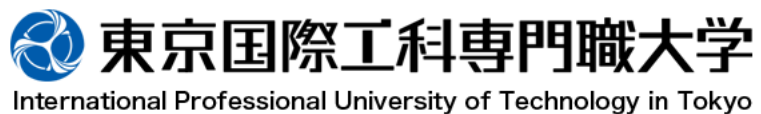




# 情報セキュリティ応用 第6回 暗号の基礎（Ⅰ）

爰川 知宏



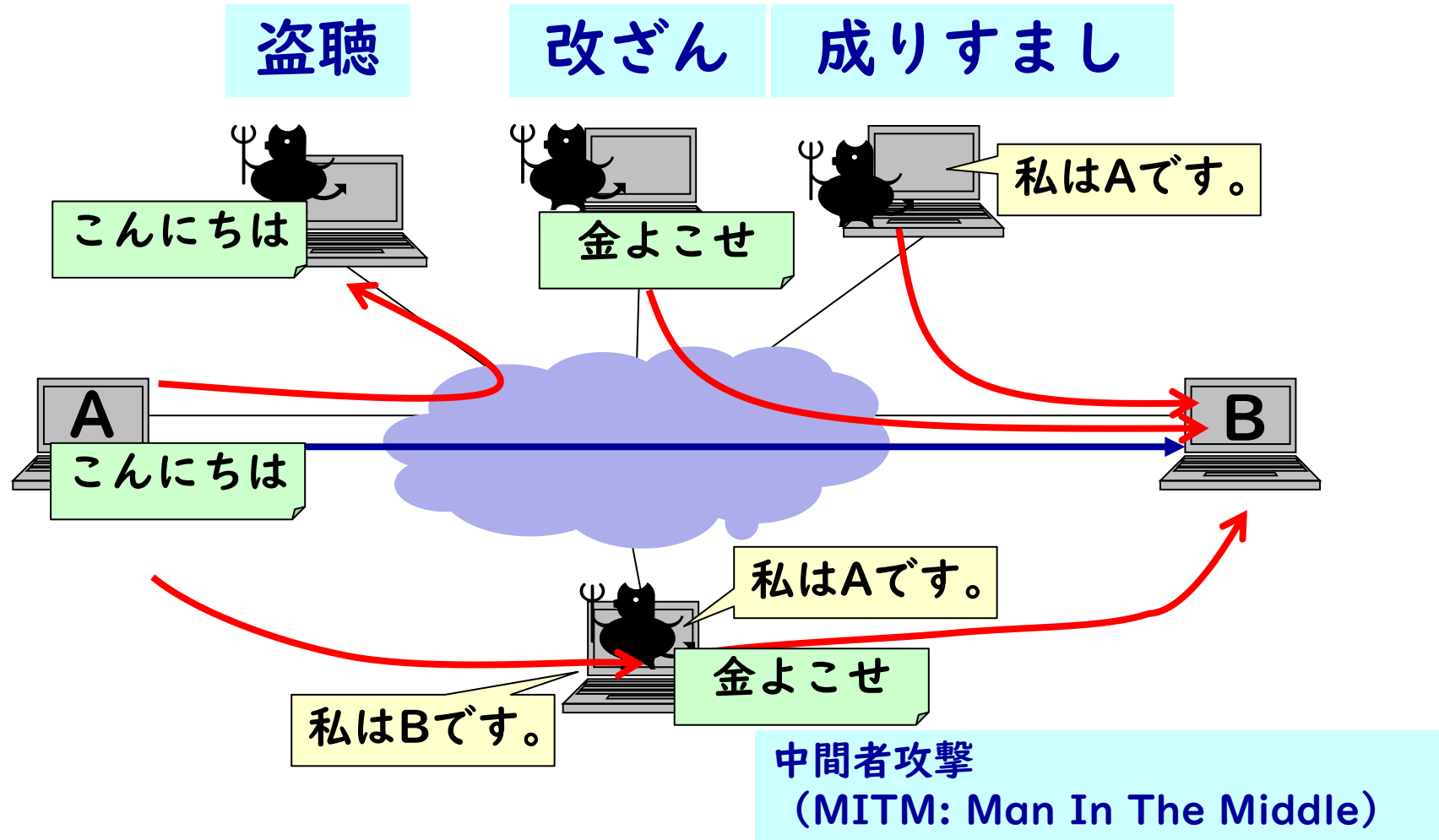
# 授業スケジュール

日時	内容	日時	内容
第1回	ガイダンス	第9回	情報・ネットワークへの脅威と対処
第2回	インターネット上の脅威	第10回	攻撃を誘発する脆弱性
第3回	攻撃の背景	第11回	技術と方法に関する確認
第4回	原因の追究（被害を受ける側の限界）および確認	第12回	セキュリティマネジメント
第5回	認証と認可	第13回	ソーシャルリスクへの対処
第6回	暗号の基礎（1）	第14回	セキュリティマネジメント、ソーシャルリスクに関する確認
第7回	暗号の基礎（2）	第15回	全体のまとめ
第8回	暗号の応用		期末試験

