4/23暗号技術

2025年4月22日　11:20

連絡：
- 7/16は、アクセンチュア様による演習授業予定。

暗号とは
- 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加工してあるもの。
- 上記の加工方法のことを「暗号」と呼ぶこともある。

わたしたちと暗号
- インターネット通信で、通信内容を他者に見られないように使う。
  - ○ 暗号化通信に対応しているものの例：
    - HTTPS：
      - 自分の端末から接続先webサーバまたはプロキシサーバまでの間の通信を暗号化する。
    - 無線LAN（で暗号化設定してあるもの）：
      - 自分の端末から無線アクセスポイントまでの間の通信を暗号化する。
    - IMAP over SSL：
      - メールを受信する際に、メールサーバから自分の端末までの間の通信を暗号化する。
  - ○ すべてのプロトコルが暗号化通信しているわけではない。以下のものは暗号化しない。
    - HTTP
    - SMTP
    - 有線LAN(ethernet)

- 技術者・開発者寄りの話：
  - ○ 暗号の仕組みを知るには、古典暗号を知り、現代暗号を知るのが定番ではある。ただし、古典暗号は実用的でない。
  - ○ 暗号の仕組みを理解するには、自分で実装すると理解が深まる。ただし、実システムで暗号技術を使うときは、オープンソースなどの既存の定番ライブラリを使う方が総合的には良い。
    - ∵自分よりも優れた人たちが実装し、既に使われて検証されているので。

2.1.1 基礎用語
- 平文（ひらぶん）　…　元の入力文。
- 暗号文　　…　平文を暗号化し、他人には読めなくした文。

- 暗号化　…　入力された平文を理解困難な暗号文に書き換えること。
- 復号　　…　暗号化の逆。暗号文を平文に戻すこと。

- 鍵　　…　暗号化や復号を行う際に使うパラメータのこと。
  - ○ 暗号化鍵　…　暗号化で使う鍵。
  - ○ 復号鍵　　…　復号の際に使う鍵。

- 共通鍵暗号化方式　…　暗号化と復号で同じ鍵を使う暗号化方式。
- 公開鍵暗号化方式　…　暗号化の鍵と復号の鍵の形が異なる暗号化方式。

- ケルクホフスの原理　…　復号鍵以外のすべてが公知になっていたとしても、なお安全であるべきだ、という考え方。現代暗号の原則。

- 古典暗号　…　ケルクホフス原理を満たさないような暗号。
- 現代暗号　…　ケルクホフス原理を満たすような暗号。

基本的な数学的知識：
- 排他的論理和(XOR)
- 剰余算（モジュロ演算）
- 乱数、疑似乱数

排他的論理和(XOR)の復習(p.62)
- 2値のA、Bの値が一致するときに0、一致しないときに1になるもの。

A	B	A⊕B

0	0	0
0	1	1
1	0	1
1	1	0

- 表記法：
  - Y = A ⊕ B
  - Y = A XOR B

- XORの性質：
  - ある値Aに対して、任意の値Bで2回XORを取ると、元のAに戻る。
  - Y = A ⊕ B ⊕ B すると、Y = A になる。
    - ∵ B ⊕ B を先に見ると、ここは必ず0になる。そして一方の値が0のとき、XORは他方の値をそのまま出力するので、Y=Aになる。

- 平文を2進数の数列で捉え、各桁にXORをとることで、元の平文とは一見異なる数列を得ることができる。さらにもう一度XORを取れば元の平文に戻せる。

剰余算（モジュロ演算、p.84）
- ある整数Aを整数Bで割ったときの余りを求める計算のこと。
  - 小学3年の割り算の感覚。

- 表記法（A=割られる数、B=割る数、C=剰余）：
  - A mod B = C
  - A ≡ B (mod C)　（※「AとBは、Cを法にして合同」と呼ぶ）
  - C言語などでの演算記号は% (例： C = A % B;)

