第9回　保護とセキュリティ（2）

ネットワークセキュリティと暗号，認証

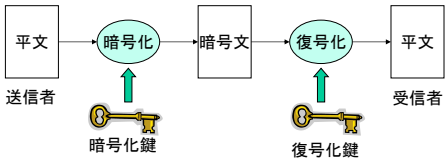
ネットワークセキュリティの必要性

- 不正アクセス
  - 利用権限を越えてネットワーク経由でシステムの利用を図る
  - 情報の窃盗，改ざん，システムの破壊
    - パスワードの管理を厳重に行う必要がある
    - ssh（セキュアシェル）：暗号を用いた安全な本人認証
- （意図的に作られた悪質な）不正プログラム
  - ウイルス：感染，潜伏，発症機能をもつ不正プログラム
  - ファイルなどに自身を付着させ（感染），条件が揃うまで動かず（潜伏），その後悪質な振る舞いをする（発症）
    - OSの保護機能，バグによる脆弱性の対処
    - 怪しいソフトは受け取らない，開かない．
- 注：悪意を持った不正行為を行う者をクラッカーという．ハッカーは，コンピュータ技術に長け，その技術を生産（善意）的なことに利用する者を指す．但し，日本のマスコミは，クラッカーのことを「ハッカー」と言っている．

暗号（cryptography）

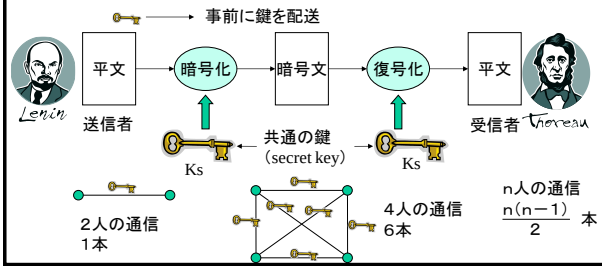
- 当事者以外にわからないﾃﾞｰﾀに変換．暗号化により以下が実現
  - 秘匿通信：盗聴されても（ﾃﾞｰﾀが盗まれる）意味がわからない
  - 認証：本人であること，ﾃﾞｰﾀが本人のものであることを確認
  - 完全性の保障：ﾃﾞｰﾀが改ざんされていないことを受信者が確認

平文（ひらぶん）（元のﾒｯｾｰｼﾞ）を暗号化鍵をﾊﾟﾗﾒｰﾀとする関数により，暗号文に変換
上記の変換を暗号化，変換手順を暗号化アルゴリズムという．
受信者は，復号化鍵を使って，暗号文を平文に戻す（復号化）．



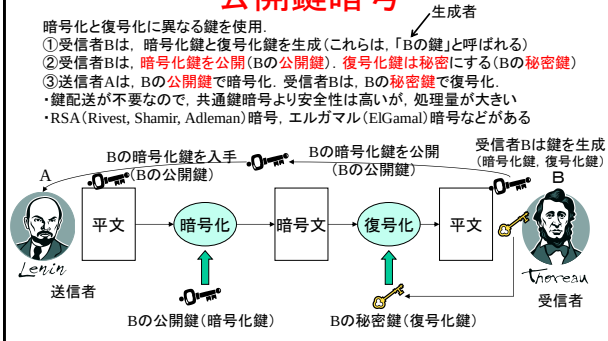
共通鍵暗号

共通鍵暗号：暗号化と復号化に同じ鍵（Ks）を使用
・処理が高速なので，大量データの暗号化に適している
・鍵を秘密にして配送する必要がある．
・通信相手毎に異なる鍵を使用する必要がある（通信相手が多いと，鍵の管理が大変）
・DES（Data Encryption Standard），IDEA，RC5などの暗号がある



公開鍵暗号

暗号化と復号化に異なる鍵を使用．
①受信者Bは，暗号化鍵と復号化鍵を生成（これらは，「Bの鍵」と呼ばれる）
②受信者Bは，暗号化鍵を公開（Bの公開鍵）．復号化鍵は秘密にする（Bの秘密鍵）
③送信者Aは，Bの公開鍵で暗号化．受信者Bは，Bの秘密鍵で復号化．
・鍵配送が不要なので，共通鍵暗号より安全性は高いが，処理量が大きい
・RSA（Rivest, Shamir, Adleman）暗号，エルガマル（ElGamal）暗号などがある



