CS第1　レポート課題3

課題　暗号解読に挑戦

本日の講義内容

1. プログラミング言語における関数　宿題
2. 暗号通信とは
3. レポート課題3（予告）
   - 課題の説明
   - 解読法のヒント
4. 現代の暗号通信方法

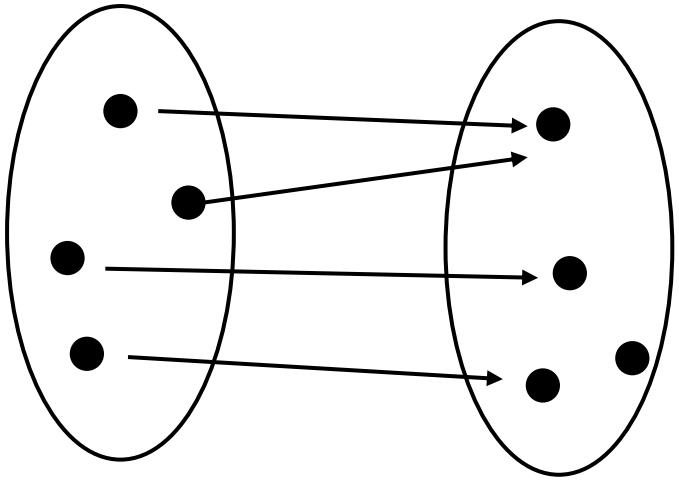
CS第1　演習ガイド

1. Terminal を動かす
2. 講義のウェブページから プログラム をダウンロードする.
   - Downloads（ダウンロード）フォルダに day5.zip が展開される
3. Terminal で day5 をCS1 に移動
   - cd Documents/CS1
   - mv ~/Downloads/day5 ./
   - cd day5

1. プログラミング言語における関数

数学の関数

定義域の値を一つの値域の値に対応付ける関係



例） $\sin(x) = y$

角度　対応する三角比

プログラミング言語における関数

引数から値を計算するプログラム

```
def mult(x, y):
    seki = 0
    while y > 0:
        seki = seki + x
        y = y - 1
    return seki
```

mult : 関数名
x, y : (仮)引数

注: 関数をサブルーチン，メソッドという場合もあります

関数を使ってみよう

プログラミング言語における関数

引数から値を計算する手続き

```
def mult(x, y):
    seki = 0
    while y > 0:
        seki = seki + x
        y = y - 1
    return seki
```

mult : 関数名
x, y ：(仮)引数

```
$ python -i mult.py
>>> mult(3,5)
15
>>> mult(mult(3,5), 4)
60
```

- mult.py を読んでPython を対話的に使う
- mult(3,5) : 関数適用, 3と5が実引数

Python での関数定義

```
def 関数名(引数, ..., 引数):
    ...
    y = ...
    return y
```

← 目標の関数値を計算するブロック

← return 命令で計算を終了して関数値を返す.

※ return 命令は複数個所にあってもよい.

単純な例:

```
def add1(x):
    return x+1
```

例：関数

復習：掛算の計算

mult.py

```
x = 入力データ
y = 入力データ
seki = 0
while y > 0:
    seki = seki + x
    y = y - 1
print(seki)
```

a + b の計算

```
a = seki
b = x
wa = a
while b > 0:
    wa = wa + 1
    b = b - 1
seki = wa
```

```
x = 入力データ
y = 入力データ
seki = 0
while y > 0:
  a = seki
  b = x
  wa = a
  while b > 0:
    wa = wa + 1
    b = b - 1
  seki = wa
  y = y - 1
print(seki)
```

例：関数

復習：掛算の計算

mult.py

```
x = 入力データ
y = 入力データ
seki = 0
while y > 0:
    seki = seki + x
    y = y - 1
print(seki)
```

mult.py

```
x = 入力データ
y = 入力データ
seki = 0
while y > 0:
    seki = add(seki, x)
    y = y - 1
print(seki)
```

add(a, b) の定義

```
def add(a, b):
    wa = a
    while b > 0 :
        wa = wa + 1
        b = b - 1
    return wa
```

