期末試験のお知らせ

7月30日(木) 13:30〜14:30

(60分試験)

8-208教室 (ここじゃない)

- 今日の講義内容まで
- 学生証必携
- 学部・大学院合併で行なう

レポート提出について

期日: 8月7日(金)20時頃まで

内容:

配布プリントのレポート問題の例のような内容及び授業に関連する内容で、授業内容の理解または発展的な取組みをアピールできるようなもの(詳細はプリント参照のこと)

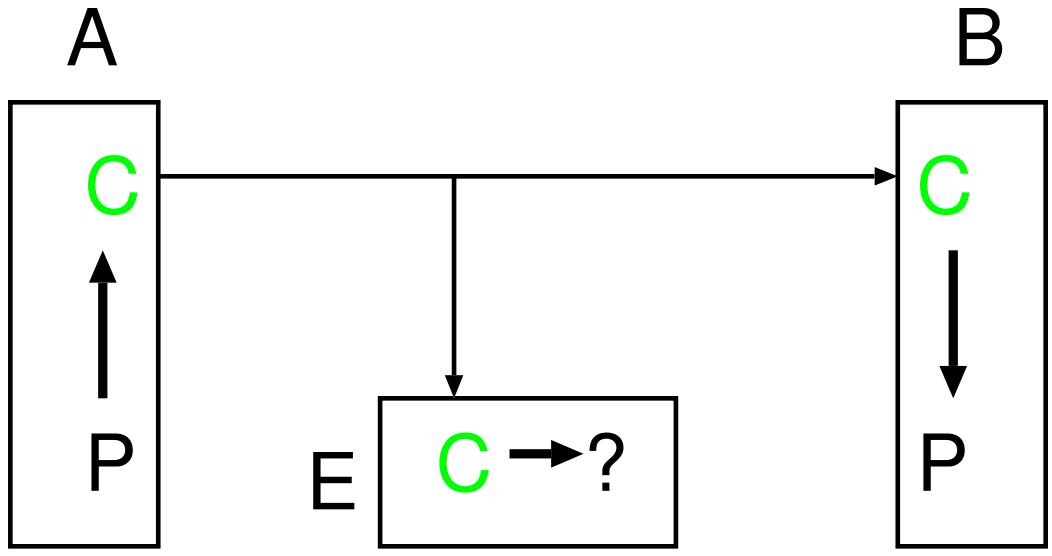
提出方法:

- **授業時に手渡し**
- **4-574 室扉のレポートポスト**
- **電子メイル**

情報通信を行なう際の要請

- **効率的に** $\longrightarrow$ **情報理論**
- **確実に** $\longrightarrow$ **符号理論**
- **安全に** $\longrightarrow$ **暗号理論**

暗号 (cryptography)



- 送信者 A が平文 P を暗号化、暗号文 C を送信
- 受信者 B が暗号文 C を受信、平文 P に復号
- 盗聴者 E は暗号文 C を知っても
  平文 P を復元できない

⟶ B だけが**復号鍵**を持っていることが必要

暗号 (cryptography)

仮定：

公開された情報伝達路 (盗聴可能 と仮定) で、

暗号方式を公開 して通信

- **秘密鍵暗号 (共通鍵暗号)**
- **公開鍵暗号**

暗号 (cryptography)

- **共通鍵暗号** (秘密鍵暗号)
  - ⋆ 送信者・受信者で同じ鍵を秘密裡に共有
  - ⋆ 共通の鍵で暗号化・復号を行なう

- **公開鍵暗号**
  - ⋆ 暗号化鍵 (公開鍵)・復号鍵 (秘密鍵) が別
  - ⋆ 公開された暗号化鍵を用いて暗号化
  - ⋆ 復号鍵は受信者だけの秘密

秘密鍵 (共通鍵) 暗号

暗号化鍵・復号鍵が同じ

- 換字暗号・Caesar 暗号
- 線型ブロック暗号
- Vernam 暗号
- DES (Data Encryption Standard)
- AES (Advances Encryption Standard)

秘密鍵 (共通鍵) 暗号の特徴

暗号化鍵・復号鍵が同じ

- **一般に原理は簡単で高速**
- **事前の鍵共有の必要**
- **通信相手毎に別の鍵が必要**

現在の情報化社会では
様々な場面で暗号が使われている

例: インターネット取引
(ネットショッピングなど)

- **不特定多数の人と暗号通信をしたい**
- **事前に鍵を共有できない**

$\longrightarrow$ 共通鍵暗号では実現が困難

$\longrightarrow$ 公開鍵暗号・鍵共有方式のアイデア
(1976〜77)