- 端数を求めたいときに有効。
- ある値AをB未満の値で循環させたいときに有効。

乱数、疑似乱数（p.64）
- 乱数の種類：
  - 真の乱数（真性乱数）：
    - 文字通りの乱数であり、次にどのような値が得られるか予想できない。
    - コンピュータで真性乱数を利用する場合、物理現象を乱数計算に取り込む。ただし比較的低速。
  - 疑似乱数：
    - 規則的に生成された、乱数のように見える数値列。
    - コンピュータで生成する乱数の大半は疑似乱数。
    - 規則があって再現できる。
      - 例えばC言語ではsrandom(X);を使って乱数の出方を初期化するが、Xに以前と同じ値を使うことで同じ乱数列を再生成できる。
    - 出力される値に周期性がある。
      - 例：1, 4, 2, 6, 6, 3, 4, 2, 6, 6, 3, 4, 2, 6, 6, 3, 4, 2, 6, 6, 3, 4・・・
        - 冒頭数個だけ見ると乱数っぽいが、途中から{2, 6, 6, 3, 4}の数列が周期的に繰り返されている。
      - 例えば、C言語のrand()には周期性が短いという問題が明らかになっており、使わないことが推奨されている。代わりにrandom()を使う。
        - https://www.jpcert.or.jp/sc-rules/c-msc30-c.html
        - $ man 3 rand
    - よい疑似乱数とは・・・
      - 無作為性（randomness）
        - 一見ばらばらに見えるか。
        - 値の出方に偏りがないか（一様乱数の場合）。
      - 予測困難性（unpredictability）
        - 過去の出力値から次の出力値の予測ができるか。
        - 出力値の周期性が十分長いかどうか。
      - 再現困難性（irreproducibility）
        - 再現が困難かどうか
        - 暗号化ではよく疑似乱数を使うが、再現できる乱数を使っているということは、オフライン攻撃の可能性を残すといってよい。

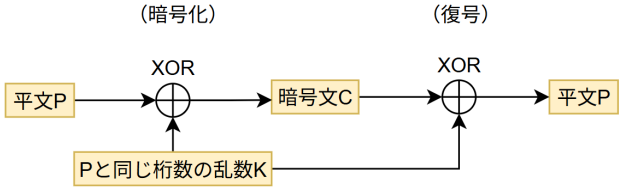
- 閑話休題：
  - セキュリティの観点では真性乱数がベストだが、それ以外の観点、例えばシミュレーションや開発デバッグ時には真性乱数より再現性のある疑似乱数の方が好都合であることも多い。
  - 乱数いろいろ：

- 一様乱数
  - 各出力値の出現確率がすべて均等であるような乱数のこと。
- 正規乱数
  - 各出力値の出現確率が正規分布の形に分布するような乱数のこと。すなわち平均値近くの値がよく出現し、平均値から離れた値ほど出現しにくい。
- パーリンノイズ
  - CGでの景色のシェーディングなどで利用されるもの。
  - 例えば海面の高さを考えた時、ランダム性が含まれるようである。しかし、一様乱数のような分布かというとそうではなく、規則も含まれるようである。
    - 例えばある地点Aでの海面の高さをHaとおいたとき、Aから10cm離れた地点での海面の高さはHaに近いはずである。
    - 一方、Aから100m離れた地点での海面の高さはHaと無関係なランダムな値になっていると考えられる。
  - パーリンノイズは、上記のように近い場所同士の値は近く、離れた場所同士の値はランダムになるような乱数を出力する。

2.1.2 代表的な古典暗号

- シーザー暗号
  - ○ 暗号化：
    - 鍵を1以上の整数Kとし、平文の各文字Cをアルファベット順にK文字ずつずらすことで暗号化する。
      - 例（K=1）：平文=ABC　→　暗号文=BCD
      - 例（K=1）：平文=HELLO　→　暗号文=IFMMP
  - ○ 復号：
    - 各文字を-K文字ずつずらす。
  - ○ 「換字暗号」の一種。
    - なお、換字暗号の他に「転置暗号」（平文の文字列を並び替える方法）がある。
  - ○ ブルートフォース攻撃（総当たり攻撃）ですぐ解読されてしまう。
    - K=1からK=25まで試し、それぞれを目視で読めば答えは何か分かる。

