

課題 暗号解読に挑戦

本日の講義内容

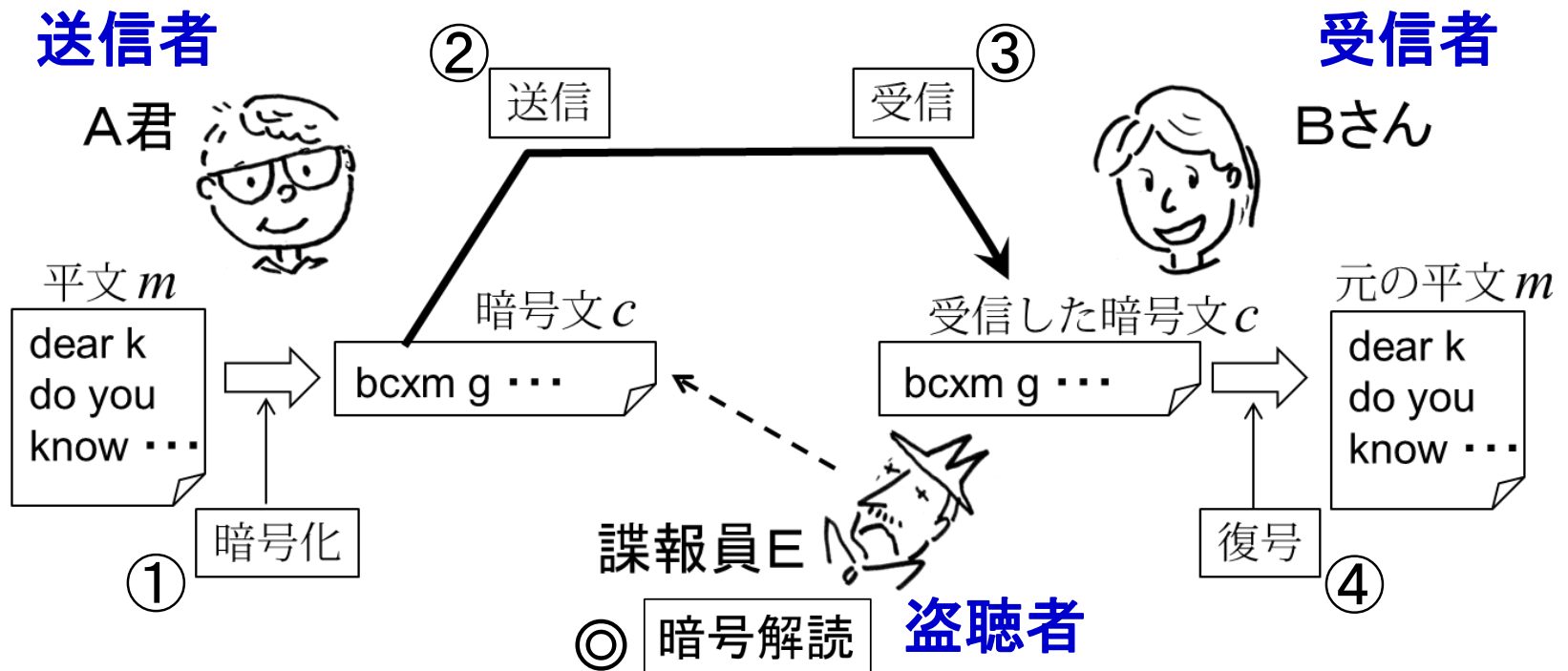
1. 暗号通信とは 教科書 5.3
2. 関数, サブルーチン 宿題
暗号
3. レポート課題3 (予告)
 - 課題の説明
 - 解読法のヒント 教科書 5.3
4. 現代の暗号通信方法

1. 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



1. 暗号通信

暗号方式 \longleftrightarrow

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

1. 暗号通信

暗号方式 \longleftrightarrow 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

暗号文

Grrg ebh !

復号

送信者

A君

平文 m

dear k
do you
know ...



①

暗号化

②

送信

暗号文 c

bcxm g ...

③

受信

受信した暗号文 c

bcxm g ...

受信者

Bさん

元の平文 m

dear k
do you
know ...



④

復号

諜報員E



2. 関数とサブルーチン

平文 Good bye ! $\xrightleftharpoons[\text{復号}]{\text{暗号化}}$ Grrg ebh ! 暗号文

暗号化と復号を計算で表わそう

まずは計算の目標を関数として表す

※ 計算法は不要 !!