・秘密鍵を持つ人（受信者B）だけが，暗号文を平文に戻せる
・公開鍵（暗号化鍵）では，復号化できない→通信相手が何人いても，鍵は1つでよい．

参考　RSA暗号

$p, q \in P$（素数の集合）　$n=pq$　$\varphi(n)=(p-1)(q-1)$と素な整数eを選び，$ed=1 \bmod \varphi(n)$を満たすdを求める（$\varphi(n)$はnと互いに素なn以下の自然数の個数）．
(n, e)を公開鍵，(d, p, q)を秘密鍵とする．
［巨大な合成数の素因数分解（nからp,qを見つける）が難しいことを利用］

暗号化
送りたい平文のメッセージを$m(0<m<n)$とする．
$c=m^e \bmod n$　を計算して，暗号文cを送る．

復号化
$c^d \bmod n$を計算

オイラーの定理より$m^{\varphi(n)}=1 \ (\bmod n)$
$ed=k\varphi(n)+1$と書けるので（kは整数）
$c^d=(m^e)^d = m^{ed}=m^{k\varphi(n)+1}$
$=m(m^{\varphi(n)})^k=m(1)^k=m \ (\bmod n)$

例
p=1231，q=4567，n=1231×4567＝5621977
$\varphi(n)$＝（1231-1）×（4567-1）＝5616180
e=65537, d=3988493
（$ed/\varphi(n)$=261393865741/5616180=46543･･･1）
平文のﾒｯｾｰｼﾞ　m=10000
$c=10000^{65537} \bmod 5621977=4030596$
$c^d=4030596^{3988493} \bmod 5621977= 10000=m$

但し，このままだとm=10000の暗号文は常に
C=4030596⇒総当たり計算で解読されてしまう

RSA PKCS #1 v1.5 （RFC 2313）
メッセージmに乱数列PSと区切り記号00を付加し，
そのビット列を暗号化

00	BT	PS(乱数)	00	m

復号化後に削除

参考　ElGamal暗号

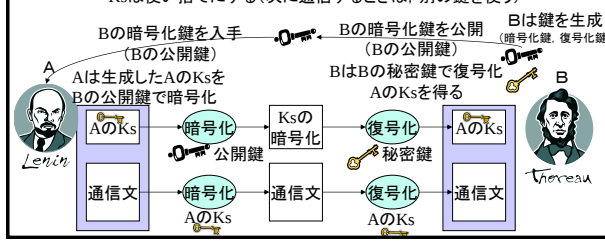
$p \in P$（素数の集合）　　$g \in Z^*_p$（乗法群をなす整数の剰余類の集合）
自然数a $(1 \leqq a \leqq p-2)$　$A=g^a \bmod p$
(p, g, A)を公開鍵，aを秘密鍵とする．
（g, Aからaを計算するのは簡単だが，g, Aからaを見つけるのは難しいことを利用）

暗号化
送りたい平文のメッセージを$m(0 \leqq m<p)$とする．
自然数の乱数r $(1 \leqq r \leqq p-2)$をランダムに選び
$B=g^r \bmod p$　　$c=A^r m \bmod p$
を計算して，暗号文(B, c)を送る．
（平文mに乱数rを付与することで，同じmの暗号文が毎回異なるようにする）

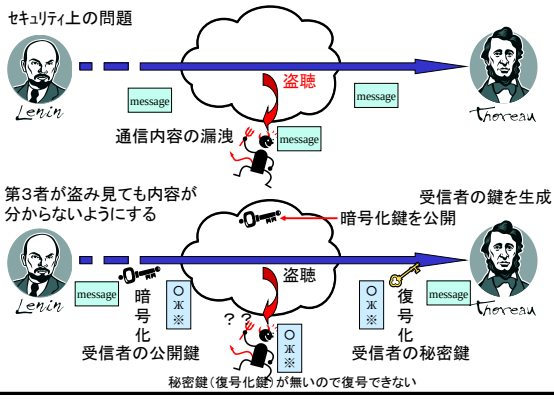
復号化
$B^{-a}c \bmod p$を計算
$B^{-a}c=g^{-ra}g^{ar}m=g^{-ra+ar}m=g^0m=m$　$(\bmod p)$

組み合わせ方式

- 公開鍵暗号を使って，共通鍵暗号の鍵（Ks）を送る
- ①Bは暗号化鍵，復号化鍵を生成し，暗号化鍵を公開（Bの公開鍵）
- ②AはKs（AのKs）を生成し，Bの公開鍵で暗号化してBに配送
- ③Bは暗号化されたAのKsをBの秘密鍵で復号化し，平文のKsを得る
- ④以後の通信は，AのKsを用いた共通鍵暗号で行う
  - Ksは使い捨てにする（次に通信するときは，別の鍵を使う）



