HITACHI
Inspire the Next

RSAのビット長2倍化技術

～ CRTなしで 小さな法の剰余を用いて 大きな法の剰余を計算できるか？ ～

桶屋　勝幸
㈱ 日立製作所　システム開発研究所

自己紹介

桶屋とは、

・企業の研究者

・暗号実装チームのチームリーダ

・ICカードなどに暗号アルゴリズムを実装

・暗号アルゴリズムに対して、実装機器の特性を考慮して最適化・安全解析

・40数本の査読付き論文の(共)著者

・学生時代は整数論を専攻してました

中国人剰余定理

問題：
法が9以下の剰余乗算(xy mod N)器のみを用いて、
35を法とする剰余乗算を実現せよ

答え：

35＝5×7

CRT(中国人剰余定理)を用いる。

類似問題？

問題：
法が9以下の剰余乗算器のみを用いて、
37を法とする剰余乗算を実現せよ

答え：

ビット長2倍化技術を用いる ← 後で説明します

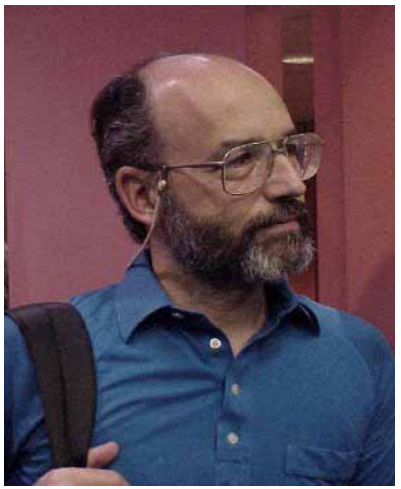
目次

RSA

Cited from http://theory.lcs.mit.edu/~rivest/

Ronald Rivest