# 本日の目標

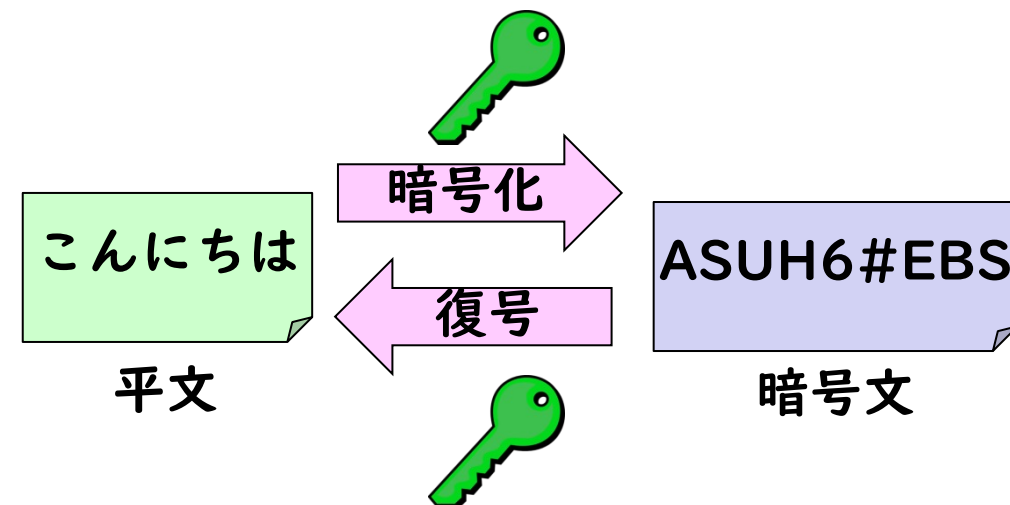
- 現代暗号は、インターネット上で攻撃される側はもとより攻撃する側にとっても重要な技術となっている。今回は、古典暗号と現代暗号の違い、共通鍵暗号および公開鍵暗号の概要について理解する。
- **重要キーワード**
  - 共通鍵暗号、疑似乱数、公開鍵暗号

# 盗聴・なりすまし・改ざん



# 暗号

- 第三者が通信文を見ても特別な知識（方式&鍵）なしでは読めないように変換する技術
- 古典暗号
  - コンピュータ登場以前の暗号
  - 代表的な方式は換字、加算、転置等
  - 方式も当事者間でのみ共有



**The enemy knows the system**

Shannon's maxim / ケルクホフスの原理

方式が他人にバレると解読されるリスクが高まる（安全性評価が不十分なため）

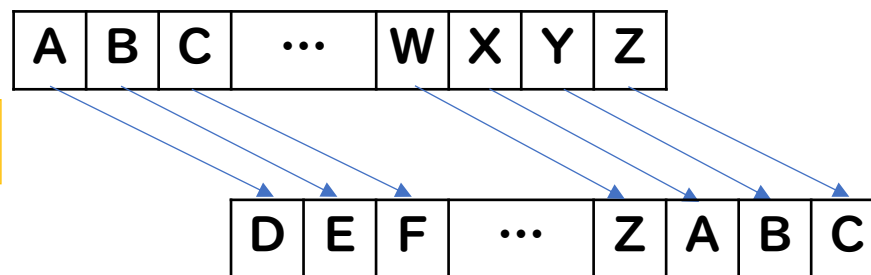
# 【演習】シーザー暗号(シフト暗号)

- 広く知られた換字式古典暗号の代表例

IPUT IS GREAT

→ 3文字ずらす    この場合の鍵は「3」

→ LSXW LV JUHDW



- 簡単な作り方(WSLを利用)

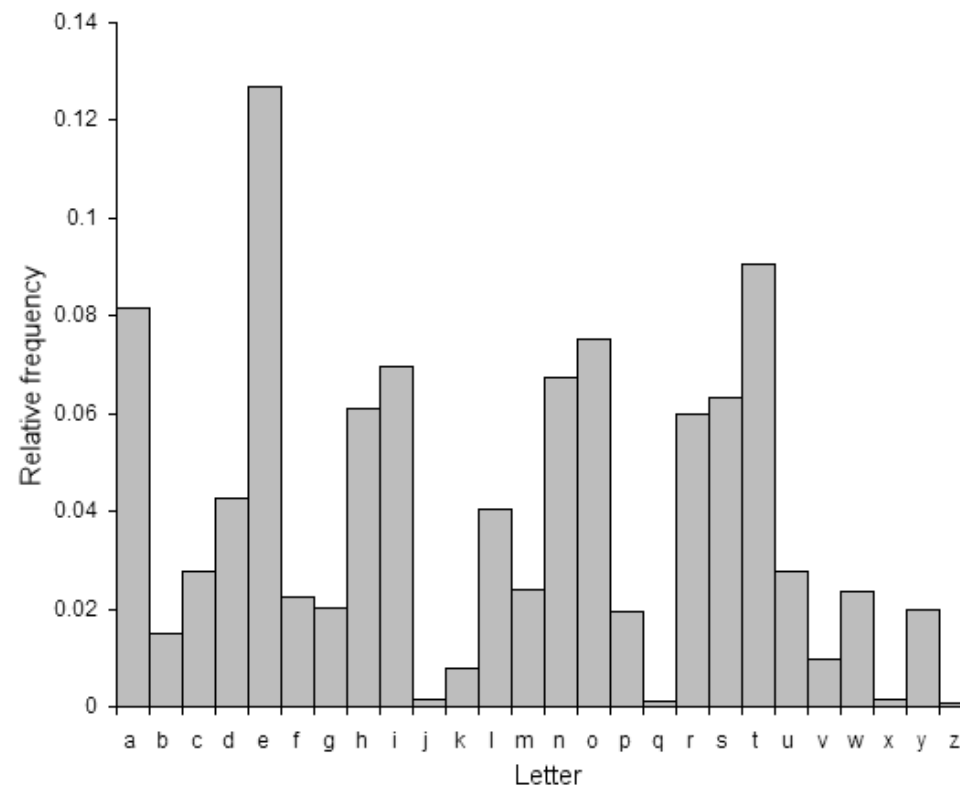
暗号化    \$ echo "IPUT IS GREAT" | tr A-Z D-ZA-C

入力                      置換 3文字ずらす(A-Z → D-ZA-C)

復号    \$ echo "LSXW LV JUHDW" | tr D-ZA-C A-Z

# シーザー暗号/単一換字式暗号の解読

- ブルートフォース攻撃
  - シーザー暗号でかつアルファベットだけならば、わずか26通り
- 出現頻度、単語パターンによる推測
  - 出現文字数の分布等から絞り込みが可能
- 平文と暗号文のペアを入手