- バーナム暗号

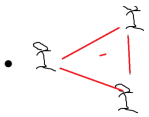


  - ○ 暗号化：
    i. まず平文Pを2進数で表し、その桁数が何bitかを知る。ここで桁数をN bitとおく。
    ii. N bitの乱数Kを生成する。このKが鍵になる。
    iii. P ⊕ K = Cで暗号文Cを得る。
  - ○ 復号：
    - C ⊕ K = P で平文Pを得る。
  - ○ 現実的には実現困難だが、理論上は強力な暗号化手法。
  - ○ 問題点3つ(p.65)：
    - 暗号化鍵の安全な配送が困難。
    - 暗号化鍵を再利用しないので、使い勝手が良くない。
    - 疑似乱数を使う限り、完璧とは言い切れない。
  - ○ 素朴なバーナム暗号化のソースコード例：

```
#include <stdio.h>
#include <string.h>
#define N 256

char plaintext[N];   // 平文
char ciphertext[N];  // 暗号文
char key[N];         // バーナム暗号用の鍵
int i;

int main(){
  //---- 入力
  fprintf(stderr, "平文を入力して下さい： ¥t");
  scanf("%s", plaintext);
```

```
    fprintf(stderr, "鍵を入力して下さい： ¥t");
    scanf("%s", key);

    //---- バーナム暗号化
    for (i = 0; i < strlen(plaintext); i++){
        ciphertext[i] = plaintext[i] ^ key[i];
    }
    ciphertext[i] = '¥0';

    //---- 結果出力
    printf("平文(16進数表現)： ¥t");
    for (i = 0; i < strlen(plaintext); i++) printf("%02X ", plaintext[i]);
    printf("¥n");

    printf("鍵(16進数表現)： ¥t");
    for (i = 0; i < strlen(key); i++) printf("%02X ", key[i]);
    printf("¥n");

    printf("暗号文(16進数表現)： ¥t");
    for (i = 0; i < strlen(plaintext); i++) printf("%02X ", ciphertext[i]);
    printf("¥n");

    printf("暗号文(文字列表現)： ¥t%s¥n", ciphertext);

    return 0;
}
```

2.1.3　共通鍵暗号化方式と公開鍵暗号化方式

- 共通鍵暗号
  - ○　暗号化に使う鍵　＝　復号に使う鍵　となる方式
  - ○　一般的に公開鍵暗号に比べて高速
  - ○　暗号化通信する際に、送受信者間で安全に鍵を共有する方法が別途必要になる。
  - ○　スケーラビリティにはすぐれない。
    - n名の集団の中で任意の2名が秘密の通信を行う場合、全部で${}_nC_2$個の鍵が必要であり、各ユーザはn-1個の秘密を保持する必要がある。
      - n=3の場合、下図の赤線のように3個の鍵が必要になる。
    - 

- 公開鍵暗号
  - ○　暗号化に使う鍵　≠　復号に使う鍵　となる方式
  - ○　一般的に共通鍵暗号に比べて低速
  - ○　暗号化通信する際に、送受信者間で安全に鍵をやり取りできる。

- ハイブリッド暗号
  - ○　共通鍵暗号と公開鍵暗号を組み合わせ、良い所どりをする。
  - ○　公開鍵暗号での通信を使って、共通鍵暗号の鍵を送受信者同士で共有する。そしてそれ以降は共通鍵暗号で高速に暗号化通信を行う。

2.2　共通鍵暗号化技術

暗号化の種類：

- ストリーム暗号化
  - ○　平文を1bit単位で暗号化する仕組み。
  - ○　平文と同じ長さの鍵を生成し、平文と鍵とでXORを取る流れ。
- ブロック暗号化
  - ○　平文をブロック単位（例えば64bitごと）で暗号化する仕組み。
  - ○　鍵生成部とデータ撹拌部に分かれる。

DES

- 以前はよく使われた（代表的な）ブロック暗号化技術の一つ。
- ブロックの単位は64bit。
- 鍵の長さは64bit。ただし、一部パリティビットがあるので、実質56bit。
  - ○　総当たり攻撃の考え方だと、2^56パターンの攻撃で確実に解読される。

- Feistel構造を採用
  - ○ 処理対象のデータを上位桁と下位桁とに分離し、XORや関数にかけて数値を変える処理の事。
- 1999年の時点で数十時間で解読される状況になっており、現在は使われない手法になっている。

3DES
- DESを3回（1:ある鍵Aで暗号化、2:別の鍵Bで復号、3:別の鍵Cで暗号化）使う方式。
- 鍵を3種類（あるいは2種類）に増やすことで、解読に必要な鍵の長さを伸ばす考え方。
- 今のところ使える手法（例：NISTは2030年まで利用を承認）。

AES
- SPN構造を採用。Feistel構造よりも高速に処理できる。
- 最近の主流のブロック暗号化の一つ。
  - ○ 電子政府推奨暗号(CRYPTREC)に入っている。
- ブロック単位は128bit。
- 鍵の長さは128、192、256bitの3種類。
- 家庭用無線LANの暗号化でよく使われている。