公開鍵暗号

暗号化鍵 (公開鍵)・復号鍵 (秘密鍵) が別

- **事前の鍵共有の必要無し**
  $\longrightarrow$ 見ず知らずの人からも送ってもらえる

- **認証・署名機能がある**
  $\longrightarrow$ 改竄・なり済ましの対策
  $\longrightarrow$ 否認防止の機能も持つ

公開鍵暗号

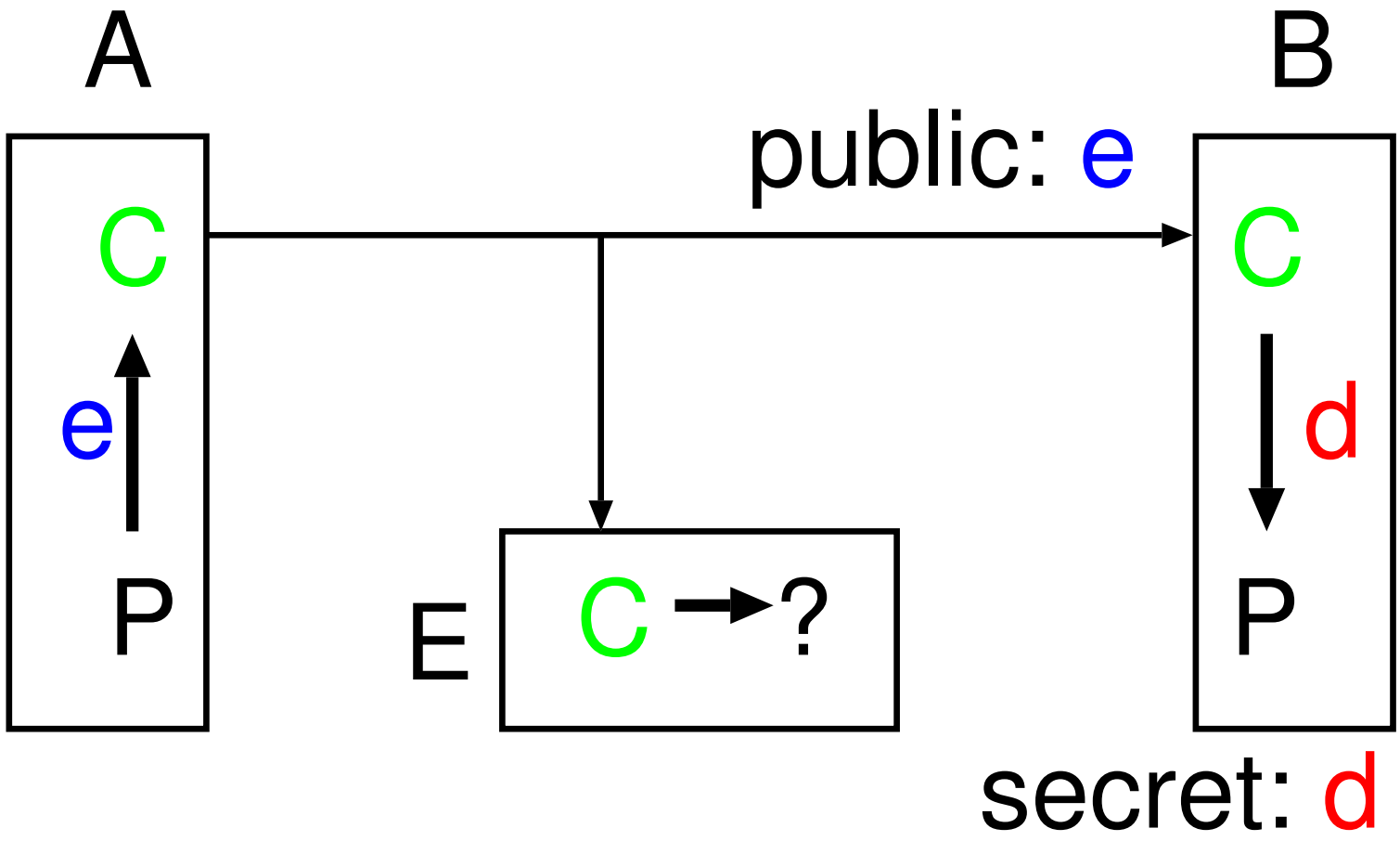
但し、一般には、
暗号化・復号が共通鍵暗号に比べて低速

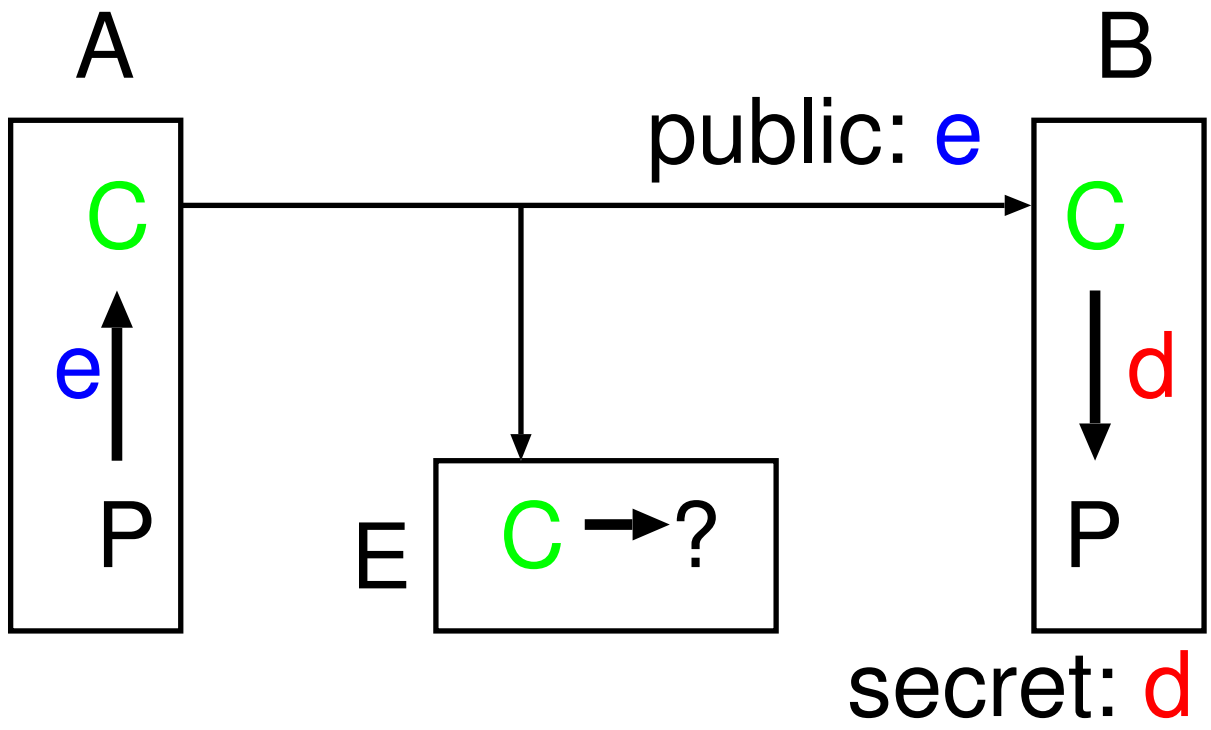