Wikipediaより



# (参考)古典暗号が出てくる小説

エドガー・アラン・ポー「黄金虫」  
(1843年)

53 ‡ ‡ † 305))6\*;4826)4 ‡ .)4 ‡ );806\*;48 † 8  
¶ 60))85;1 ‡ (;: ‡ \*8 † 83(88)5\* † ;46(;88\*96  
\*?;8)\* ‡ (;485);5\* † 2:\* ‡ (;4956\*2(5\*—4)8  
¶ 8\*;4069285);)6 † 8)4 ‡ ‡ ;1( ‡ 9;48081;8:8 ‡  
1;48 † 85;4)485 † 528806\*81( ‡ 9;48;(88;4  
( ‡ ?34;48)4 ‡ ;161;:188; ‡ ?;

[https://www.aozora.gr.jp/cards/000094/files/2525\\_15827.html](https://www.aozora.gr.jp/cards/000094/files/2525_15827.html)

アーサー・コナン・ドイル「踊る人形」  
(1903年)



[https://www.aozora.gr.jp/cards/000009/files/50713\\_68371.html](https://www.aozora.gr.jp/cards/000009/files/50713_68371.html)

解読方法、解読結果は、元の小説を是非読んでみましょう！



# Enigma (エニグマ)

- 第2次世界大戦でナチス・ドイツが用いた換字式暗号機
- キーボード入力すると、ランプボードの1文字が点灯。  
それを書き写して通信に使用
- 暗号鍵はローター配置とプラグボードの配線で変更
- 最終的には連合国側が解読に成功
  - 数学（群論）による解析 → 言語学者から数学者への転換
  - ブルートフォース攻撃（アラン・チューリングらが自動化）
  - スパイや捕虜からの暗号機、暗号書の捕獲
  - 使用者の癖などの人間心理面からの分析
  - 交信データの解析



Wikipediaより

# 現代暗号

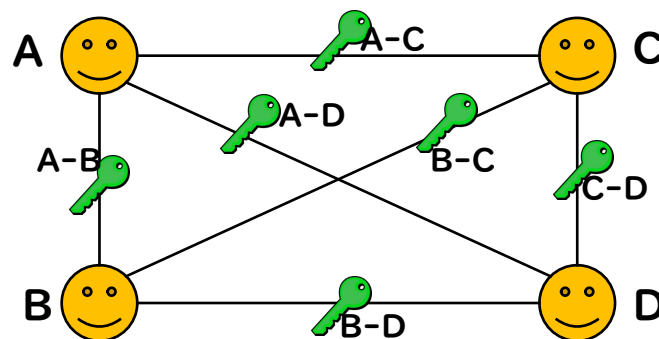
- 鍵データと公開された計算方式により実現
  - 方式が公知なのが前提
- 数学的に解読困難である方式を採用
  - 世界中の研究者が検証することで安全性を担保

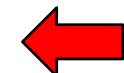

暗号研究は言語学者から数学者へ

- 現代暗号に求められる要件
  - 鍵がなければ復号できない（暗号文から平文の推測が困難）
  - 鍵が容易に盗まれない
  - 鍵の偽造が難しい

# 共通鍵暗号（対称暗号）

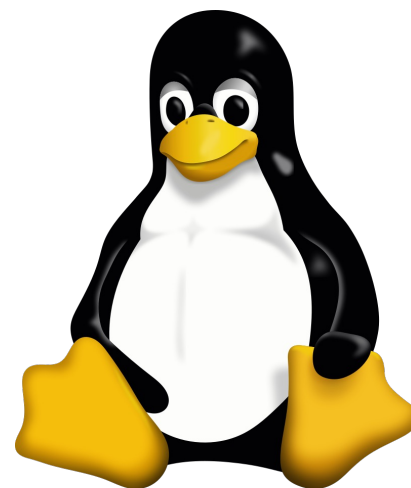
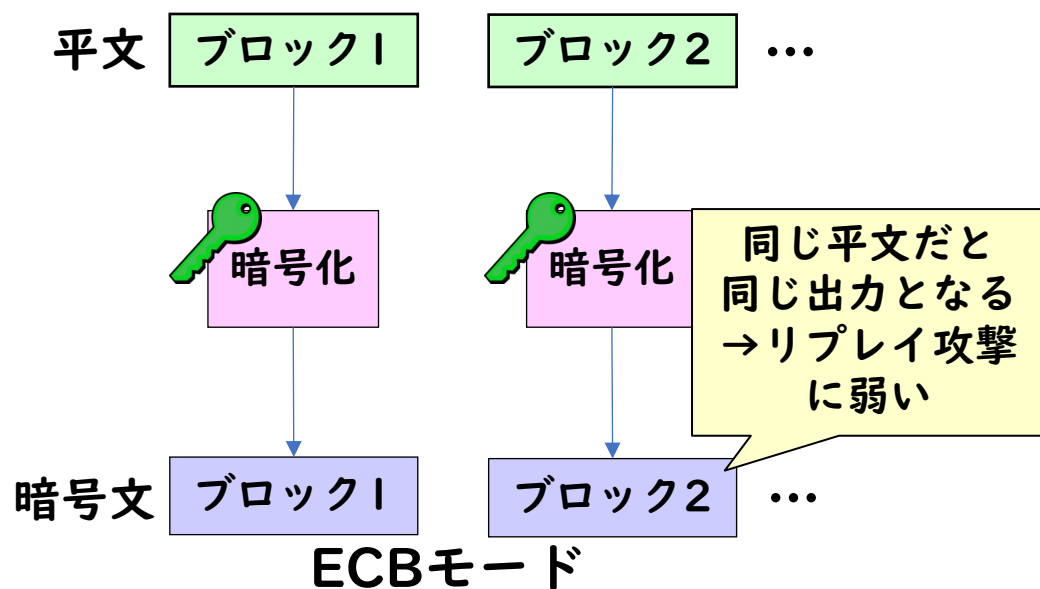
- 暗号化、復号に**共通の鍵データ**を用いる方式