RC4
- ストリーム暗号の一種。
- 以前は、SSL/TLS(https通信)や無線LANのWEPで使われていた。
- 現時点では使うべきでない手法。

ChaCha20
- 比較的新しく提案されたストリーム暗号の一種。
- AEADのアルゴリズムの一つ。すなわち、暗号化（機密性の担保）に加えて完全性も担保する仕組みがある。

4/30↓

2.2.3　DESの仕組み

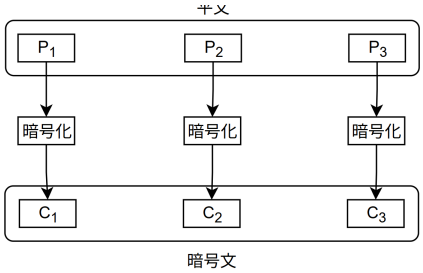
- 入力された平文をブロックに分け、各ブロックについて、所定の暗号化を繰り返し行う（撹拌）。
- 3DESで暗号化・復号するLinuxコマンドの例：
  - ○ $ echo 平文 | openssl des3 -e -a -pass pass:パスワード
    - -des3　…　暗号化アルゴリズムを3DESにする
    - -e　…　暗号化する
    - -a　…　暗号文を可読文字で表現する（BASE64エンコード）
    - -pass　…　パスワードを指定する
    - ※実際には、-pbkdf2オプションも使って鍵を複雑にした方が良い。
  - ○ $ echo 暗号文 | openssl des3 -d -a -pass pass:パスワード
    - -d　…　復号する

2.2.4　AESの仕組み

- AES-256-CBCで暗号化・復号するLinuxコマンドの例：
  - ○ $ echo 平文 | openssl aes-256-cbc -e -a -pass pass:test
    - -e　…　暗号化する
    - -a　…　暗号文を可読文字で表現する（BASE64エンコード）
    - -pass　…　パスワードを指定する
    - ※実際には、-pbkdf2オプションも使って鍵を複雑にした方が良い。
  - ○ $ echo 暗号文 | openssl aes-256-cbc -d -a -pass pass:test
    - -d　…　復号する

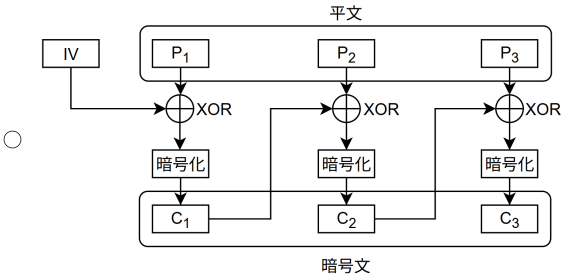
2.2.5　パディング
- ブロック暗号化で、平文をブロック単位の倍数にそろえるために使うもの。
- 代表的な手法は表2.7にある。
- PKCS#5
  - ○ 平文…EM
  - ○ PS …パディングする文字列。
    - PSをどうするか（何をどれだけ詰めるのか）の例はp.84の数式参照。
    - EMの長さが丁度ブロック長の倍数だったときもパディングも行う（p.84の一番下の0808…の式の例）。
  - ○ 教科書の説明でのブロック単位は8byte、すなわち64bit。

2.2.6 ブロックチェイニング

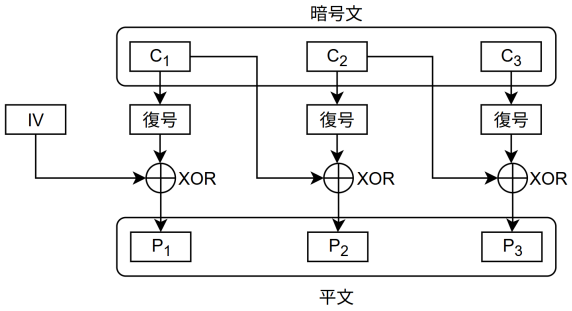


- ○ ECBモードの図。
- ○ 特徴(欠点)： Ci = Cjであれば Pi=Pj
- ○ このモードは通常使うべきでない。
  - 攻撃者が長期的に暗号文を盗聴したとき、一部の暗号文ブロックで値が一致する組が観測されることがある。このとき、両ブロックの元の平文の内容も一致していることが攻撃者にばれてしまう。