実際には
これが実行される

まるで，数学の「関数」の
ように使うことができる

練習：Python での関数定義

整数の列の平均を計算

average.py

```
def sum(a):
    n = len(a)
    # 以下が総和の計算部分
    return s

a = list(map(int, input().split()))
print(sum(a) // len(a))
```

参考

```
a = list(map(int, input().split()))
n = len(a)
# 以下が総和の計算部分
s = 0
for k in range(n):
   s = s + a[k]
print(s)
```

実行例

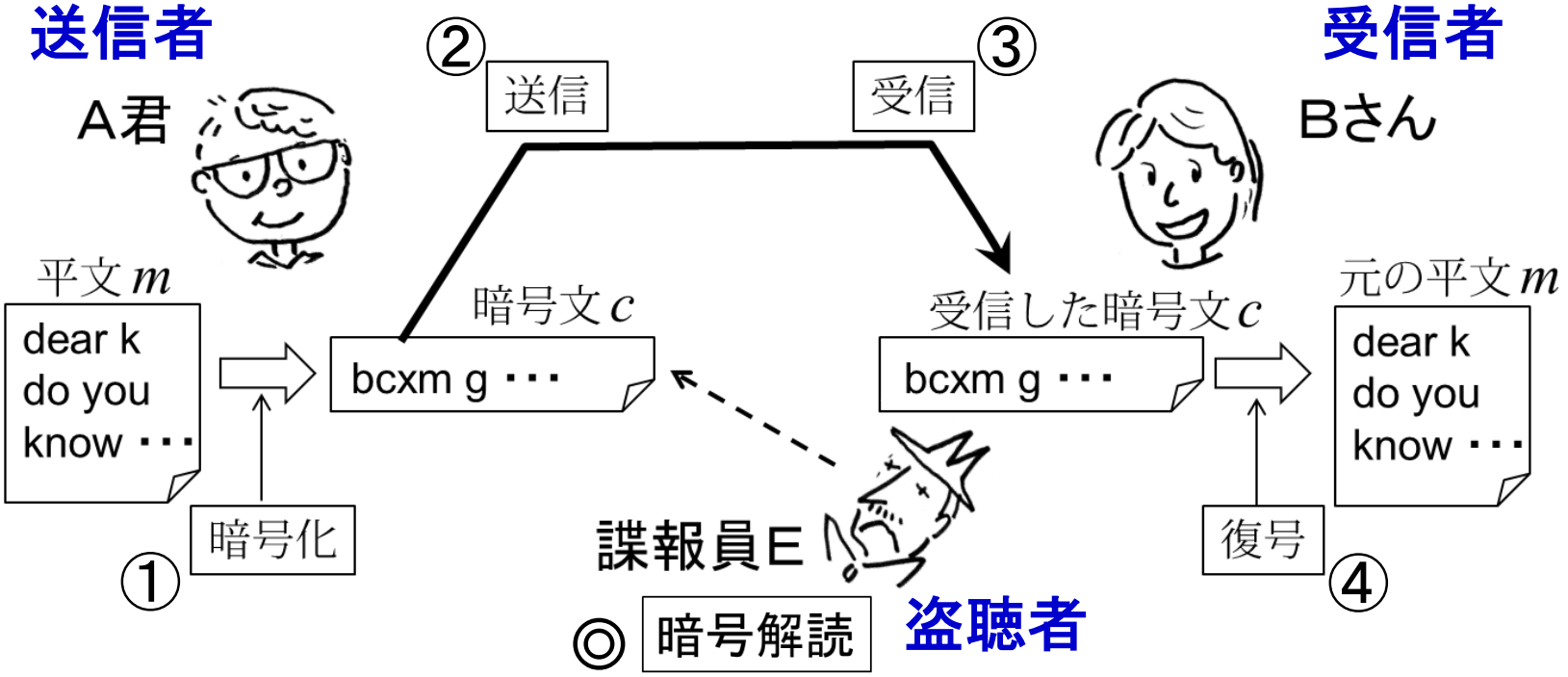
```
$ python average.py
1 2 3 4 5 6 7
4
$ python average.py
6 5 4 3 2 1
3
```

2. 暗号通信

通信文を見られても，その内容がわからないように符号化して通信すること

データを保管する場合など
必ずしも通信しない場合もある

暗号通信の基本的な流れ



2. 暗号通信

暗号方式 ⟷ 暗号文を作る方法（**暗号化法**，**復号法**）
（より一般的には，暗号通信のやり方）

例） **シーザー暗号**：ローマ皇帝シーザーが使ったと言われる方式
エニグマ：第二次世界大戦時にドイツ軍が使った方式
DES, AES：現在使われている代表的な暗号方式

シーザー暗号は各文字をアルファベット上で k 字シフト換字（k 字先の文字に換えること）して暗号を作る暗号方式のこと.