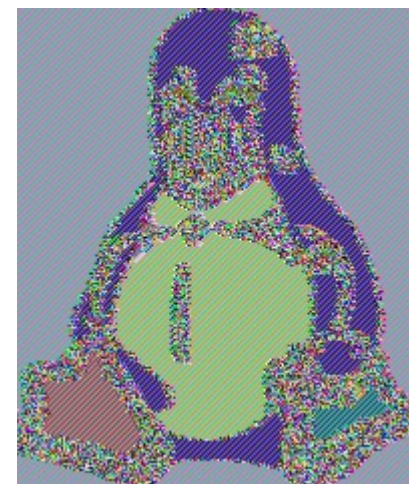
- 
- 
- 接続相手数分の鍵が必要
  - 鍵受渡し時に盗聴等での不正奪取の危険あり

# 共通鍵暗号方式：ブロック暗号

- データを一定の長さごとに区切って暗号化(AES, 3DESなど)
- ECBモード：ブロックごとに個別に暗号化 (Electronic CodeBook)  
→非常に脆弱なので使うべきでない



元画像

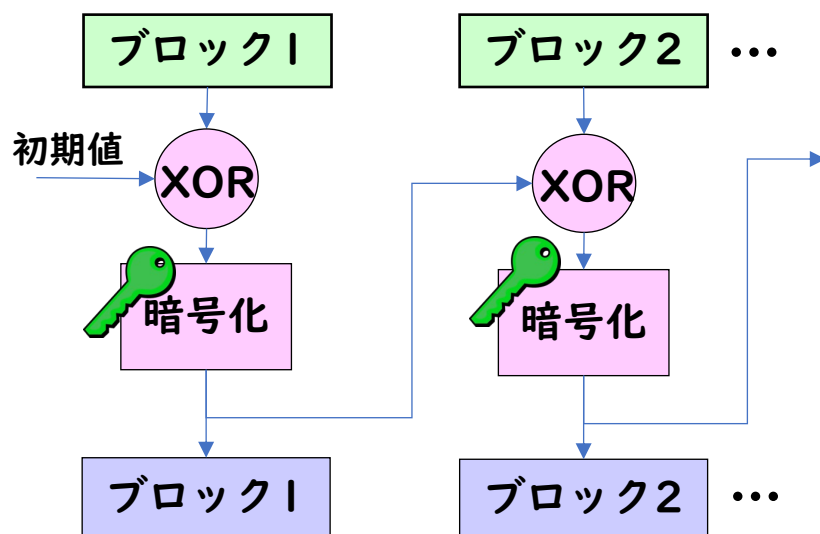


ECBモードで暗号化した画像

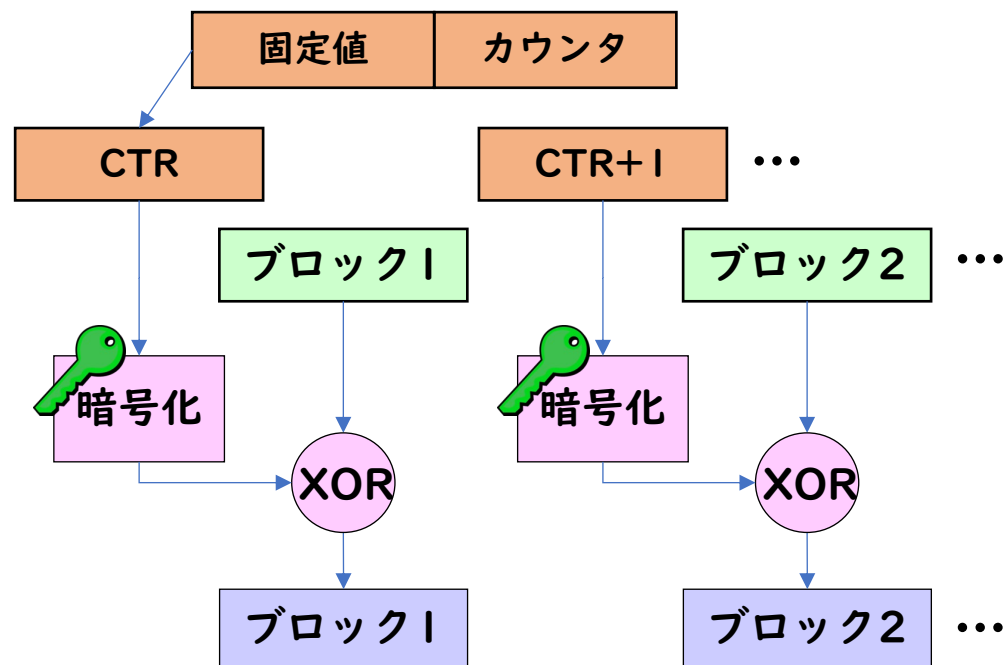
[https://en.wikipedia.org/wiki/Block\\_cipher\\_mode\\_of\\_operation](https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation)