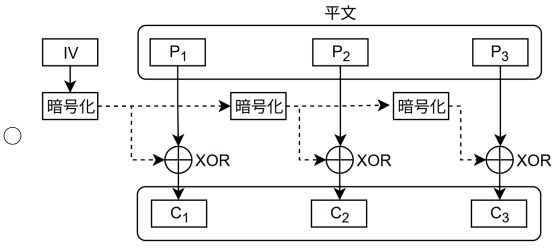
- CBCモード



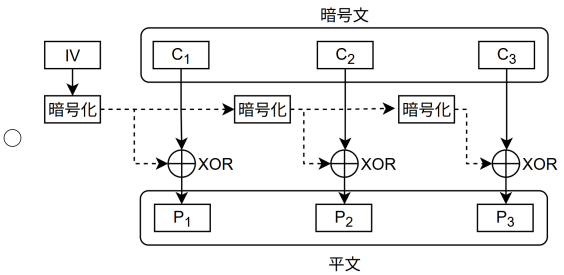
  - IV… Initial Vector（初期化ベクトル)。ブロック長にあわせてランダムに生成した数値。
  - たとえPi=Pjの平文ブロックがあったとしてもその暗号文はCi≠Cjになるので、長期的に盗聴されても安全といえる。



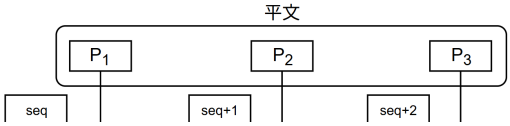
- OFBモード

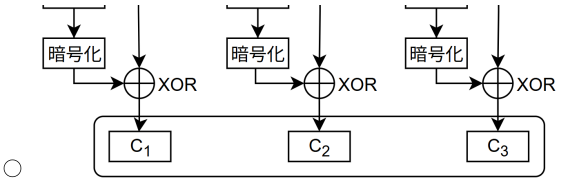


  - IVを繰り返し暗号化して任意の長さの文字列を得られるので、これをバーナム暗号の鍵として使っている。
  - ストリーム暗号としても使える。



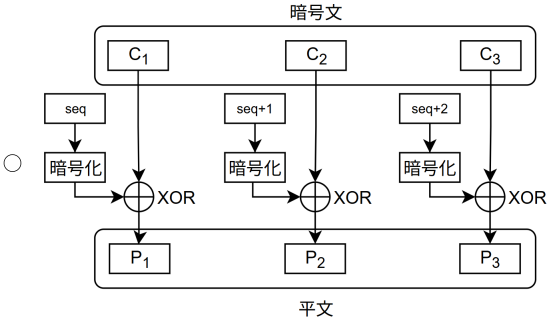
- CTRモード





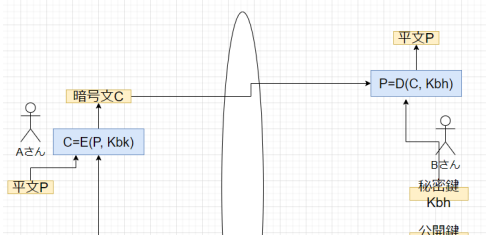
暗号文

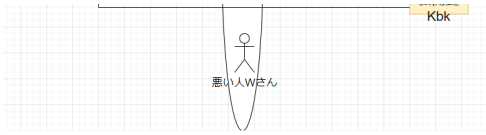
- バーナム暗号の鍵として、カウンタの値を暗号化して得られた数値を使うもの。
- 並列処理が可能なので高速。



- 2.3 公開鍵暗号化技術
  - RSAのしくみ。
    - 暗号化に使う鍵 ≠ 復号に使う鍵 となる方式
  - RSAの鍵生成手順：
    i. 素数p, qを決める（なるべく大きな方が良い）
    ii. n=pq, z=(p-1)(q-1) を求める。
    iii. zと互いに素な数eを見つける。
      a. 互いに素 … zとeの最大公約数が1になるような数のこと
    iv. ed mod z = 1になるようなdを見つける。
    v. ここまで求めたパラメータを次のように用いる。
      a. 公開鍵（暗号化鍵）：(n, e)
      b. 秘密鍵（復号鍵） ：(n, d)
        ※ ただしnは隠しても意味がないもの。
      c. 暗号化関数：暗号文C = E(平文P, 暗号鍵) = P^e mod n
      d. 復号関数 ：平文P = D(暗号文C, 復号鍵) = C^d mod n
  - 演算の例：
    i. 鍵生成：
      a. p=5、q=11
      b. n=55、z=40
      c. zと互いに素になる数として、e=3を求める。
      d. 3d mod 40 = 1になるような数として、d=27を求める。 (81 mod 40 = 1)
      e. 公開鍵（暗号化鍵）：(55, 3)
      f. 秘密鍵（復号鍵） ：(55, 27)
    ii. 暗号化：
      a. 平文P=7とすると、 P^e mod n = 7^3 mod 55 = 13
    iii. 復号：
      a. C^d mod n = 13^27 mod 55 = 7