そこで、

- **始めに公開鍵暗号方式で鍵を送付・共有**
- **その鍵を用いて秘密鍵暗号方式で通信**

というように、組合わせて用いることが多い

公開鍵暗号による暗号通信



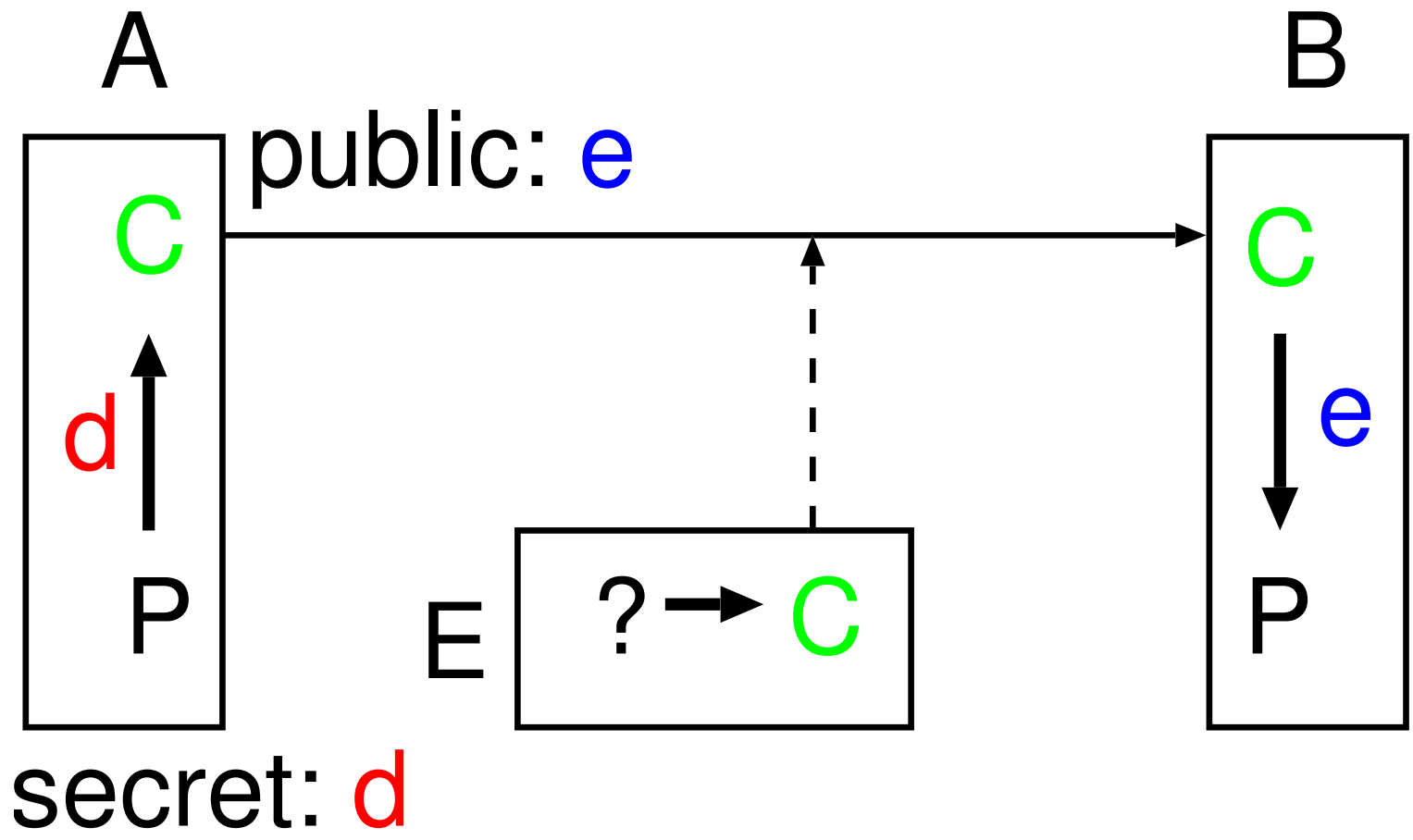


しかし、これだと誰でも暗号化できるので、

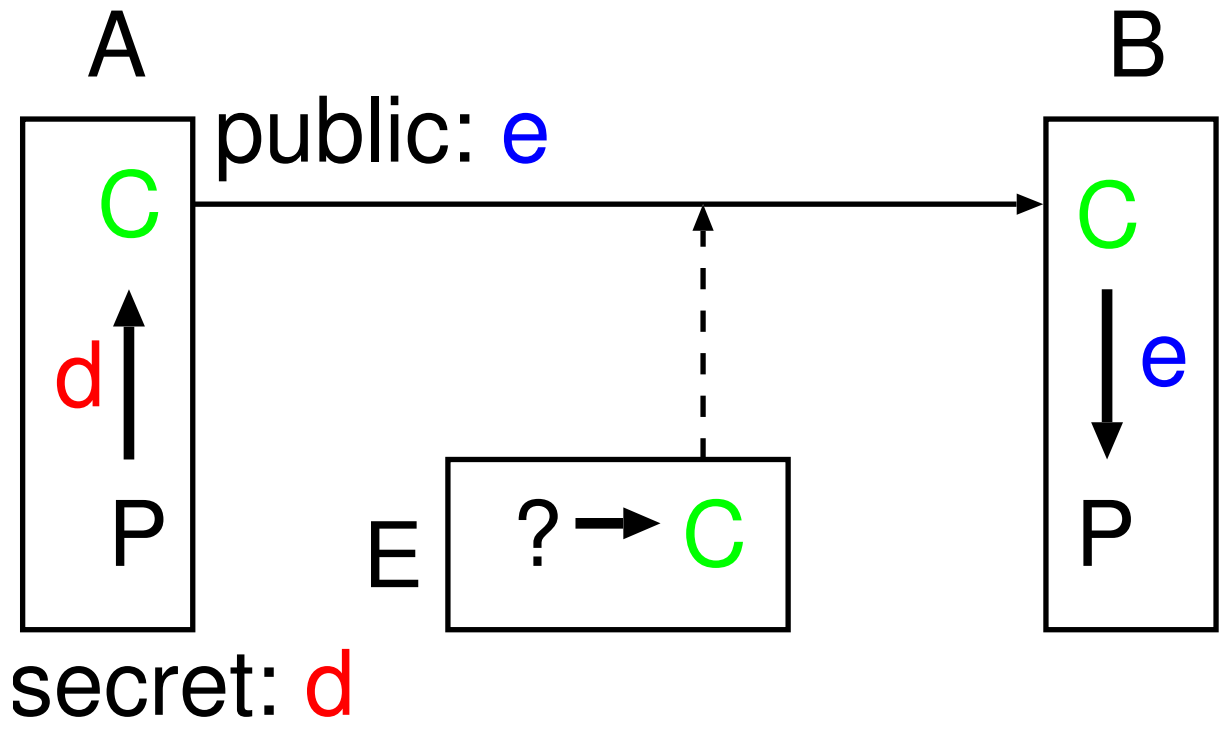
A 氏が送った保証がない

⟶ 署名の必要性

公開鍵暗号を用いた署名



公開鍵暗号を用いた署名



盗聴者 E 氏は

平文 P は判らないが、暗号文 C は盗聴可能