数学で出てくる関数とは
何かを何かに対応させる関係

例) $\sin(x) = y$
 ↑ ↑
 角度 対応する三角比

暗号化用関数

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

復号用関数

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

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

2. 関数とサブルーチン

関数 \longleftrightarrow 何かを何かに対応させる関係, 計算の目標を表す

サブルーチン(Ruby では「関数」と呼ばれている)

\longleftrightarrow 関数をどうやって計算するかのパログラムを書いたもの

復習: 掛算の計算

mult.rb

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

a + b の計算

```
wa = a
while b > 0
  wa = wa + 1
  b = b - 1
end
wa が答え
```

```
x = 入力データ
y = 入力データ
seki = 0
while y > 0
```

```
  wa = seki; b = x;
  while b > 0
    wa = wa + 1
    b = b - 1
  end
  seki = wa
  y = y - 1
```

```
end
puts(seki)
```

2. 関数とサブルーチン

復習: 掛算の計算

mult.rb

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

mult.rb

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

add(a, b) の定義

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

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

2. 関数とサブルーチン

関数 \longleftrightarrow 何かを何かに対応させる関係, 計算の目標を表す

サブルーチン (Ruby では「関数」と呼ばれている)

\longleftrightarrow 関数をどうやって計算するかのパログラムを書いたもの

サブルーチンの Ruby での書き方

ango.rb

```
def enc(k, m)
  計算を書く
  c = ...
  return(c)
end
```

プログラム本文

```
k = 3 # 暗号鍵の設定
hirabun = gets.chomp # 平文を入力
angobun = enc(k, hirabun) # 暗号文に変換
puts(angobun) # 暗号文を出力
```

サブルーチン enc の
定義部分. enc という
関数をどうやって計算
するか, をここに書く.

2. 関数とサブルーチン

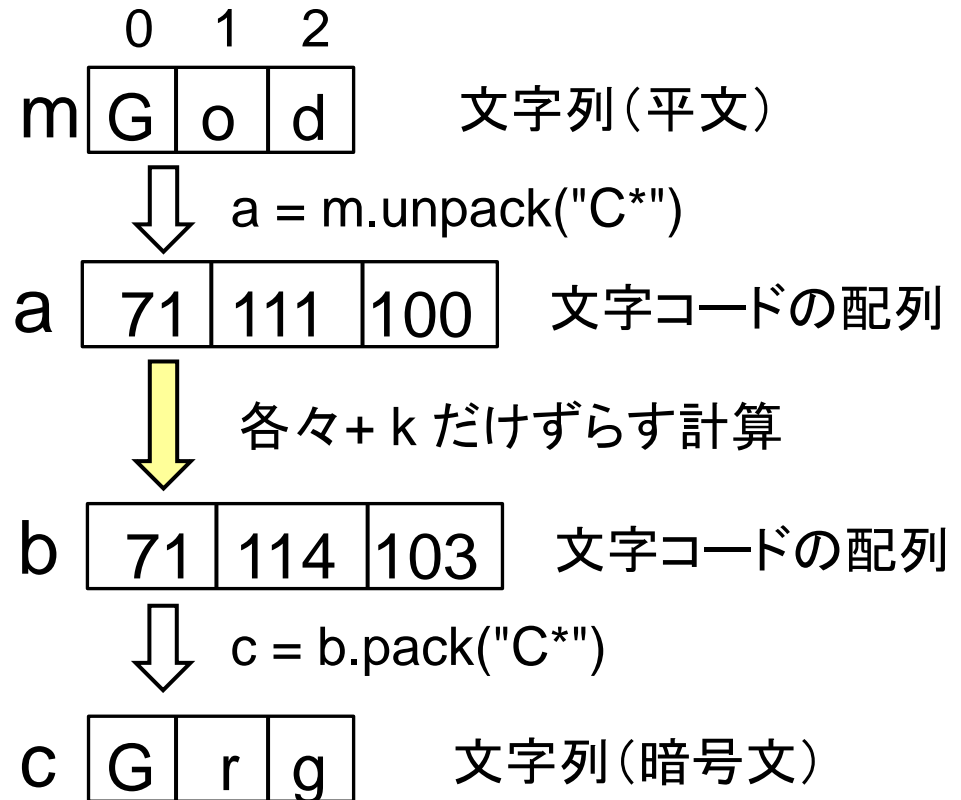
ango.rb

```
def enc(k, m)
  計算を書く
  c = ...
  return(c)
end
##### プログラム本文 #####
k = 3
hirabun = gets.chomp
angobun = enc(k, hirabun)
puts(angobun)
```