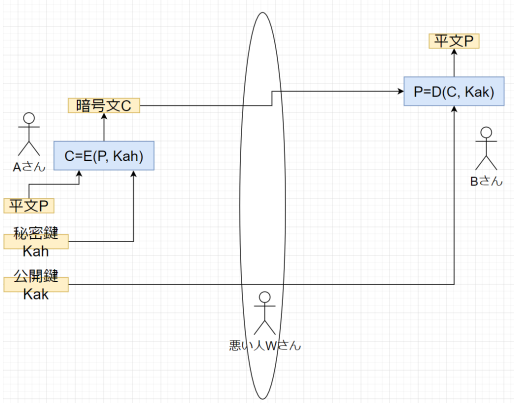
- 公開鍵暗号を使った素朴な暗号化通信の手順：
  - パラメータの定義：
    - 平文＝P、暗号文＝C
    - 送信者＝Aさん、受信者＝Bさん
    - Bさんの秘密鍵＝Kbh、Bさんの公開鍵=Kbk
    - 暗号化関数：E(P, カギ)
    - 復号関数 ：D(C, カギ)





- ○ ※ 悪い人WはKbkとCを盗聴できるが、Kbhを知らないので復号できない。

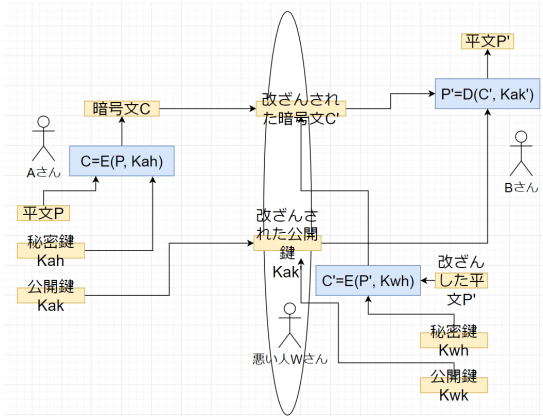
- RSAの利用：
  - ○ 暗号化以外にディジタル署名（電子署名）にも使える。



  - ○ ↑仮にWさんがCを改ざんしたとしても、D(C', Kak)で適切に復号できないので、改ざんに気づける。

- （4/30ここまで。前回までの宿題の〆切は**5/14**）

-
  - ○ ↓一方、仮にWさんがCおよびKakを改ざんした場合、Bさんは改ざんに気づけない。



  - ○ ↑この問題を解消するには、公開鍵とその持ち主の対応関係が保証されなければならない。それを実現させる仕組みが「公開鍵基盤（PKI：Public Key Infrastructure）」である。

- RSA以外の公開鍵暗号化方式：
  - ○ エルガマル暗号
  - ○ ECC（楕円曲線暗号）
    - RSAと比べ、短いbit数の鍵で同等以上の暗号強度を持つ。（例：RSAの2048bitとECCの224bitが同等の強度。）
  - ○ ディジタル署名用のアルゴリズム；
    - DSA (DSS)
      - エルガマル暗号を改良して署名専用にしたもの
    - ECDSA
      - ECCを署名専用にしたもの