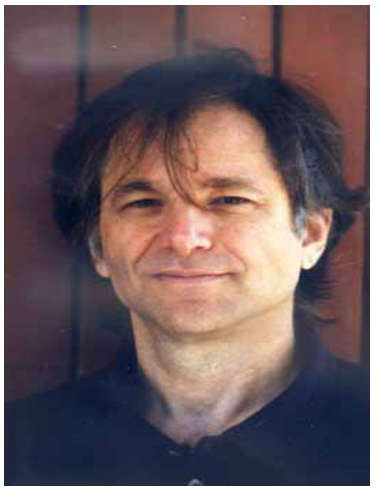
R



Cited from http://www.fermat.de/vergang/ s1/rsa.html

Adi Shamir

S

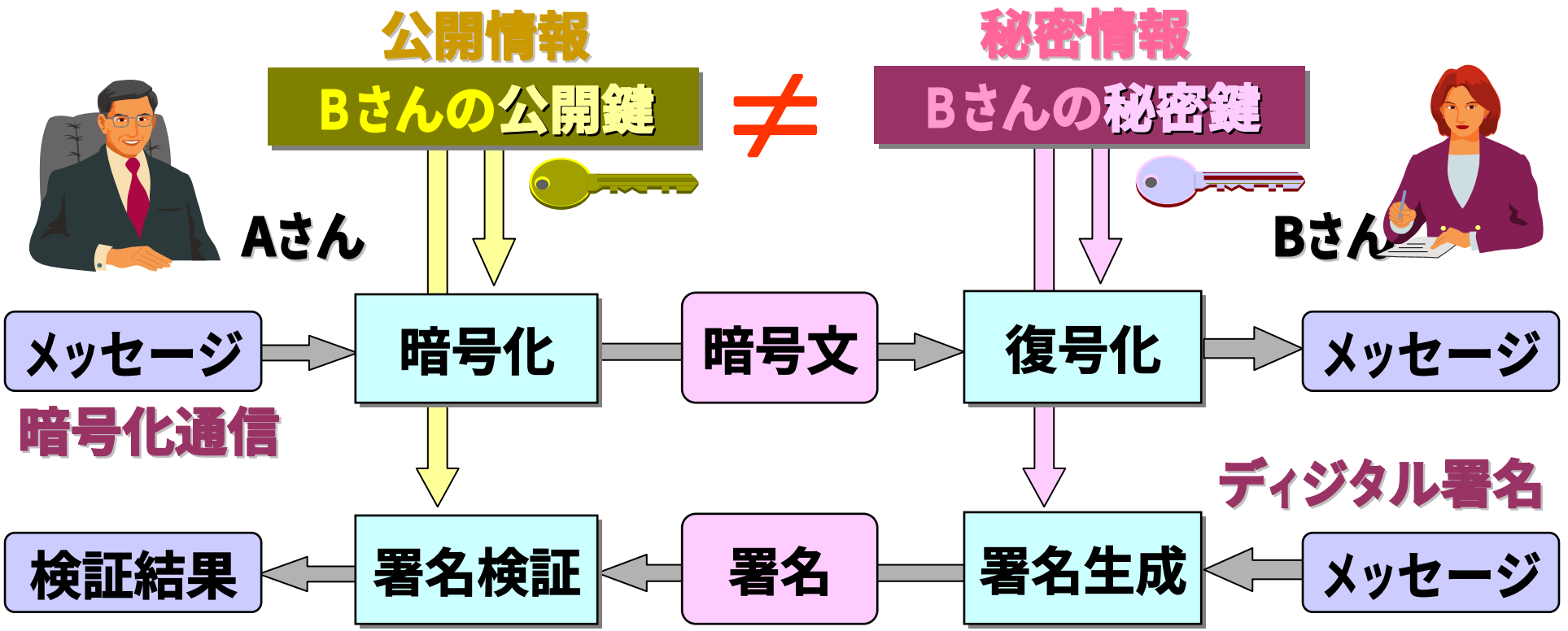


Cited from http://www-hto.usc.edu/People/adleman_intro.html

Leonard Adleman

A

公開鍵暗号



ディジタル署名は個人・機器認証等に用いられる

RSA暗号

de facto standard of public-key cryptosystems

p, q: primes, $N = pq$, $ed = 1 \bmod (p-1)(q-1)$,

e, N: public key, d: secret key,
M: message, $M \in \{0,1,2,\ldots,N-1\}$.

Encryption: $C = M^e \bmod N$

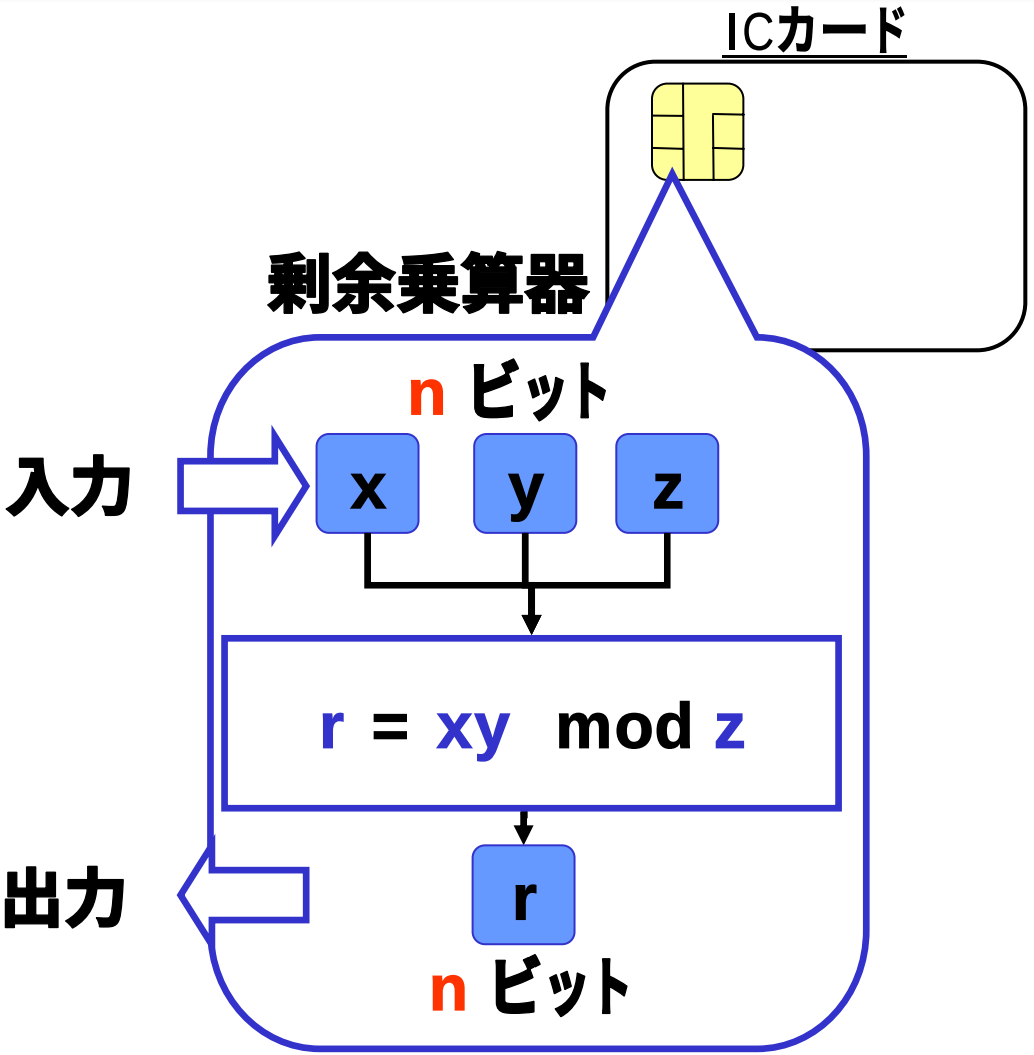
e: small ($2^{16}+1$)