公開鍵暗号（秘匿通信）



鍵の種類と暗号化の方式

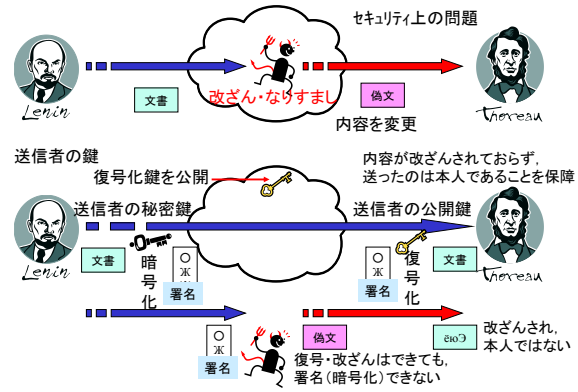
- 暗号化鍵：平文を暗号文に変換する（暗号化）ときに使用
- 復号化鍵：暗号文を平文に戻す（復号化）ときに使用

- 共通鍵暗号：暗号化鍵＝復号化鍵である暗号化方式
  - この共通鍵は，絶対に秘密にしておく必要がある
  - 正式名称は，「秘密鍵」（Secret key）という
    - Secret：秘密の，隠れた
- 公開鍵暗号：暗号化鍵≠復号化鍵である暗号化方式
  - 公開鍵（Public key）：公開する方の鍵
  - 秘密鍵（Private key）：公開しない（秘密にしておく）方の鍵
    - Private：私的な，非公開の，内密の
  - 暗号：受信者の暗号化鍵を公開鍵，復号化鍵を秘密鍵にする
    - 盗み見ても秘密鍵を持たないと中身が分からない．
  - ﾃﾞｨｼﾞﾀﾙ署名：送信者の暗号化鍵を秘密鍵，復号化鍵を公開鍵にする
    - 秘密鍵を持った者だけが署名を作成できる．

利用者IDと認証

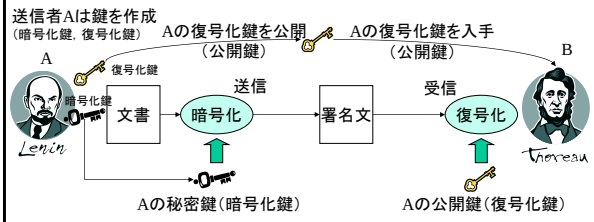
- 利用者ID（アカウント）：登録された利用者に付けられた識別子
  - ｼｽﾃﾑを利用する際に，OSに利用者IDを提示する
- 認証：利用者が，利用者IDで表される本人かどうかを確認すること
- ﾊﾟｽﾜｰﾄﾞ：合言葉．認証の方式として最も一般的なもの．
- パスワードを守るためのOSの仕組み
  - 裸のﾊﾟｽﾜｰﾄﾞを持たない．Unixのﾊﾟｽﾜｰﾄﾞﾌｧｲﾙは，暗号化情報．
    - 入力ﾊﾟｽﾜｰﾄﾞ→暗号化関数で変換→ﾊﾟｽﾜｰﾄﾞﾌｧｲﾙと照合
    - ﾊﾟｽﾜｰﾄﾞﾌｧｲﾙが盗まれてもﾊﾟｽﾜｰﾄﾞは漏れない．
  - 誤ったﾊﾟｽﾜｰﾄﾞを何回か投入したら認証失敗として打ち切る
    - 記録をとる（不正侵入攻撃かも知れない）
- 辞書に載っている単語をﾊﾟｽﾜｰﾄﾞにすると危険
  - 全単語を暗号化関数で変換してﾊﾟｽﾜｰﾄﾞﾌｧｲﾙと比較
    - たかが数万の単語ならあっという間に総当りできる

ディジタル署名の目的



ディジタル署名（基本方式）

ﾃﾞｨｼﾞﾀﾙ署名：メッセージの作成者が本人であることを確認する
（受け取ったメッセージは送信者本人が作成したものである）
①ﾒｯｾｰｼﾞが改ざんされていない，②署名したのは本人である
③「自分は署名しなかった」と主張することができない否認防止にもなる