例）$k = 3$　英小文字だけを対象とする

Good bye !
↓
Grrg ebh !

a	b	c	d	e	f	g	h	...	w	x	y	z
↓	↓	↓	↓	↓	↓	↓	↓	...	↓	↓	↓	↓
d	e	f	g	h	i	j		...	z	a	b	c

2. 暗号通信

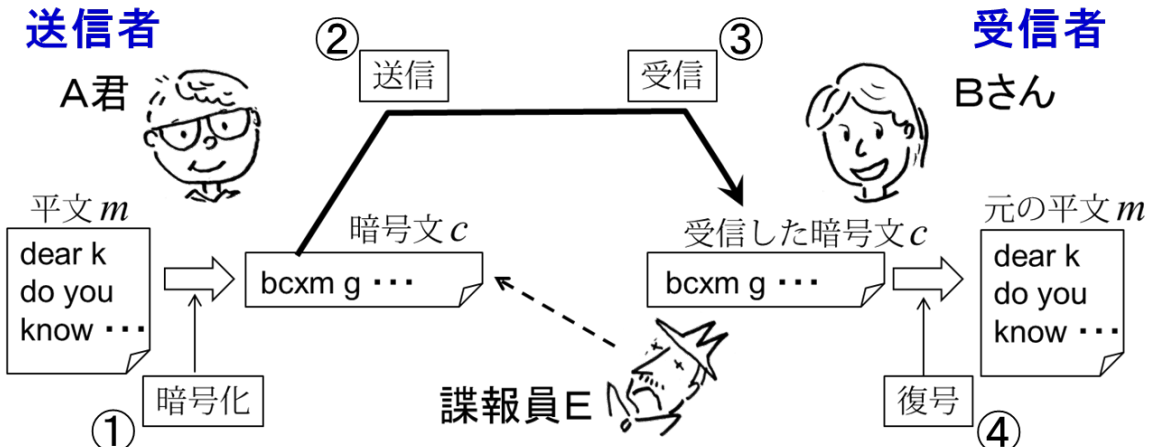
暗号方式 ⟷ 暗号文を作る方法（**暗号化法**，**復号法**）
（より一般的には，暗号通信のやり方）

暗号化 ＝ 暗号文を作ること
復号 ＝ 暗号文から平文に戻すこと
秘密鍵 ＝ 暗号化・復号に必要な鍵
シーザー暗号では，ずらす文字数 k

$k = 3$

a	b	c	d	e	f	g	h	…	w	x	y	z
↓	↓	↓	↓	↓	↓	↓	↓	…	↓	↓	↓	↓
d	e	f	g	h	i	j	…		z	a	b	c

平文 Good bye ! ⟷（暗号化 / 復号）**暗号文** Grrg ebh !



2. 暗号通信

平文 Good bye ! ⇄（暗号化 / 復号）Grrg ebh ! 暗号文

暗号化と復号を計算で表わそう

まずは計算の目標（プログラムの仕様）を数学の関数として表す

※ 計算法は不要 !!

