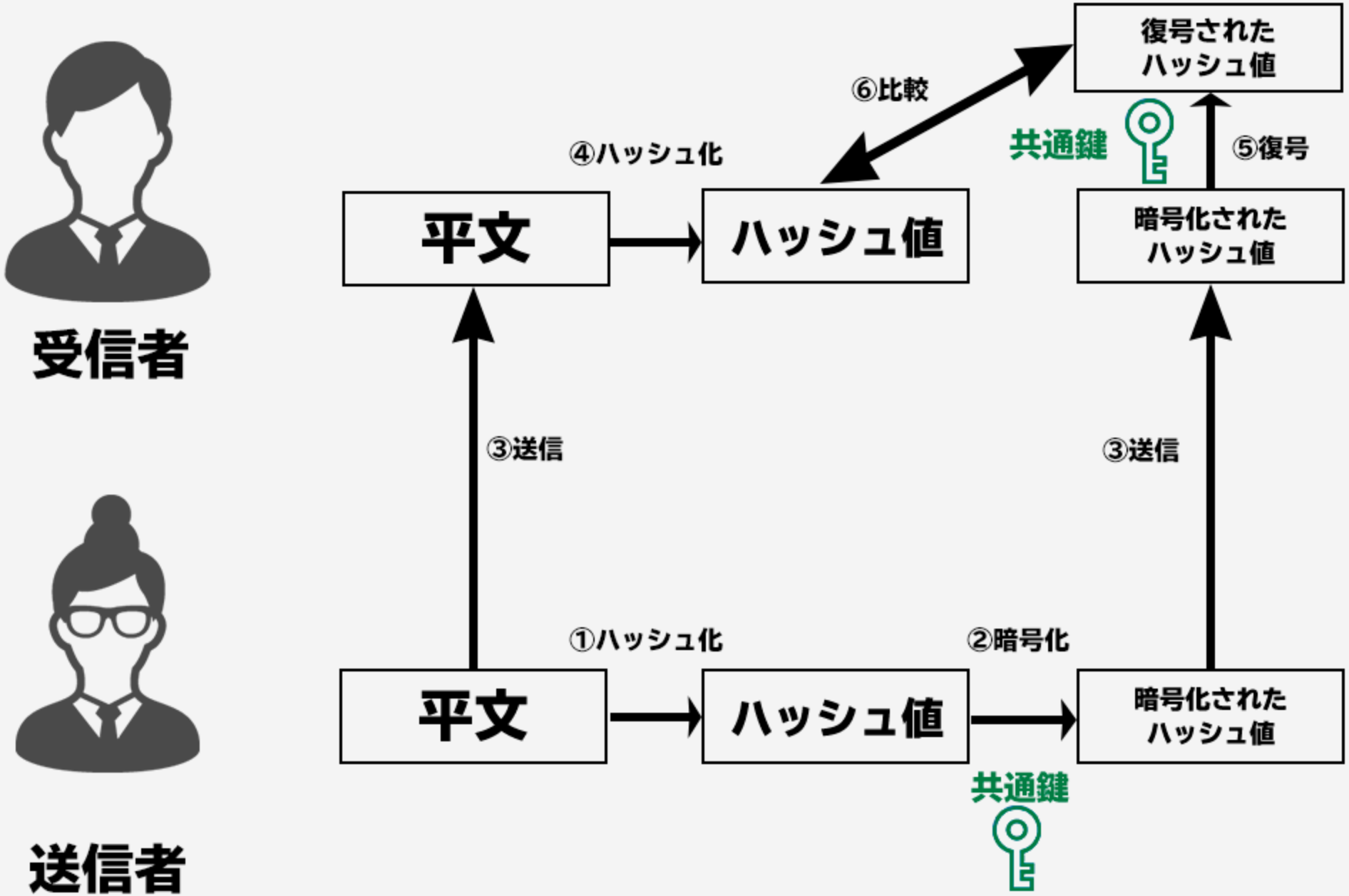
署名の作成フェーズ

1. 署名者が公開鍵Pと秘密鍵Sを用意し、公開鍵を検証者に安全に渡しておく。
2. ハッシュ関数を用いて、送信したい平文Mのハッシュ値Hを求める。
3. 署名者はあらかじめ用意しておいた秘密鍵Sを用いて、ハッシュ値Hを暗号化する。それを署名値Hsとする。
4. 平文であるメッセージMと署名値Hsをセットで検証者に送る。