$\longrightarrow$ いつも同じ署名は使えない

公開鍵暗号を用いた署名

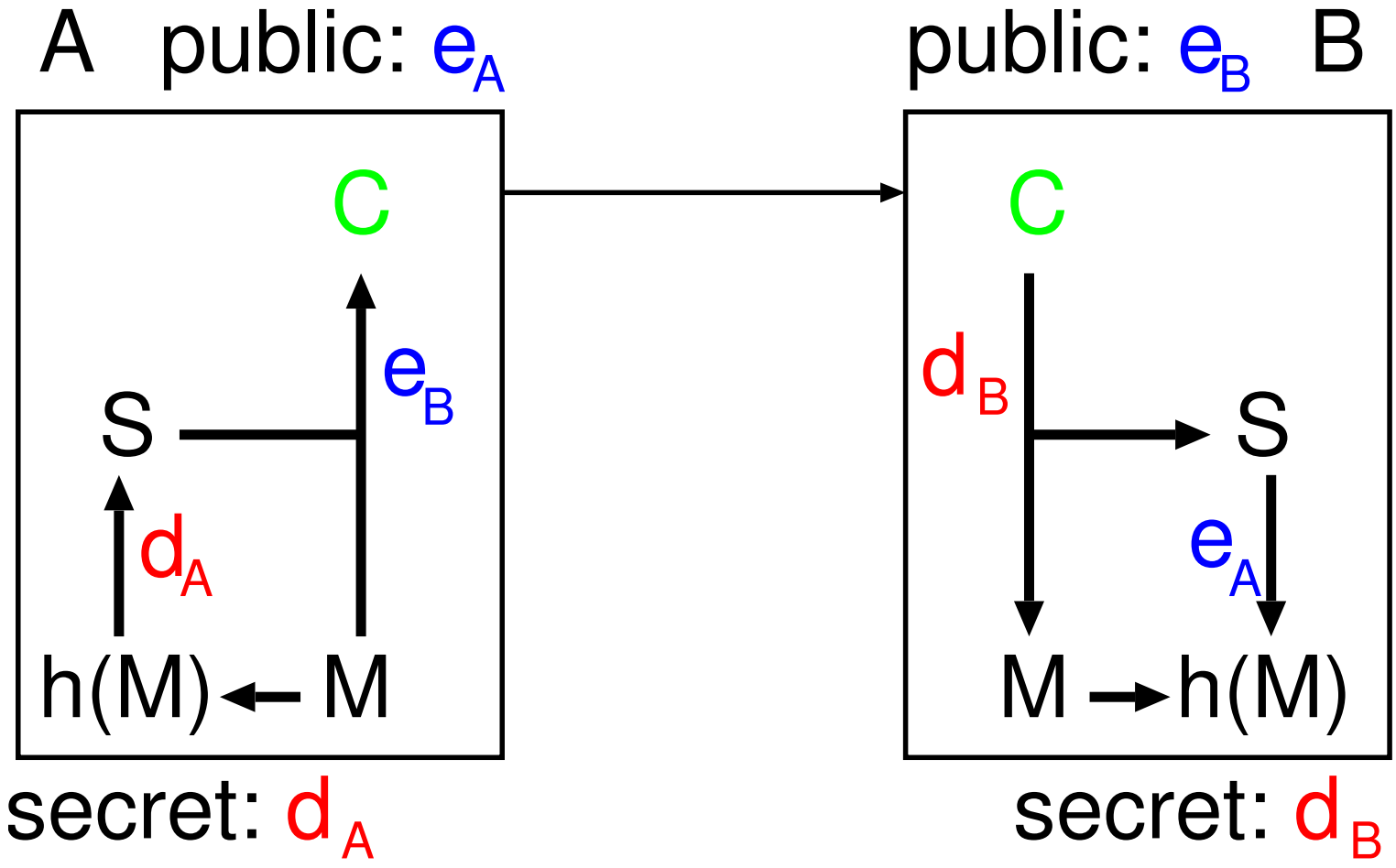
実際には、メッセージ本文 M に対して、

M から決まる短い値 (ハッシュ値) $h(M)$ を

送信者 A 氏の秘密鍵で暗号化した文字列 S を
本文 M に添付して、

受信者 B 氏の公開鍵で一緒に暗号化して送る

公開鍵暗号を用いた署名



公開鍵暗号の特徴

- 暗号化は誰でも出来る
- 復号は秘密鍵を知らないと出来ない
  (もの凄く時間が掛かる)