暗号化用関数

$\mathrm{enc}_{\mathrm{caesar}}$(秘密鍵 k, 平文 m)
= k 字シフト換字して作った暗号文 c

復号用関数

$\mathrm{dec}_{\mathrm{caesar}}$(秘密鍵$k$, 暗号文 c)
= k 字逆シフト換字して戻した平文 m

$\mathrm{enc}(3, \text{"Good"}) = \text{"Grrg"}$　　$\mathrm{dec}(3, \text{"Grrg"}) = \text{"Good"}$

2. 暗号通信

ango.py

```
def enc(k, m):
  計算を書く
  c = ””
  return c
end
##### プログラム本文 #####
k = 3
hirabun = input()
angobun = enc(k, hirabun)
print(angobun)
```

では，どう書くか？

宿題

考え方

1. 添え字 i の文字を平文 m から取ってくる
2. その文字を暗号化した文字を，変数 c の文字列に連結

復号化関数も
同様にプログラム化しよう

2. 暗号通信　参考

code.py

```
code_a = 97              # 文字 a の文字コード
kosu = 26                # 英字アルファベットの数

bun = input()            # 入力文字列
leng = len(bun)

print("** 文字, 文字コード, a からの差分**")
for i in range(leng):
    moji = bun[i]                    # bun の i 文字目を得る (i は 0から始まる)
    code = ord(moji)                 # その文字のコードを得る
    sabun = code - code_a
    if 0 <= sabun and sabun < kosu:  # 小文字アルファベットならば
        print(moji, ": ", code, ", ", sabun)   #   情報を表示する
    else:                            # そうでないときは
        print(moji, ": ", code)      #   差分は表示しない
```

2. 暗号通信

参考

code2string.py

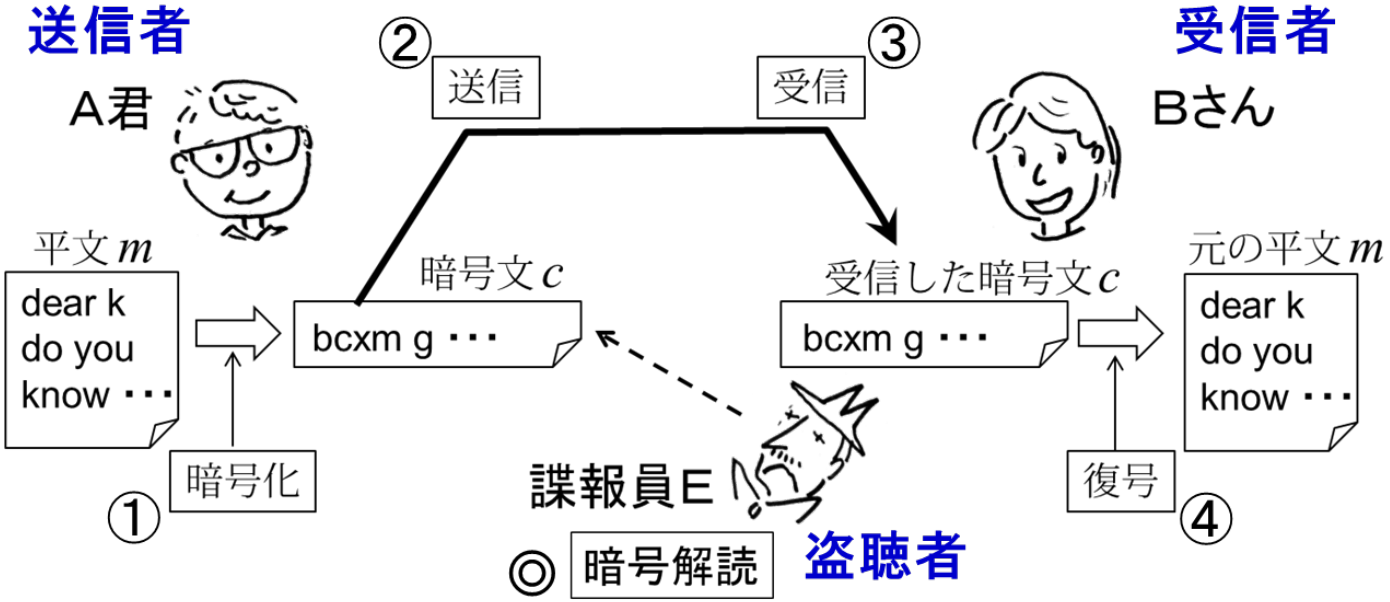
```
a = list(map(int, input().split()))   # 整数の列を配列 a に入力
s = ""
n = len(a)
for i in range(n):
    s = s + chr(a[i])          # i 番目の整数を文字に変換し文字列sの最後に連結
print(s)
```