# ブロック暗号のモード

- CBCモード(Cipher Block Chaining)：直前ブロックも使用
- CTRモード(CounTeR)：カウンターを暗号化



CBCモード

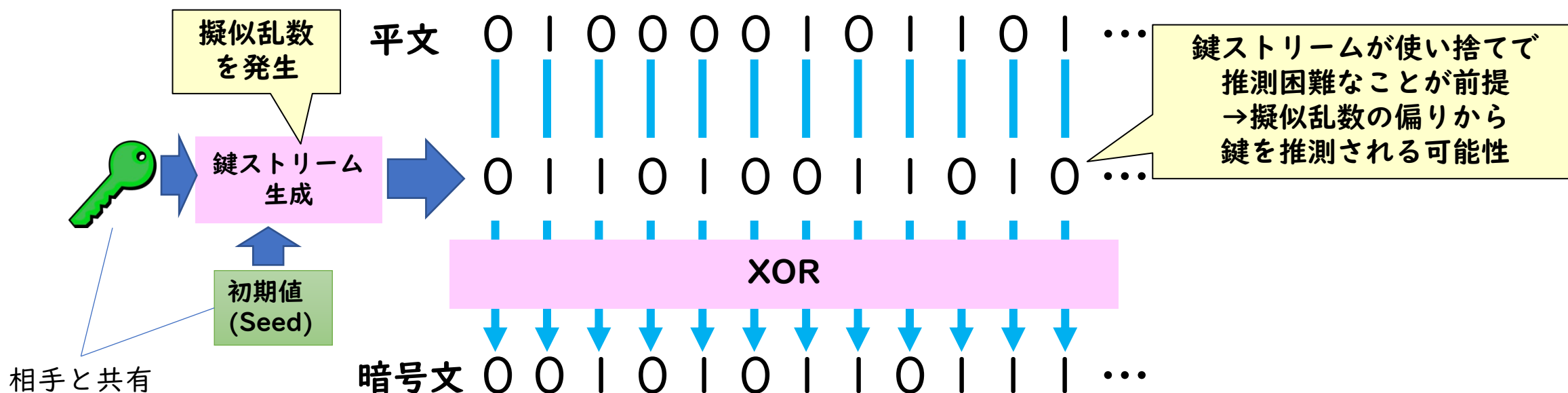


CTRモード

他にも色々なモードあり

# 共通鍵暗号方式：ストリーム暗号

- データをビット／バイト単位で逐次暗号化(RC4, KCipher-2など)
  - ブロック暗号よりも処理が軽いため、小規模の実装向け



# 擬似乱数

- コンピュータプログラムにより乱数を擬似的に再現したもの  
⇔ 真の乱数
- 乱数に求められる性質
  - 無作為性：統計的な偏りや規則性がない
  - 予測不可能性：次に生成される乱数を予測できない
  - 再現不可能性：恣意的な数の生成ができない
- エントロピー（乱雑さ）
  - 外部から持ってきた情報（時刻、ノイズなど）を使って擬似乱数をさらに攪拌



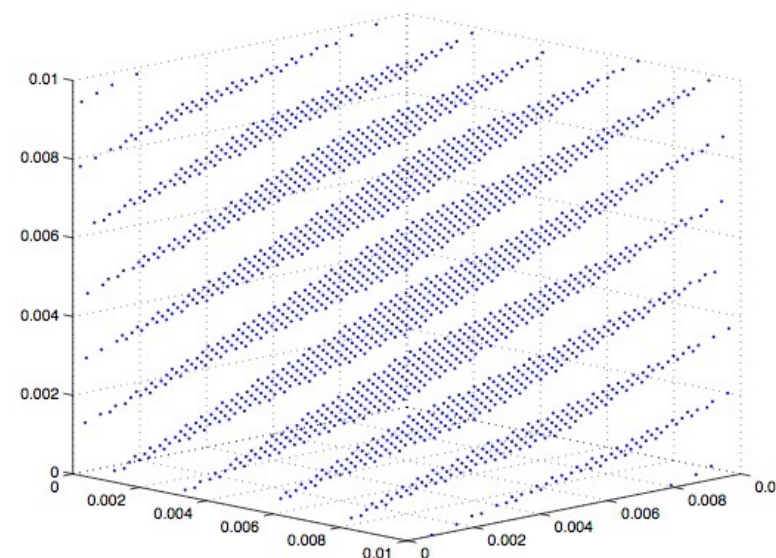
# 擬似乱数の生成例：線形合同法

0～Mの間での擬似乱数Xの生成（整数値）

$$X_{n+1} = (A \times X_n + B) \bmod M$$

- 但し、 $M > A$ 、 $M > B$ 、 $A > 0$ 、 $B \geq 0$
- $X_0$ は任意に設定（Seed値）

- 実装がシンプルで少メモリで動作可能ではあるが、
  - 一定周期で同じ値が生成される  
→ 予測/再現不可能性低い
  - 特に下位のビットについて偏り大  
→ 無作為性低い



[https://omitakahiro.github.io/random/random\\_variables\\_generation.html](https://omitakahiro.github.io/random/random_variables_generation.html)

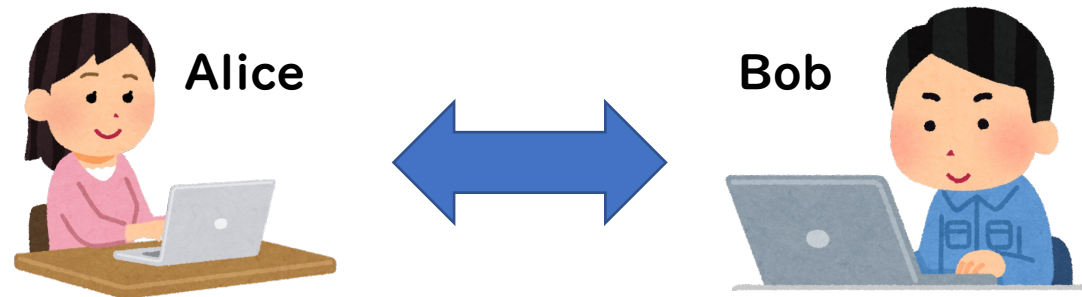
# 暗号の解読

- 鍵の総当たり（ブルートフォース）
- 既知暗号文攻撃
  - 同じ鍵を用いた暗号文を多数入手できたことから鍵を推定
- 既知平文攻撃
  - 平文と暗号文のペアを多数入手できたことから鍵を推定
- 選択平文／暗号文攻撃
  - 攻撃側が恣意的に選んだ平文／暗号文に対する暗号文／平文が入手できたことから鍵を推定