そんな都合の良い仕組みが本当にあるのか ？

$\longrightarrow$ ある!! (多分大丈夫)

計算困難な問題 を利用
(素因数分解・離散対数問題)

代表的な公開鍵暗号方式

- **RSA 暗号 (Rivest-Shamir-Adleman)**
- **Diffie-Hellman 鍵共有**
- **ElGamal 暗号**

公開鍵暗号の例: RSA 暗号

Rivest, Shamir, Adleman (1977)

- **大きな素数 p, q を選び、積 $n = pq$ を作る**
- **n を用いて、公開鍵 e・秘密鍵 d の対を作る**
- **暗号化の計算は n と公開鍵 e とから可能**
- **復号は秘密鍵 d を用いる**
- **n と公開鍵 e とから秘密鍵 d を求めるには、n の素因子分解 $n = pq$ が必要**
- **しかしそれは困難 (膨大な計算時間が掛かる)**

RSA 暗号 (Rivest-Shamir-Adleman)

素因数分解の困難さを利用

p, q : **相異なる大きい素数**
(実際には現在は 512 **bit 程度)**

$N := pq$: **RSA 方式の法 (modulus)**

$m := \mathrm{lcm}(p-1, q-1)$

$$(\boldsymbol{Z}/N\boldsymbol{Z})^\times \simeq (\boldsymbol{Z}/p\boldsymbol{Z})^\times \times (\boldsymbol{Z}/q\boldsymbol{Z})^\times$$
$$\simeq \boldsymbol{Z}/(p-1)\boldsymbol{Z} \times \boldsymbol{Z}/(q-1)\boldsymbol{Z}$$

: 指数 m **の有限アーベル群**

RSA 暗号 (Rivest-Shamir-Adleman)

p, q：**相異なる大きい素数**

$N = pq,\ \ m = \mathrm{lcm}(p-1, q-1)$

e **を** $\gcd(e, m) = 1$ **となるように選ぶ**

d **を** $ed \equiv 1 \pmod{m}$ **となるように求める**

- (N, e)：**公開鍵 (暗号化鍵)**
- d：**秘密鍵 (復号鍵)**

RSA 暗号 (Rivest-Shamir-Adleman)

p, q：**相異なる大きい素数**

$$N = pq, m = \mathrm{lcm}(p-1, q-1)$$

$$ed \equiv 1 \pmod{m}$$

- (N, e)：**公開鍵 (暗号化鍵)**
- d：**秘密鍵 (復号鍵)**

平文 $M \bmod N$ **の暗号化**：$\boxed{C = M^e \bmod N}$

暗号文 $C \bmod N$ **の復号**：$\boxed{M = C^d \bmod N}$

RSA 暗号 (Rivest-Shamir-Adleman)

公開鍵 (N, e) から秘密鍵 d が計算できるか ？

- **N の素因数分解 $N = pq$ を知っていれば容易**
- **事実上 N の素因数分解と同程度の困難さ**

「困難さ」 $\cdots$ 計算時間が掛かる

RSA 暗号の安全性 $\Longleftrightarrow$ 素因数分解の困難さ

“計算量的安全性”

計算量 (complexity)

- **時間計算量**: 計算に掛かるステップ数
- **空間計算量**: 計算に必要なメモリ量

通常は、決まった桁数の四則演算 1 回を
1 ステップと数えることが多い

入力データの大きさ (bit 長) n に対する
計算の回数の増加の オーダー で表す
(定数倍の違いは気にしない)

通常 Landau の O-記号を用いて表す

計算量 (complexity)

- **時間計算量**: 計算に掛かるステップ数
- **空間計算量**: 計算に必要なメモリ量

通常は、決まった桁数の四則演算 1 回を
1 ステップと数えることが多い

入力データの大きさ (bit 長) n に対する
計算の回数の増加の オーダー で表す
(定数倍の違いは気にしない)

通常 **Landau の O-記号**を用いて表す

Landau の O-記号・o-記号

$f, g : \boldsymbol{N} \longrightarrow \boldsymbol{R}_{>0}$ に対し、

$$f = O(g) \overset{\leftarrow}{\Longleftrightarrow} \exists N > 0, \exists C > 0 : \forall n : \\ (n \geq N \Longrightarrow f(n) \leq Cg(n))$$

$$f = o(g) \overset{\leftarrow}{\Longleftrightarrow} \frac{f(n)}{g(n)} \longrightarrow 0 \ (n \to \infty) \\ \Longleftrightarrow \forall \varepsilon > 0 : \exists N > 0 : \forall n : \\ (n \geq N \Longrightarrow f(n) \leq \varepsilon g(n))$$

計算量 (complexity)

問題を解くアルゴリズムによって決まる

$\cdots$ アルゴリズムの計算量

$\longrightarrow$ アルゴリズムの効率 の評価

問題の計算量:
その問題を解くアルゴリズムの計算量の下限

最も効率良く解くと、どれ位で解けるか
$=$ どうしてもどれ位必要か
$=$ どれ位難しい問題か

$\longrightarrow$ 問題の難しさ の評価

計算量 (complexity)