実行例

```
$ python code2string.py
97 98 99 122
abcz
```

3. レポート課題3（予告）

※詳細は次週説明する



暗号解読 ⟷ 秘密鍵を知らない者が暗号文から平文を得ること

ヒント

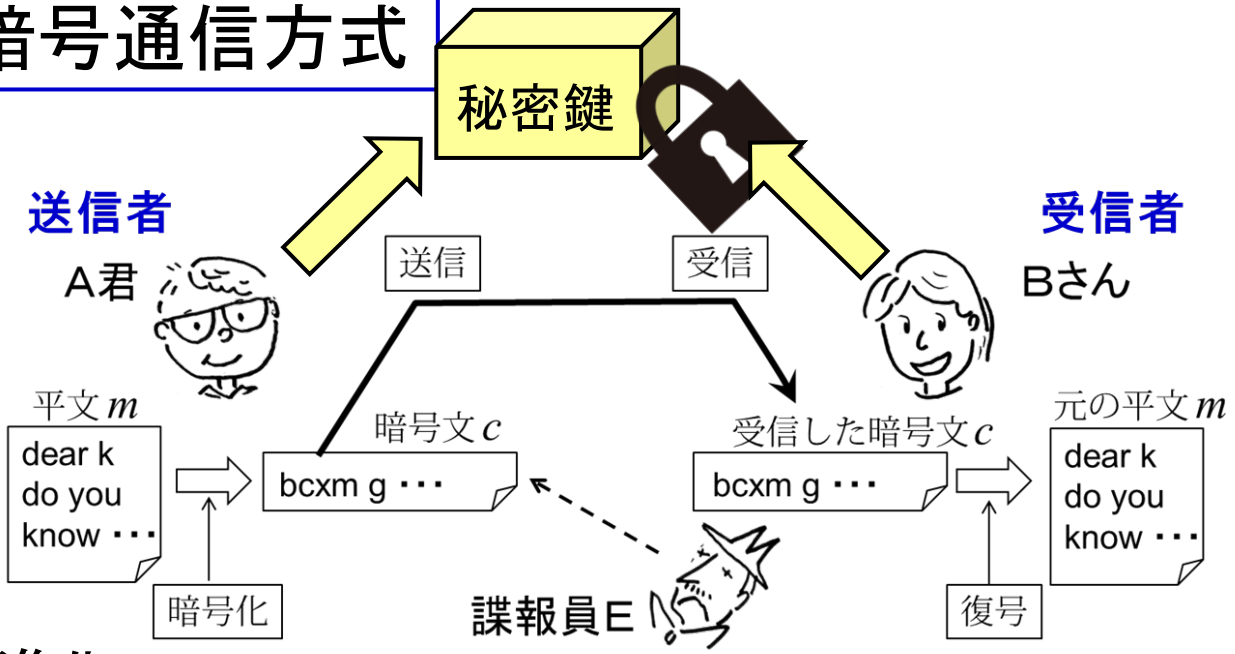
「踊る人形」
コナン・ドイル作

明らかだよ
ワトソン君

考えて来て下さい

比較的長い英文を
暗号化したものを解読する
という前提で考えてよい

4. 現代の暗号通信方式



暗号方式の進化

シーザー暗号: ローマ皇帝シーザーが使ったと言われる方式
エニグマ: 第二次世界大戦時にドイツ軍が使った方式
DES, AES: 現在使われている代表的な暗号方式