攻撃成功  
しやすさ

# 今回、次回の演習について

- 2人ないし3人のペアで実施してください。
- どうしても難しい場合は、LMSに置いた素材ファイルを活用してください。



- WSL上での操作(コマンド)は授業の中でも多少説明しますが、わからない人は周囲のわかる人に聞きながら進めてください。

# 【演習】 GPGを使った暗号処理

- 本日の演習ではWSL上のGPG(GnuPG)を用います。

```
$ gpg --version
```

```
gpg (GnuPG) 2.2.27
```

```
libgcrypt 1.9.4
```

```
(省略)
```

```
Home: /home/(ユーザ名)/.gnupg
```

```
# GPGのバージョン
```

```
# ライブラリのバージョン
```

```
# 鍵データの保管先ディレクトリ
```

```
$ ls ~/.gnupg
```

```
ls: cannot access '/home/(ユーザ名)/.gnupg': No such file or directory
```

もし.gnupgが既に存在する(上記のエラーにならない)場合は、一旦退避しておく。

```
$ mv ~/.gnupg ~/.gnupg-orig
```

# 【演習】 共通鍵暗号を使う

## ① WSL上でGPGを使って共通鍵暗号化

```
$ echo "IPUT is great." > test.txt # 適当なファイルを作成
```

```
$ cat test.txt
```

```
IPUT is great.
```

```
$ gpg -c -a --cipher-algo AES test.txt # AESで暗号化の場合  
(パスフレーズを聞かれるので適当に入力)
```

```
$ cat test.txt.asc
```

```
-----BEGIN PGP MESSAGE-----
```

```
(よくわからない文字列)
```

```
-----END PGP MESSAGE-----
```

暗号化されたデータ  
(-aでテキスト化)

①パスフレーズ(共通鍵)  
で暗号化する

②暗号化した  
データを送る

test.txt.asc

①パスフレーズを  
別途伝えておく

③教えてもらった  
パスフレーズで  
復号する

## ② Slack DM等で上記暗号化ファイルを相手に送付

- ・ パスフレーズ (共通鍵) は別ルートで伝えておく

# WSLとWindowsのファイルやりとり

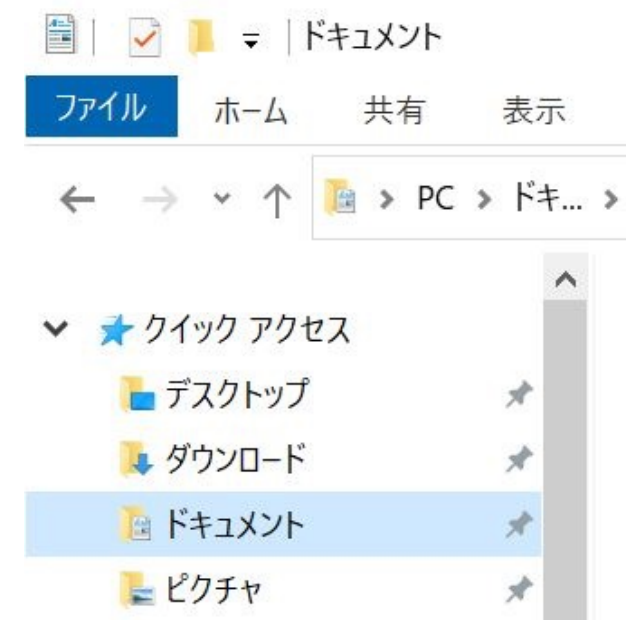
- WSL上ではWindows上のファイルは以下のディレクトリ上に見える
  - /mnt/c/Users/(Windowsログイン名)/xxxx
  - xxxxの部分は以下のように対応
    - デスクトップ : Desktop
    - ダウンロード : Downloads
    - ドキュメント : Documents

- 例えば、ダウンロードフォルダ上で作業するには

```
$ cd /mnt/c/Users/(Windowsログイン名)/Downloads
```

```
$ ls
```

(ダウンロードフォルダのファイル一覧が表示されるはず)



# 【演習】共通鍵暗号を使う（続き）

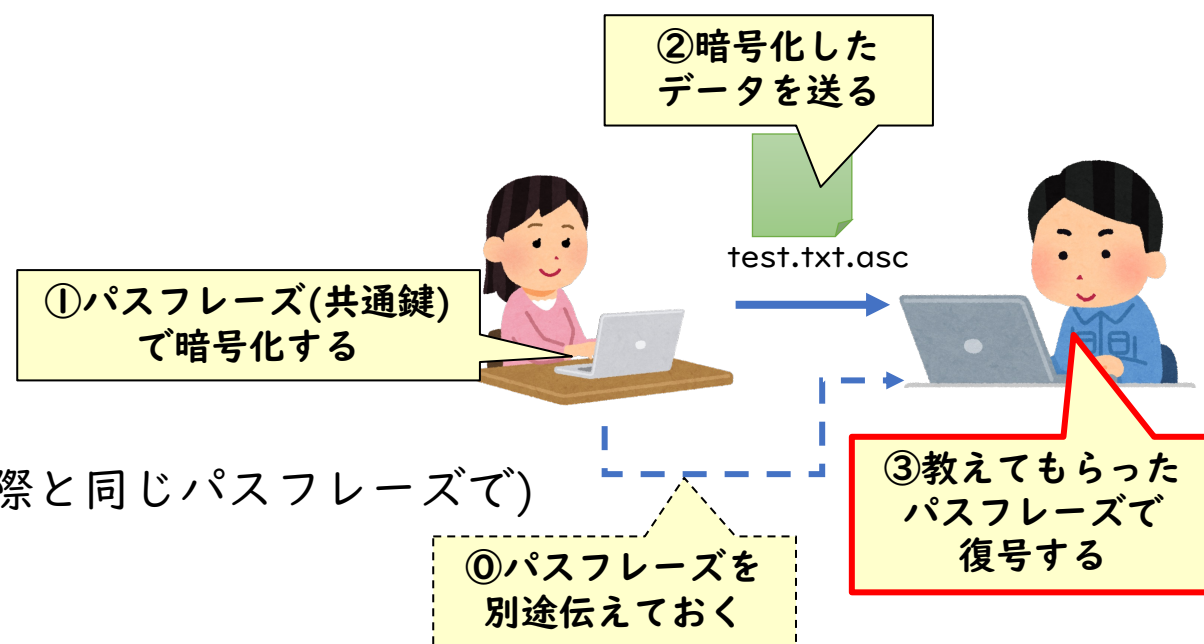
## ③ WSL上でGPGを使って復号

```
$ cat test.txt.asc          # もらったファイルの中身を確認  
-----BEGIN PGP MESSAGE-----
```