問題を解くアルゴリズムによって決まる

$\cdots$ アルゴリズムの計算量

$\longrightarrow$ アルゴリズムの効率 の評価

問題の計算量:

その問題を解くアルゴリズムの計算量の下限

最も効率良く解くと、どれ位で解けるか

$=$ どうしてもどれ位必要か

$=$ どれ位難しい問題か

$\longrightarrow$ 問題の難しさ の評価

計算量 (complexity)

問題を解くアルゴリズムによって決まる

$\cdots$ アルゴリズムの計算量

$\longrightarrow$ アルゴリズムの効率 の評価

問題の計算量:

その問題を解くアルゴリズムの計算量の下限

最も効率良く解くと、どれ位で解けるか

$=$ どうしてもどれ位必要か

$=$ どれ位難しい問題か

$\longrightarrow$ 問題の難しさ の評価

計算量 (complexity)

問題を解くアルゴリズムによって決まる

$\cdots$ **アルゴリズムの計算量**

$\longrightarrow$ **アルゴリズムの効率 の評価**

問題の計算量:

その問題を解くアルゴリズムの計算量の下限

最も効率良く解くと、どれ位で解けるか

$=$ **どうしてもどれ位必要か**

$=$ **どれ位難しい問題か**

$\longrightarrow$ **問題の難しさ の評価**

例:

- **加法:** $O(n)$
- **乗法:** $O(n^2)$
  かと思いきや $O(n \log n \log \log n)$
  (高速フーリエ変換 (FFT))

例:

- **加法:** $O(n)$

- **乗法:** $O(n^2)$
  かと思いきや $O(n \log n \log \log n)$
  (高速フーリエ変換 (FFT))

例: 互除法

- **入力: 正整数** x, y
  入力データ長:
  $n = \lceil \log_2 x \rceil + \lceil \log_2 y \rceil \sim \max\{\log x, \log y\}$
- **出力: 最大公約数** $d = \gcd(x, y)$

計算量の評価:

- **割算の回数:** $O(n)$
- 1 **回の割算: 素朴な方法でも** $O(n^2)$
  (FFT を使えば $O(n \log n \log\log n)$**)**

$\longrightarrow$ **併せて** $O(n^3)$ **(FFT で** $O(n^2 \log n \log\log n)$**)**
$\cdots$ **充分に高速なアルゴリズム**

例: 互除法

- **入力: 正整数** x, y
  入力データ長:
  $n = \lceil \log_2 x \rceil + \lceil \log_2 y \rceil \sim \max\{\log x, \log y\}$
- **出力: 最大公約数** $d = \gcd(x, y)$

計算量の評価:

- **割算の回数:** $O(n)$
- 1 **回の割算: 素朴な方法でも** $O(n^2)$
  (FFT を使えば $O(n \log n \log\log n)$**)**

$\longrightarrow$ **併せて** $O(n^3)$ **(FFT で** $O(n^2 \log n \log\log n)$**)**
$\cdots$ **充分に高速なアルゴリズム**

重要な難しさのクラス

多項式時間 P $\cdots \exists k : O(n^k)$

“事実上計算可能な難しさ”

「しらみつぶし」が入ると
大体 $O(2^n)$ 程度以上になる (指数時間 EXP)
“事実上計算不可能”

重要な難しさのクラス

多項式時間 P $\cdots \exists k : O(n^k)$

“事実上計算可能な難しさ”

「しらみつぶし」が入ると
大体 $O(2^n)$ 程度以上になる (指数時間 EXP)
“事実上計算不可能”

例: 素数判定 (PRIMES)

$n = \log_2 N$: N の二進桁数

試行除算 (小さい方から割っていく) だと
$O(n^k 2^{n/2})$ くらい掛かりそう

実は多項式時間で解ける!!

Agrawal-Kayal-Saxena
"PRIMES is in P" (2002)
(出版は
Ann. of Math. 160(2) (2004),781-793.)

例: 素数判定 (PRIMES)

$n = \log_2 N$: N の二進桁数

試行除算 (小さい方から割っていく) だと
$O(n^k 2^{n/2})$ くらい掛かりそう

実は多項式時間で解ける!!

Agrawal-Kayal-Saxena
"PRIMES is in P" (2002)
(出版は
Ann. of Math. 160(2) (2004),781-793.)

このような効率の良い素数判定は、
具体的に素因数を見付けている訳ではない

素因数分解は P であるかどうか未解決
(多項式時間アルゴリズムが知られていない)

現状で知られているのは、
“準指数時間” $L_N[u,v]$ $(0<u<1)$
のアルゴリズム
(現時点で最高速なのは $u=1/3$)

このような効率の良い素数判定は、
具体的に素因数を見付けている訳ではない

素因数分解は P であるかどうか未解決
(多項式時間アルゴリズムが知られていない)

現状で知られているのは、
“準指数時間” $L_N[u, v]$ **(**$0 < u < 1$**)**
のアルゴリズム
(現時点で最高速なのは $u = 1/3$**)**

素因数分解アルゴリズム等の計算量を表すのに

$$L_N[u,v] := \exp(v(\log N)^u(\log\log N)^{1-u})$$

が良く用いられる

$n = \log N$ **(**N **の桁数) とおくと、**

- $L_N[0,v] = e^{v\log\log N} = n^v$ **：多項式時間**
- $L_N[1,v] = e^{v\log N} = e^{vn}$ **：指数時間**

代表的な素因数分解法

- $(p-1)$-法
- 楕円曲線法 (Elliptic Curve Method)
- 二次篩法 (Quadratic Sieve)
- 数体篩法 (Number Field Sieve)

素因数分解問題の高速なアルゴリズムの発見

$\Downarrow$

RSA 暗号の解読

しかし、逆は真とは限らない
(解読には色々な方法があり得る)