1980 年頃

秘密鍵暗号方式

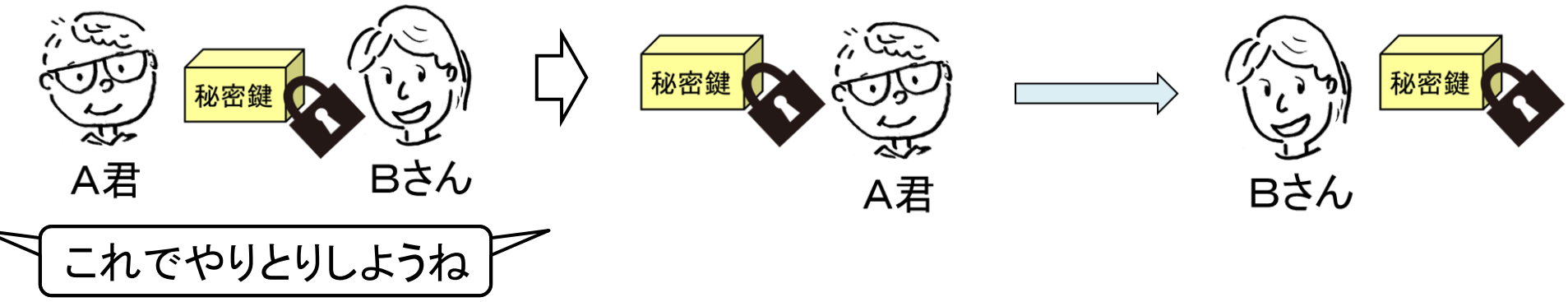
公開鍵暗号方式

公開鍵・・・皆に知らせてよい鍵，暗号化に使う

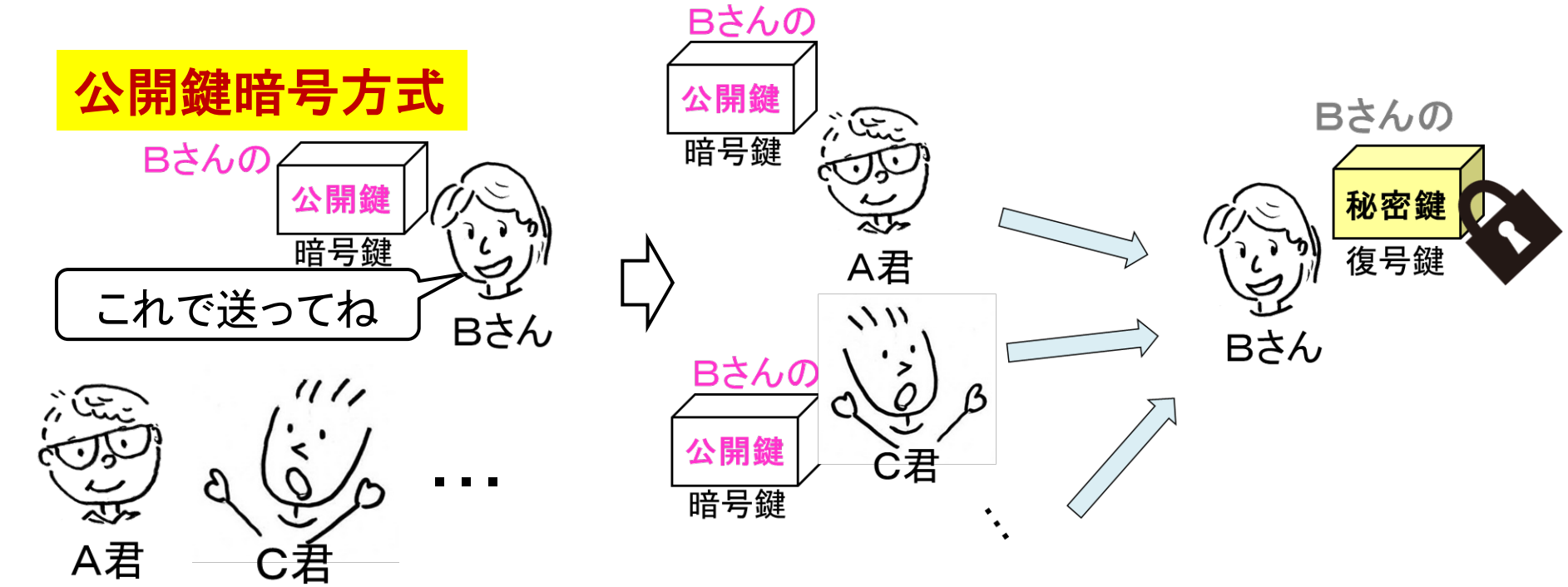
秘密鍵・・・復号に使う

4. 現代の暗号通信方式