ディジタル署名の仕組み

検証フェーズ

5. 検証者は、平文Mからハッシュ値H'を求める。
6. 署名者の発行した公開鍵Pを用いて署名値Hsを復号し、ハッシュ値Hを求める。ここで、秘密鍵Sで暗号化したデータはペアとなる公開鍵Pでしか復号できないことに注意。
7. 検証者は、署名から求めたハッシュ値HとメッセージMから求めたH'が一致するかどうかを確認する。
8. 一致した場合、メッセージMは署名者によって作成されたことを確認できる。


受信者
送信者
平文
ハッシュ値
暗号化された
ハッシュ値
復号された
ハッシュ値
平文
ハッシュ値
暗号化された
ハッシュ値
①ハッシュ化
②暗号化
③送信
③送信
④ハッシュ化
⑤復号
⑥比較
共通鍵
共通鍵

セキュリティプロトコル

セキュリティプロトコル

セキュリティプロトコルとは、通信における脅威(盗聴、改竄、ハイジャック、なりすまし、捏造、再送)を排除して、ネットワークにおける機密性、安全性、および真正性を確保することを目的とした通信プロトコルのこと。

暗号スイート

セキュリティプロトコルは、以下に示す技術を適切に組み合わせて構成されている。

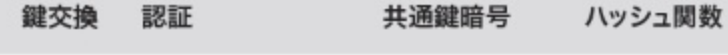
- 公開鍵暗号化技術
- 共通鍵暗号化技術
- 鍵交換方式
- メッセージ改竄通知
- 認証プロトコル

これらの組み合わせを暗号スイートと呼ぶ。

暗号スイート

鍵交換	暗号処理に使う鍵をWebサイトと端末の間で共有する
認証	本物のWebサイトであることを保証する
共通鍵暗号	端末とWebサイトの間の通信を暗号化する
ハッシュ関数	データの改ざんを検出するためのハッシュを生成する

TLSの暗号スイートの表記例



AES：Advanced Encryption Standard
ECDHE：Ephemeral Elliptic Curve Diffie–Hellman
GCM：Galois/Counter Mode
RSA：Rivest–Shamir–Adleman
SHA：Secure Hash Algorithm

SSL/TLS

SSL/TLS

SSL/TLSは、Web通信などの様々な暗号化で利用されるトランスポート層の**セキュリティ**プロトコル。
SSL(Secure Socket Layer)はTLS(Transport Layer Security)より古いプロトコルであり、これらを合わせてSSL/TLSと呼ばれることが多い。

- Web通信はHTTPというプロトコルで行われているが、HTTPでは平文で情報がやり取りされている。SSL/TLSを利用して通信を保護しているHTTPはHTTPSと呼ばれる。

これまでの話をふまえて前回のスライド

A04:Cryptographic Failuresとは

概要

- 暗号化技術を不適切に使用、または使用しないことにより、重要な情報の漏洩を引き起こすこと(かつてのカテゴリ名は「機微な情報の露出」だった)

具体例

- 古いまたは脆弱な暗号アルゴリズムやプロトコルを初期設定のまま、または古いコードで使っていないか。
- MD5やSHA1といった非推奨のハッシュ関数が使用されていないか。また暗号学的ハッシュ関数が必要とされる場合において、暗号学的でないハッシュ関数が使用されていないか。
- どんなデータであれ平文で送信していないか。これは、HTTP,SMTP,FTPといったSTARTTLSのようなTLS upgradesのプロトコルを使っている場合に該当する。外部インターネットへのトラフィックは危険である。また、ロードバランサー、Webサーバー、バックエンドシステムなどの内部の通信もすべて確認すること。

対策

- 最新の暗号強度の高い標準アルゴリズム、プロトコル、暗号鍵を実装しているか確認する。そして適切に暗号鍵を管理する。
- 前方秘匿性(FS)を有効にしたTLS、サーバーサイドによる暗号スイートの優先度決定、セキュアパラメータなどのセキュアなプロトコルで、通信経路上のすべてのデータを暗号化する。HTTP Strict Transport Security (HSTS)のようなディレクティブで暗号化を強制する。
- FTPやSMTPといったレガシーなプロトコルを機密データの伝送に使用しない。
- アプリケーションごとに処理するデータ、保存するデータ、送信するデータを分類する。そして、どのデータがプライバシー関連の法律・規則の要件に該当するか、またどのデータがビジネス上必要なデータか判定する。