「何で負けても負けは負け」

素因数分解問題の高速なアルゴリズムの発見

$\Downarrow$

RSA 暗号の解読

しかし、逆は真とは限らない
(解読には色々な方法があり得る)

「何で負けても負けは負け」

離散対数問題 (Discrete Logarithm Problem)

p：素数

$G := (\boldsymbol{Z}/p\boldsymbol{Z})^{\times}$：位数 $p-1$ の巡回群

$g \bmod p \in G$：$\bmod\ p$ の原始根 ($G = \langle g \rangle$)

とするとき、

$x \bmod p \in G$ に対し、

$$\boxed{g^a \equiv x \pmod{p}}$$

となる a を求めよ。

Diffie-Hellman 鍵共有

離散対数問題の困難さを利用して、

公開通信路で秘密裡に鍵共有を行なう方式

p：**素数**

$G = (\boldsymbol{Z}/p\boldsymbol{Z})^{\times} = \langle g \rangle$：**位数** $p-1$ **の巡回群**

A・B 両氏がそれぞれ

ランダム (秘密) に a, b **を選ぶ**

Diffie-Hellman 鍵共有

離散対数問題の困難さを利用して、

公開通信路で秘密裡に鍵共有を行なう方式

p：**素数**

$G = (\boldsymbol{Z}/p\boldsymbol{Z})^{\times} = \langle g \rangle$：**位数** $p-1$ **の巡回群**