(よくわからない文字列)

```
-----END PGP MESSAGE-----
```

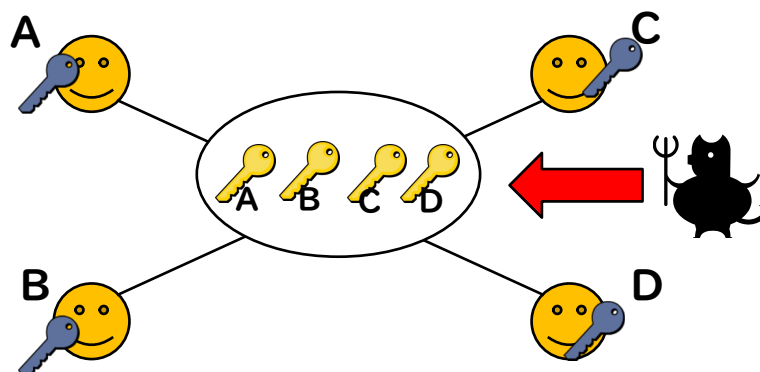
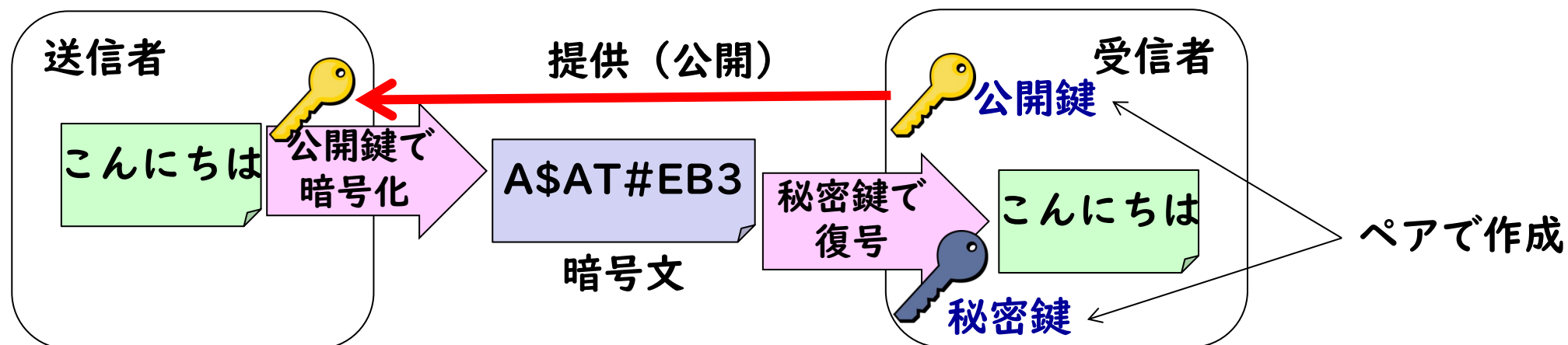
```
$ gpg -d test.txt.asc        # 復号 (暗号化の際と同じパスフレーズで)
```





# 公開鍵暗号（非対称暗号）

- 鍵を**ペアで作成**して一方を公開する方式(RSA, EdDSAなど)



秘密鍵は端末の外には  
出ないので、盗聴での  
奪取は困難

# RSA暗号

- Ron **R**ivest, Adi **S**hamir, Leonard **A**dlemanが開発した、桁数が多い合成数の**素因数分解**の困難性を安全の根拠とする公開鍵暗号方式
- 鍵ペアの生成方法
  - 秘密鍵( $p, q$ ) :  $p, q$ はいずれも素数
  - 公開鍵( $n, e$ ) : いずれも整数
    - $n = p \times q$ 。但し $n$ はメッセージ(平文)よりも大きい数字である必要
    - $e$ は $(p-1)$ とも $(q-1)$ とも互いに素となる (最大公約数が1となる)数
- 暗号化・復号
  - 暗号文 = 平文 <sup>$e$</sup>  mod  $n$  (平文の $e$ 乗を $n$ で割った余り)
  - 平文 = 暗号文 <sup>$d$</sup>  mod  $n$ 
    - $d \times e - y \times L = 1$  となる自然数解  $d, y$  を求める。
    - $L$ は $(p-1)$ と $(q-1)$ の最大公約数 → 秘密鍵がないと解けない！