では、どう書くか？

宿題

考え方



復号化関数も
同様にプログラム化しよう

2. 関数とサブルーチン

参考

code.rb

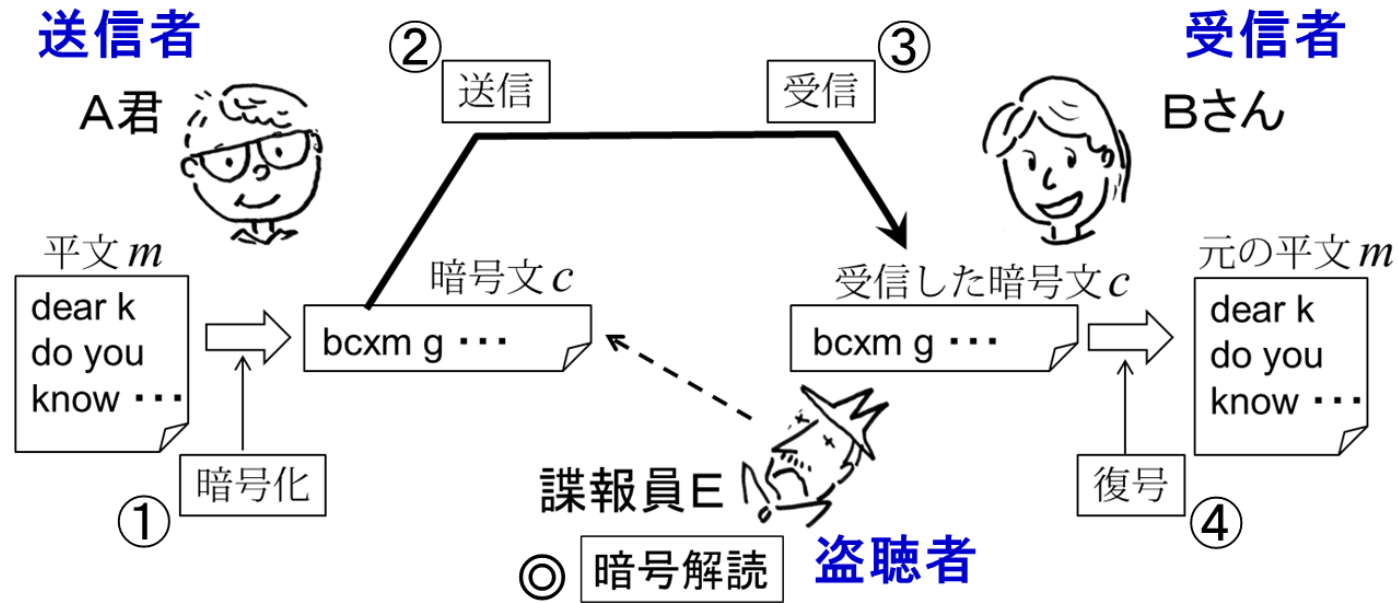
```
code_a = 97          # 文字 a の文字コード
kosu = 26            # 英字アルファベットの数

bun = gets.chomp     # 入力文字列から改行を除去
cc = bun.unpack("C*") # 文字列 → 文字コードの配列
leng = bun.length    # 文字列の長さ

for i in 0..leng-1
  moji = bun[i]       # bun の i 文字目を得る (i は 0 から始まる)
  code = cc[i]        # その文字のコードを得る
  sa = code - code_a  # 文字 a との差分
  if 0 <= sa && sa < kosu
    print(moji, ": ", code, ", ", sa, "¥n")
  else
    print(moji, ":", code, "¥n")
  end
end
end
```

3. レポート課題3(予告)