- 2.4　鍵共有アルゴリズム

- 2.4.1 Diffie-Hellman鍵共有アルゴリズム
  - ○ ユーザAとBとの間で共通鍵Kを安全に作る手順：
    i. 2人で素数pとgを決める。　（通信発生）
    ii. Aが秘密鍵x（乱数）を決める。

iii. Bが秘密鍵y（乱数）を決める。

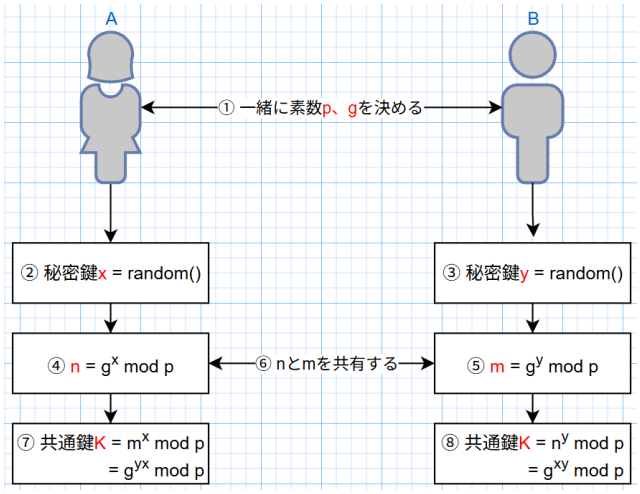
iv. Aが次式でnを求める　→　$n=g^x \bmod p$

v. Bが次式でmを求める　→　$m=g^y \bmod p$

vi. nとmを互いに交換する。　（通信発生）

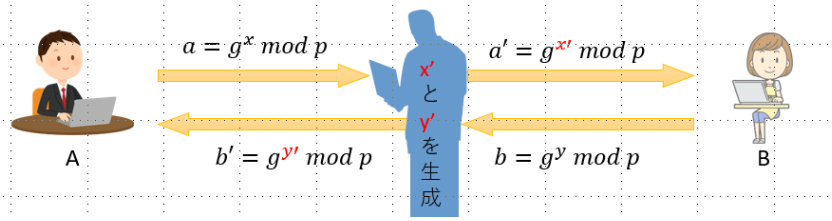
vii. Aが次式でKを求める。　→　$K=m^x \bmod p = g^{yx} \bmod p$

viii. Bが次式でKを求める。　→　$K=n^y \bmod p = g^{xy} \bmod p$



- 通信を盗聴可能な攻撃者はp, g, n, mを盗聴できる。しかしこれらからxとyを知ることは難しい。よってKを知ることが困難である。
- 以上のことから、通信経路を盗聴されている環境下においても、Kを共通鍵暗号の鍵として使える。

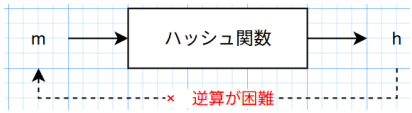
- 素朴なDH鍵共有アルゴリズムは中間者攻撃（MITM攻撃）に弱い。
  - MITM...通信途中に割り込んで、盗聴や改ざんを行う攻撃。

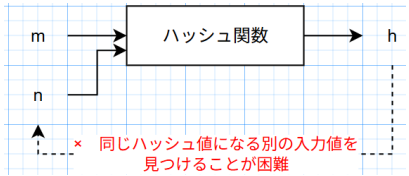


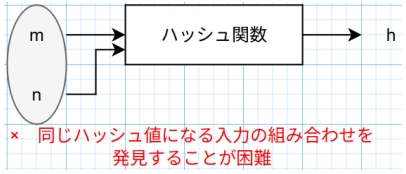
- 中間者が盗聴と改ざんが可能なとき、x'とy'を生成してA、Bそれぞれと鍵交換する。中間者は以降の通信を盗聴・改ざんできるようになり、AとBはそれを見抜けない。
- そのため、その対応策としてデジタル署名を組み合わせてDH鍵共有を行うことになる。