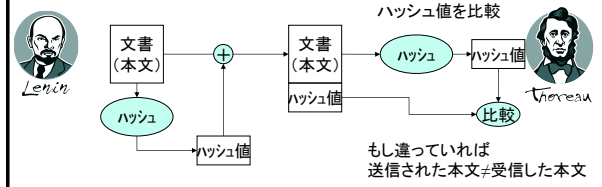


ﾒｯｾｰｼﾞを送信者の秘密鍵で暗号化（署名）．署名文を送信者の公開鍵で復号化．
秘密鍵を持つ人（送信者A）だけが署名文を作成できる
但し，メッセージ全体に署名するため，処理時間がかかる（計算量が多い）

ハッシュ値による照合

本文と本文のハッシュ値を送信

本文のハッシュ値と受け取った
ハッシュ値を比較

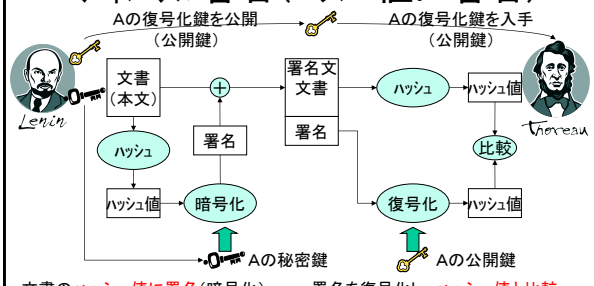


もし違っていれば
送信された本文≠受信した本文

ハッシュ：任意長のﾒｯｾｰｼﾞを固定長の短い数値に変換する関数
・MD5：128ビットのハッシュ値を生成（RFC1321）
・SHA1：160ビットのハッシュ値を生成（米国政府の標準ハッシュとして採用）

ハッシュ値の計算は，処理量が少ない．
ただし，ハッシュ値は，だれにでも計算できる．
このままでは，なりすましや改ざんを防ぐことはできない．
→ハッシュ値を用いた比較とディジタル署名の基本方式を組み合わせる．

ﾃﾞｨｼﾞﾀﾙ署名（ハッシュ値に署名）

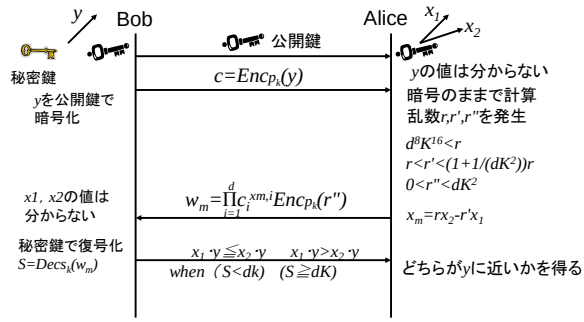


文書のハッシュ値に署名（暗号化）　　署名を復号化し，ハッシュ値と比較

なりすましの防止
ハッシュ値を正しく暗号化できるのは秘密鍵を持った本人のみ．
改ざんの有無を検出
ハッシュ値が一致するのは，改ざんされていない場合のみ．

参考：より高度な暗号の応用例

PPDM（Privacy Preserving Data Mining：プライバシー保護データマイニング）
Aliceのx_iとx_2のどちらが，Bobのyに近いか？　x_1, x_2, yを互いに秘密にしたままで判定



加法的準同型性という性質を持った暗号を用いる

参考　加法的準同型性暗号

s_k：秘密鍵（復号化鍵）
p_k：公開鍵（暗号化鍵）
Enc_{pk}：暗号化
Dec_{sk}：復号化
$m_1, m_2 \in Z^*_n$：平文

暗号文の積を復号化すると平文の和
$Dec_{sk}(Enc_{pk}(m_1) \cdot Enc_{pk}(m_2)) = m_1 + m_2$

暗号文を累乗して復号化すると平文の積
$Dec_{sk}(Enc_{pk}(m_1)^{m2}) = m_1 m_2$

⇒暗号化したままで，元のデータ（ベクトル）
の類似度が計算できる

例：Pailler暗号系は加法的準同型性を持つ

$p_k = n = pq$　　$p,q \in P$
$s_k = \lambda = lcm(p-1, q-1)$
$c = Enc_n(m) = g^m r^n \bmod n^2$　　$g \in Z^*_{n^2}$
$m = Dec_\lambda(c) = L(c^\lambda \bmod n^2) L(g^\lambda \bmod n^2)^{-1}$
　但し，$L(u) = (u-1)/n$　$(\bmod n)$