※別紙参照



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

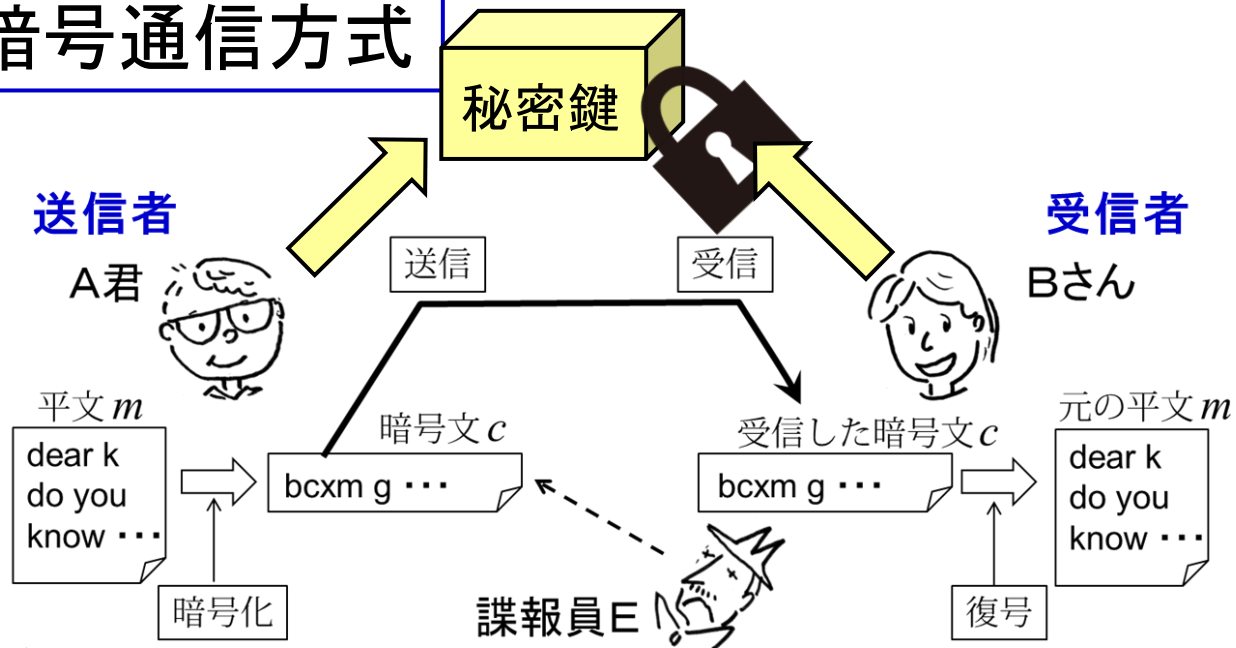
「踊る人形」
コナン・ドイル作

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

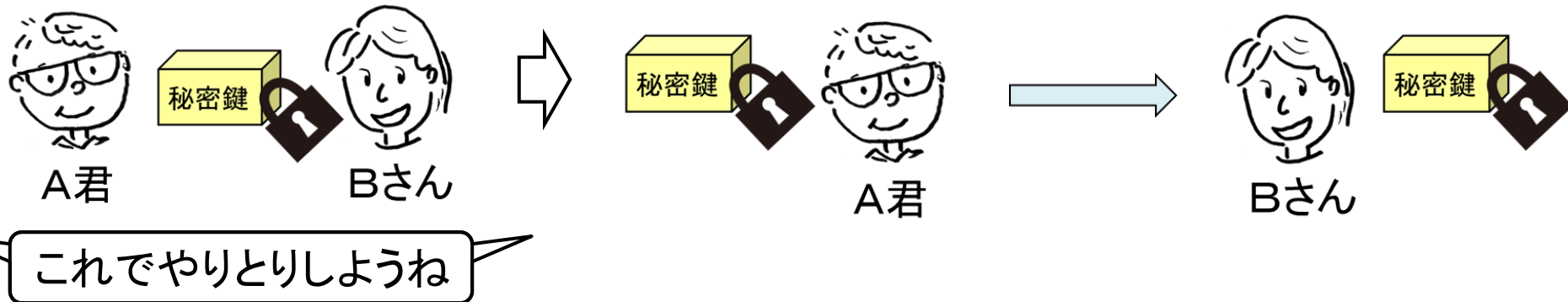
公開鍵暗号方式

公開鍵...皆に知らせてよい鍵, 暗号化に使う