# RSA暗号の強度：素因数分解

- 問題：7,240,879を素因数分解せよ

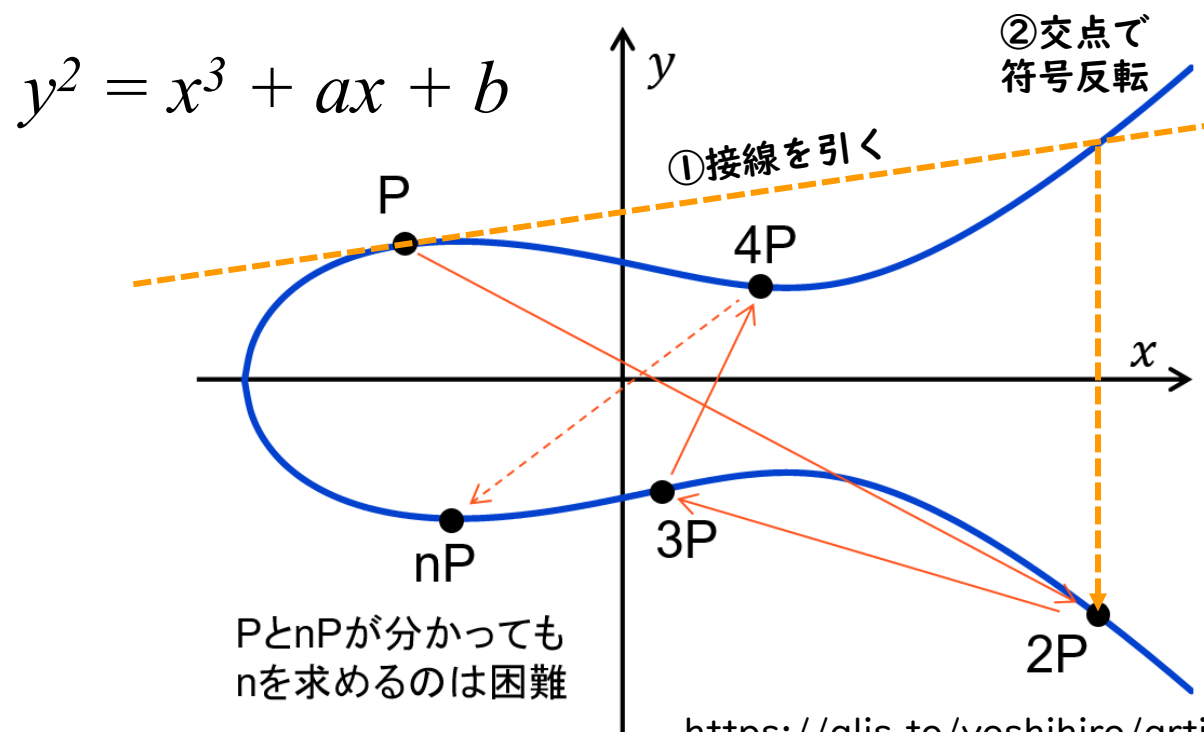
→ 答え：1,907×3,797

- なぜ難しい？

- 数学的には多項式時間で解く方法がないと言われている
  - すなわち、数が大きくなるほど計算の困難さが指数的に増える
- とはいえ、総当たりよりは効率的な（準指数時間で解ける）計算方法  
はあり
  - 2010年 NTTが768ビット(232桁)の解読成功
  - 2020年時点の世界記録は829ビット(250桁)

# 楕円曲線暗号

- 楕円曲線上の離散対数問題の困難性を安全性の根拠とする公開鍵暗号方式 (EdDSAなど)
- RSAよりも少ないビット数で強度の確保が可能



この例では  
秘密鍵：n  
公開鍵：P, nP  
として使える

<https://alis.to/yoshihiro/articles/KmQYMwQbNPOZ>

# 【演習】RSA公開鍵を生成する

## ① WSL上でGPGを使って鍵ペアを生成

```
$ gpg --gen-key
```

(名前、メールアドレス、パスフレーズを入力。

このパスフレーズは人に教えない！！)

```
$ gpg --list-keys # 公開鍵のID確認
```

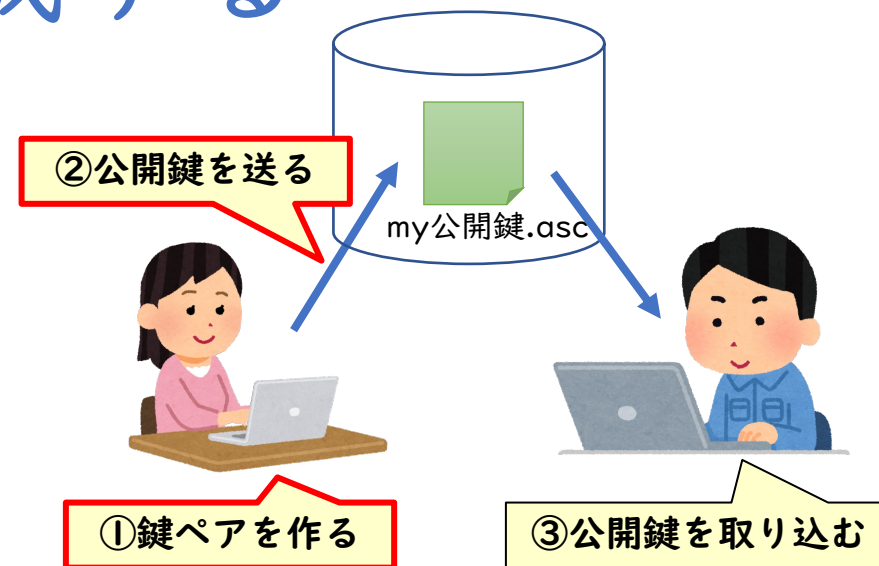
```
$ gpg --list-secret-keys # 秘密鍵のID確認
```

#公開鍵、秘密鍵は ~/.gnupg/pubring.kbx に格納される。

```
$ gpg --export --armor (公開鍵のID) > TKI23456-abcd.asc #適当なファイル名で保存
```

## ② エクスポートしたファイルを作業用フォルダに格納する

- ファイル名は 学籍番号-ローマ字名前.asc お願いします。



# 【注意】公開鍵の生成について

- 生成時には必ず適切なパスフレーズを入力すること！
  - 生成された公開鍵、秘密鍵は同じ場所に格納される。
  - すなわち、秘密鍵も取り出して人に渡すことが可能
  - 秘密鍵が不用意に外部に漏れると、鍵の安全性が失われる
- パスフレーズは秘密鍵を安全に保管するための重要な手段
  - 秘密鍵を共通鍵（パスフレーズ）で暗号化することで、万が一外部に漏れても悪用を防ぐことができる。
  - 但し、漏れた秘密鍵へのブルートフォース攻撃は容易なので、それに耐えうるパスフレーズを付けておく必要あり