秘密鍵暗号方式

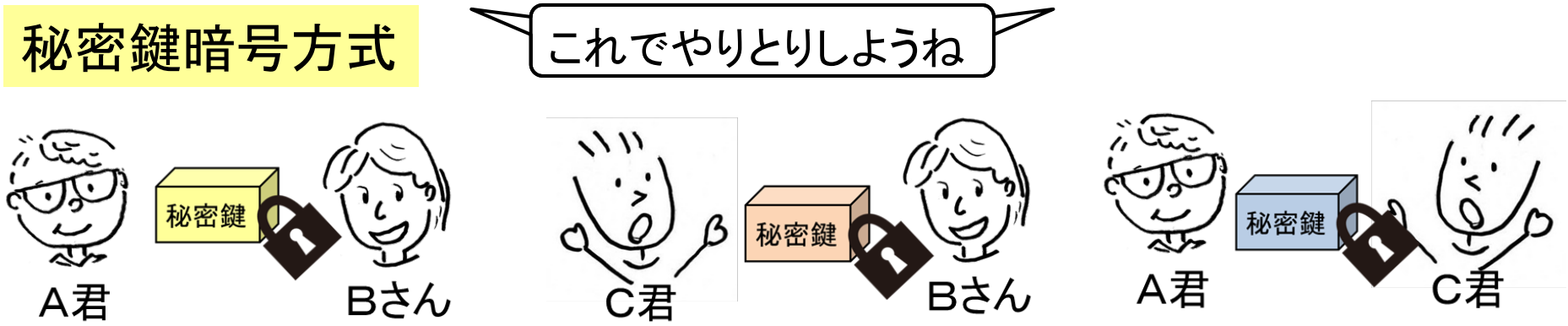


公開鍵暗号方式

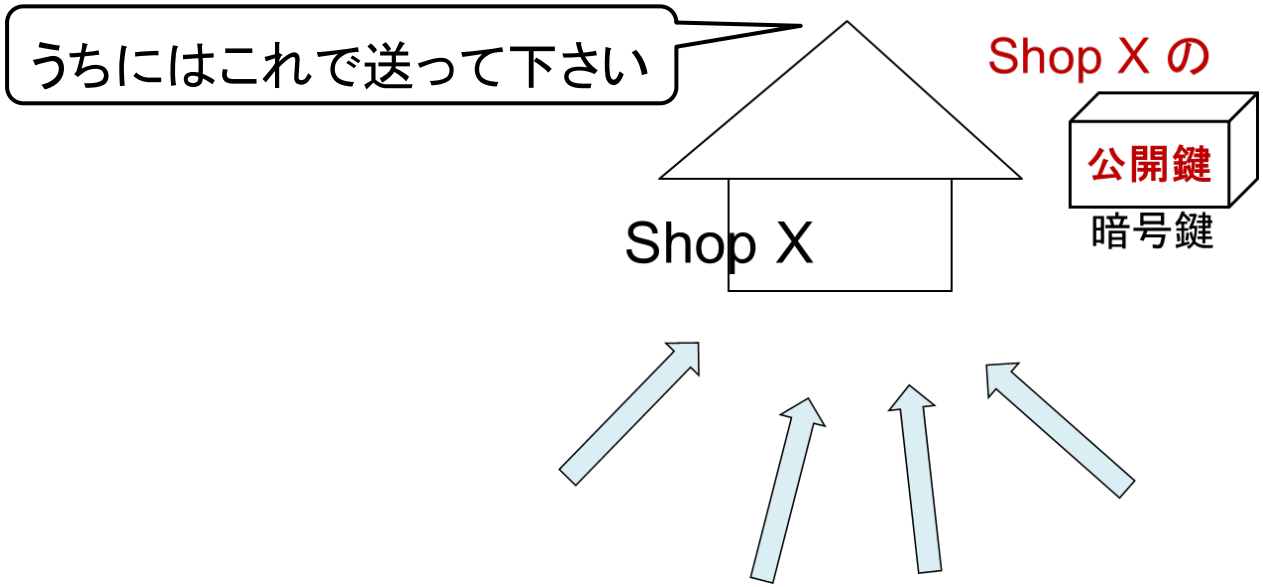


4. 現代の暗号通信方式

秘密鍵暗号方式



公開鍵暗号方式



4. 現代の暗号通信方式

公開鍵暗号方式