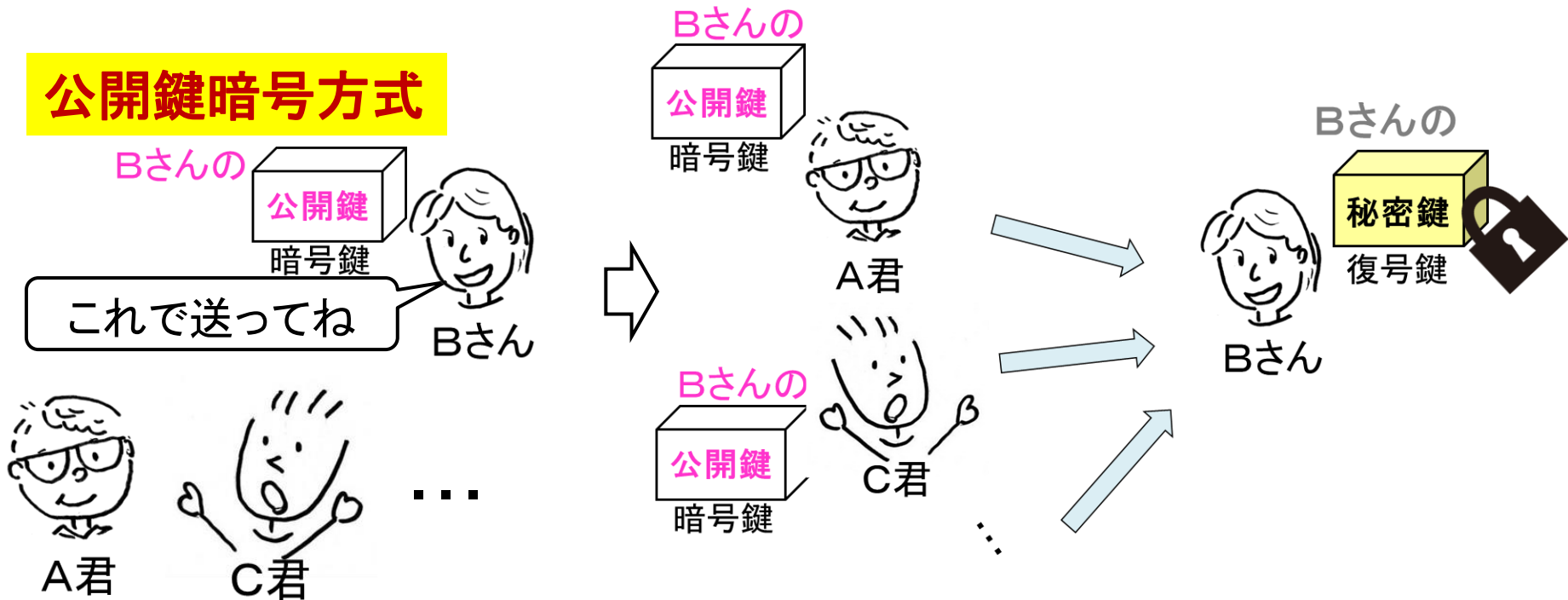
秘密鍵...復号に使う

4. 現代の暗号通信方式

秘密鍵暗号方式



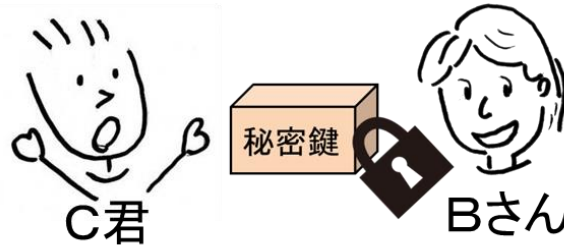
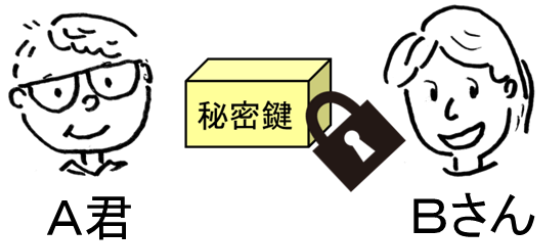
公開鍵暗号方式



4. 現代の暗号通信方式

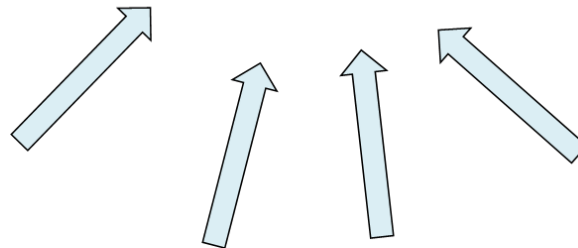
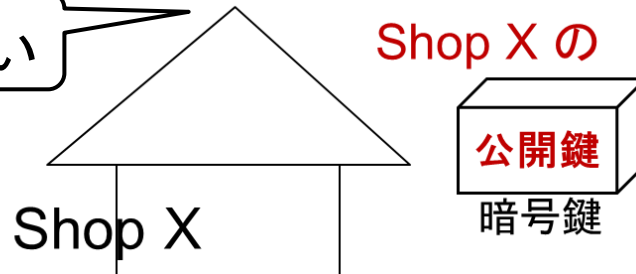
秘密鍵暗号方式