A・B 両氏がそれぞれ

ランダム (秘密) に a, b **を選ぶ**

Diffie-Hellman 鍵共有

離散対数問題の困難さを利用して、

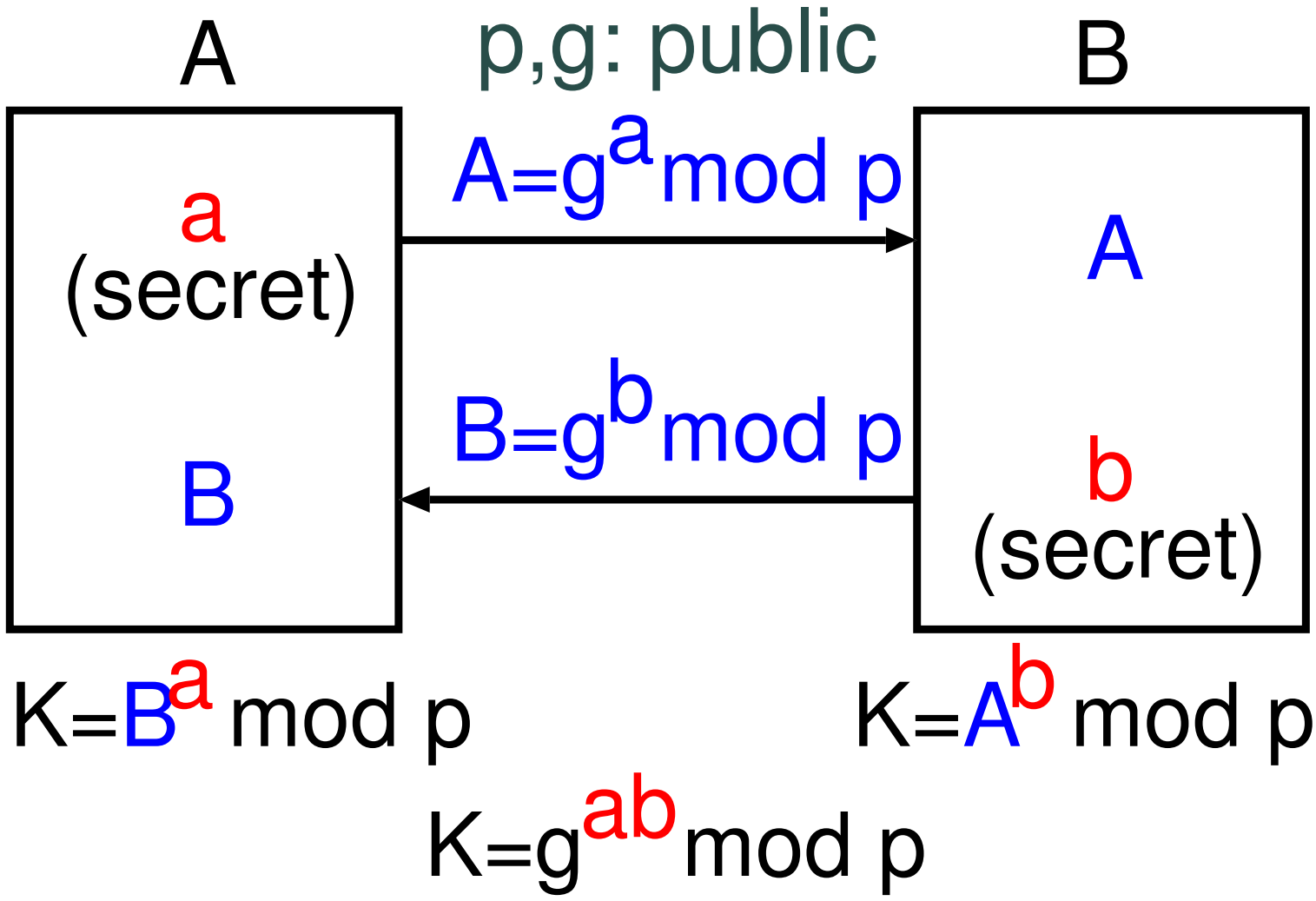
公開通信路で秘密裡に鍵共有を行なう方式

p **: 素数**

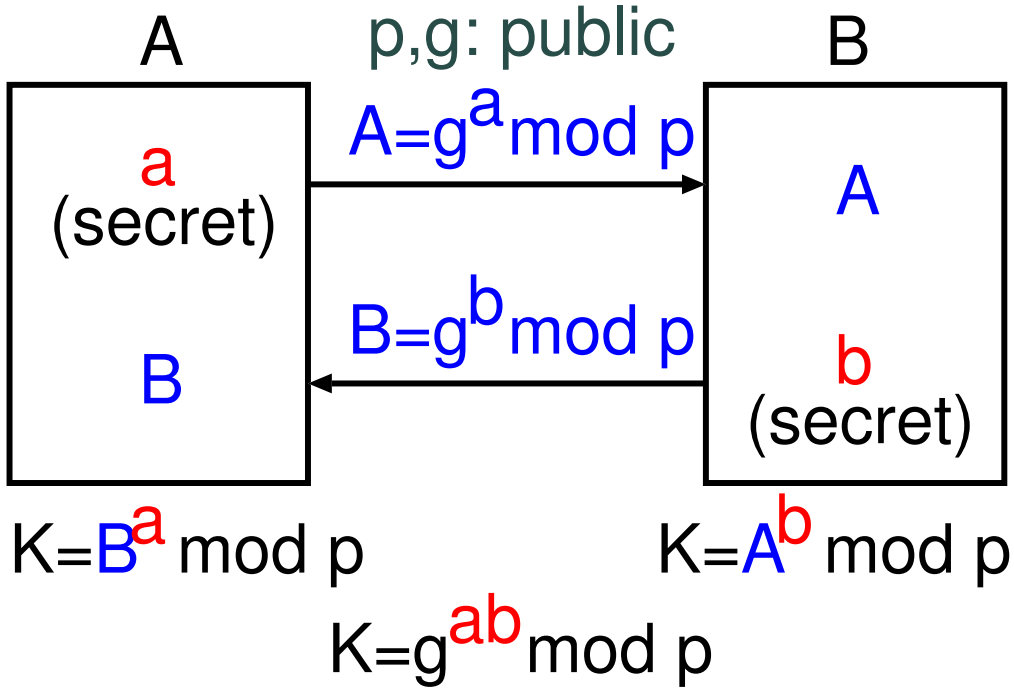
$G = (Z/pZ)^{\times} = \langle g \rangle$ **: 位数** $p-1$ **の巡回群**

A・B 両氏がそれぞれ
ランダム (秘密) に a, b **を選ぶ**

Diffie-Hellman 鍵共有



Diffie-Hellman 鍵共有



(p, g, A, B) **が判っても** a, b **が判らない (DLP)**

$\longrightarrow$ **秘密鍵** K **の共有が可能 !!**

ElGamal 暗号

離散対数問題を利用し、乱数を用いた暗号方式

p：**素数**

$G = (\boldsymbol{Z}/p\boldsymbol{Z})^{\times} = \langle g \rangle$：**位数** $p-1$ **の巡回群**

受信者 B 氏が ランダム (秘密) に b **を選び、**

$B := g^b \bmod p$ **を公開**

送信者 A 氏が ランダム (秘密) に a **を選び、**

$A := g^a \bmod p,\ C := B^a M \bmod p$ **を送信**

ElGamal 暗号

離散対数問題を利用し、乱数を用いた暗号方式

p**：素数**

$G = (\boldsymbol{Z}/p\boldsymbol{Z})^{\times} = \langle g \rangle$**：位数** $p-1$ **の巡回群**

受信者 B 氏が ランダム (秘密) に b **を選び、**

$B := g^b \bmod p$ **を公開**

送信者 A 氏が ランダム (秘密) に a **を選び、**

$A := g^a \bmod p,\ C := B^a M \bmod p$ **を送信**

ElGamal 暗号

受信者 B 氏が ランダム (秘密) に b **を選び、**

$$B := g^b \bmod p \text{ を公開}$$

送信者 A 氏が ランダム (秘密) に a **を選び、**

$$A := g^a \bmod p,\ C := B^a M \bmod p \text{ を送信}$$

受信者 B 氏は、 $M = (A^b)^{-1} C \bmod p$ **を計算**

ElGamal 暗号

平文 M $\longrightarrow$ **暗号文** (A, C)

- **送信データ長が** 2 **倍 (メッセージ膨張)**
- **乱数により、同じ文書が毎回異なる暗号化**

離散対数問題を利用した方式は
他の有限アーベル群でも可能

- 有限体上の楕円曲線の有理点の群
  (楕円曲線暗号)
- 有限体上の超楕円曲線の
  Jacobian の有理点の群
  (超楕円曲線暗号)
- 代数体の ideal 類群

疑似乱数 (psuedo random number)

充分ランダムに 見える 長い周期の数列を
発生させるアルゴリズム

“Mersenne Twister” (松本-西村、松本-斎藤)
· · · 現在、事実上最強のアルゴリズム

- **MT19937:**
  - ⋆ **極めて長周期 (周期 $2^{19937}-1$)**
  - ⋆ **極めてランダム (623 次元均等分布)**
  - ⋆ **極めて高速**
- **本来は Monte-Carlo simulation 用**
- **暗号には適切なハッシュ関数と組合わせる**

疑似乱数 (psuedo random number)

充分ランダムに 見える 長い周期の数列を
発生させるアルゴリズム

"Mersenne Twister" (松本-西村、松本-斎藤)
$\cdots$ 現在、事実上最強のアルゴリズム

- **MT19937:**
  - ⋆ **極めて長周期 (周期 $2^{19937}-1$)**
  - ⋆ **極めてランダム (623 次元均等分布)**
  - ⋆ **極めて高速**
- **本来は Monte-Carlo simulation 用**
- **暗号には適切なハッシュ関数と組合わせる**

おしまい