2.5 ハッシュ関数とディジタル署名

2.5.1 ハッシュ関数

- セキュアハッシュ関数に求められる条件：
  - 一方向性
    - 

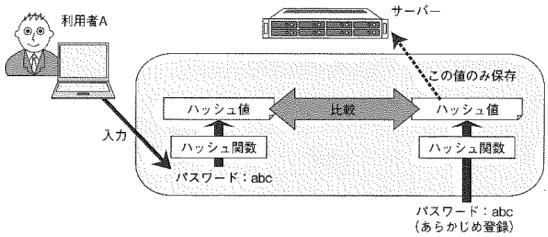
  - 第２原像計算困難性
    - 

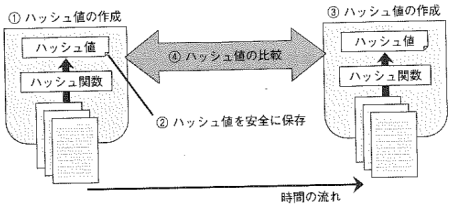
  - 衝突困難性
    - 

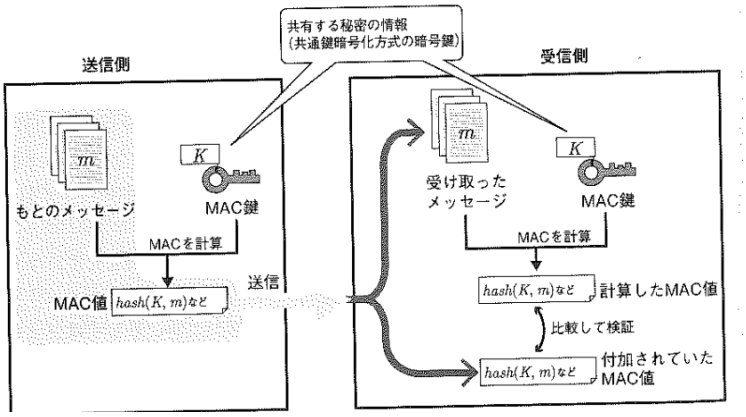
- ハッシュ関数は、あるデータの完全性（改ざんされていないか）を確認するときに使われる。

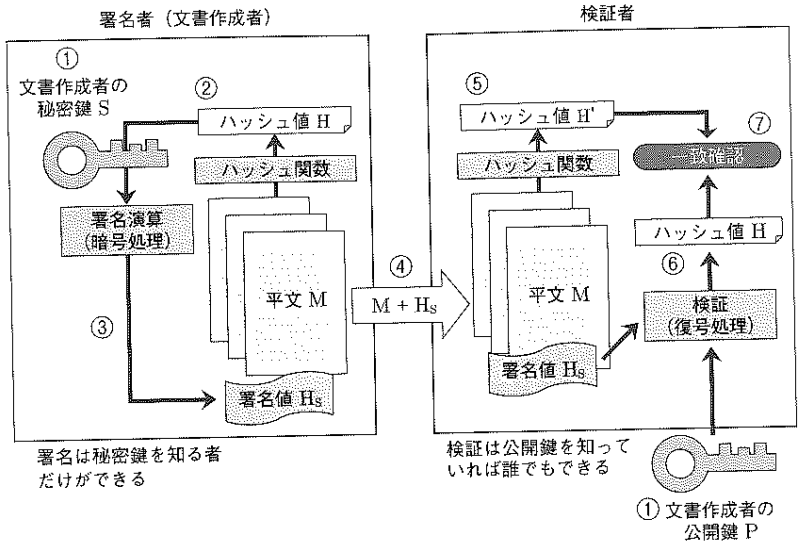
- ○ ハッシュ関数は、あるデータの完全性を検証できるような固定長のID文字列を出力する関数ともいえる。
- ○
  （参考：さくらインターネットの無償シェルサービス）
  https://www.sakura.ad.jp/services/cloudshell/
  - ・ ユーザ登録なしにUNIXのシェル環境で演習できるサービス。
  - ・ ハッシュ関数のコマンドとして、sha256sumやmd5sumなどがインストール済み。

- ○ ハッシュ関数の具体例：
  - ・ MD4、MD5、SHA、SHA-1 … 以前使われていたもの。
  - ・ SHA-2、SHA-256 … 最近使われているもの。

- ○ パスワード認証におけるハッシュ関数の利用
  - 

- ○ ハッシュ関数によるファイル改ざん検知
  - 

- ○ メッセージ改ざん通知
  - 

- 2.5.2. ディジタル署名
  - ○ 改ざん検知、否認防止のために使う。
  - ○
    

- 2.5.3 AEAD
  - ○ Authenticated Encryption with Associated Data
  - ○ 暗号化に加え、真正性と完全性を確保する暗号化方式のこと。

2.6 暗号技術における安全性

- 計算量的安全性　…　解読に必要なアルゴリズムの計算量に着目した概念。
- 等価安全性　…　表2.13。
  - ○ 例えば表の256ビット安全性の欄に着目すると、共通鍵ならば256bitでその強度を実現できるのに対し、RSAならば15360bit必要になる。一般的に鍵長は短い方が好ましいので、共通鍵暗号の方がより効率良く強度を確保できていると言える。

(5/14ここまで↑)