これでやりとりしようね



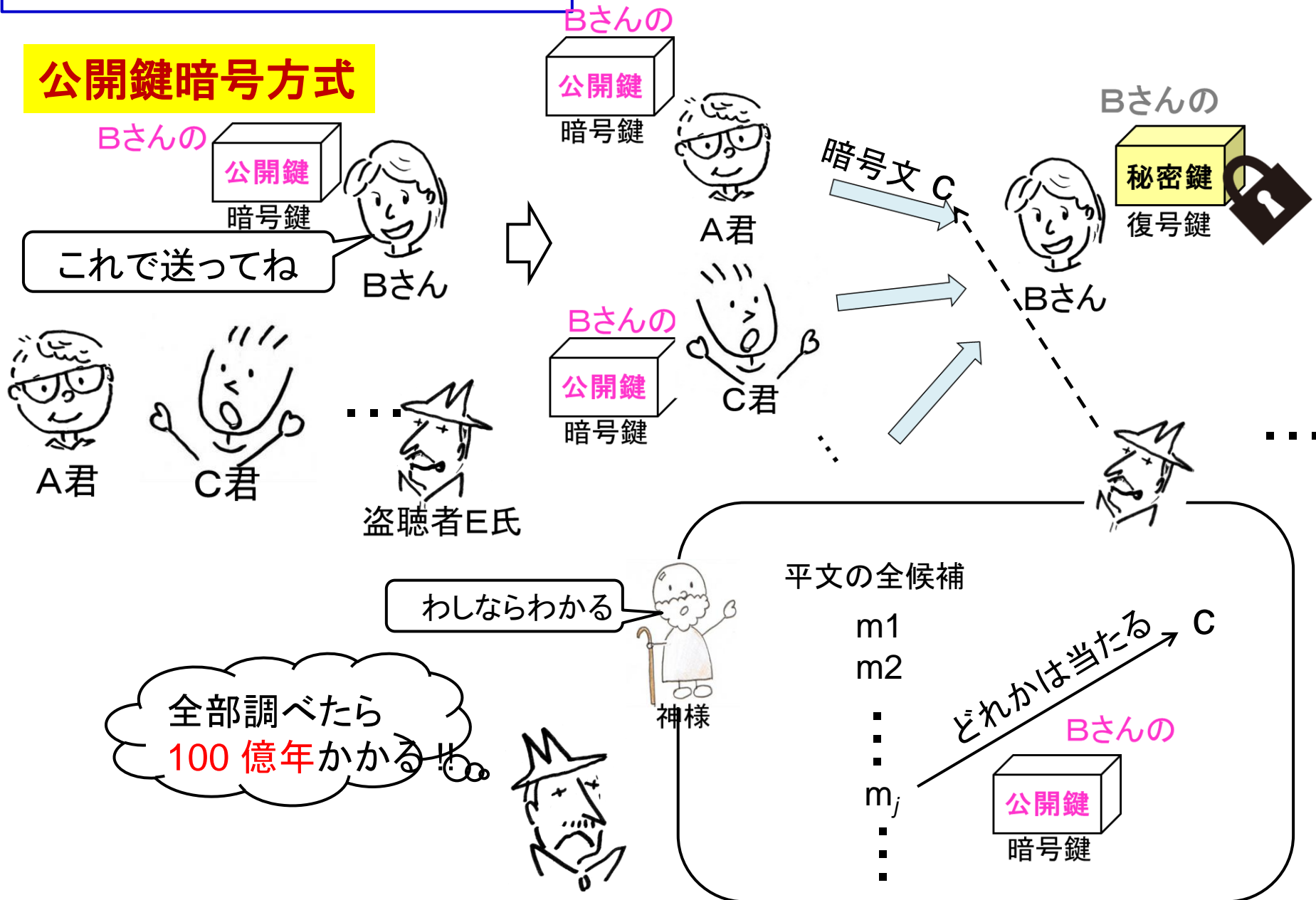
公開鍵暗号方式

うちにはこれで送って下さい



4. 現代の暗号通信方式

公開鍵暗号方式



まとめ: Ruby (5)

【サブルーチン(Ruby では関数)の定義方法】

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

← 目標の関数値を計算する
プログラム

← 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.