Decryption: $M = C^d \bmod N$

d: large ($d > N^{1/2}$)

剰余乗算器

- 剰余乗算
  - 入力：nビット x, y, 法z
  - 出力：r = xy mod z
- ビット長
  - 1024ビットが一般的
  - 2048ビット対応のものもある
    （ただし高価）
- ビット長は固定
  - いったん作ると変えられない

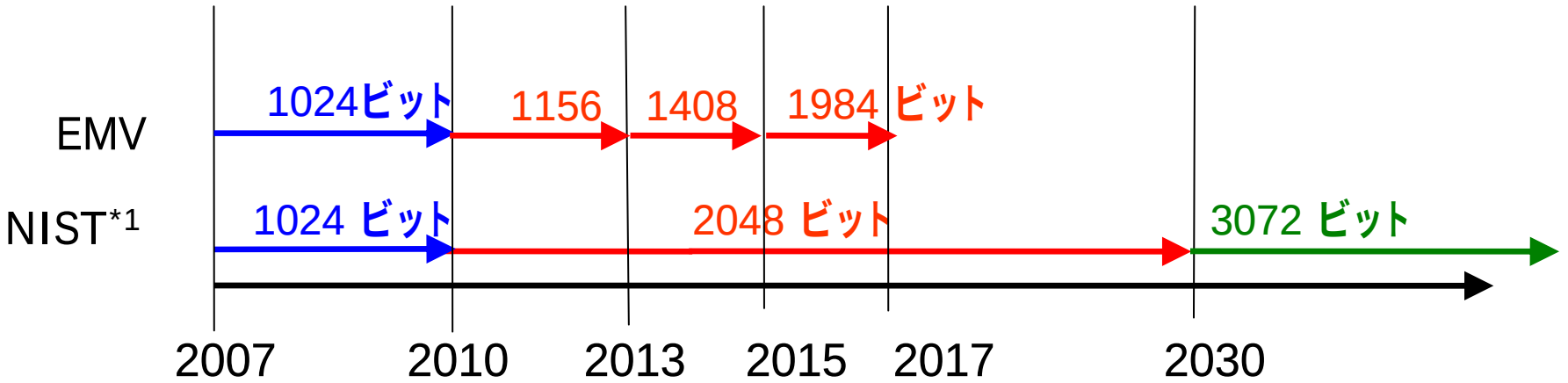


推奨ビット長

- RSA暗号
  - 素因数分解技術の進歩
  - 最低1024ビット
- 暗号技術の2010年問題

ビット長	解読された年
330	1991
512	1999
663	2005

出典：RSA challenge



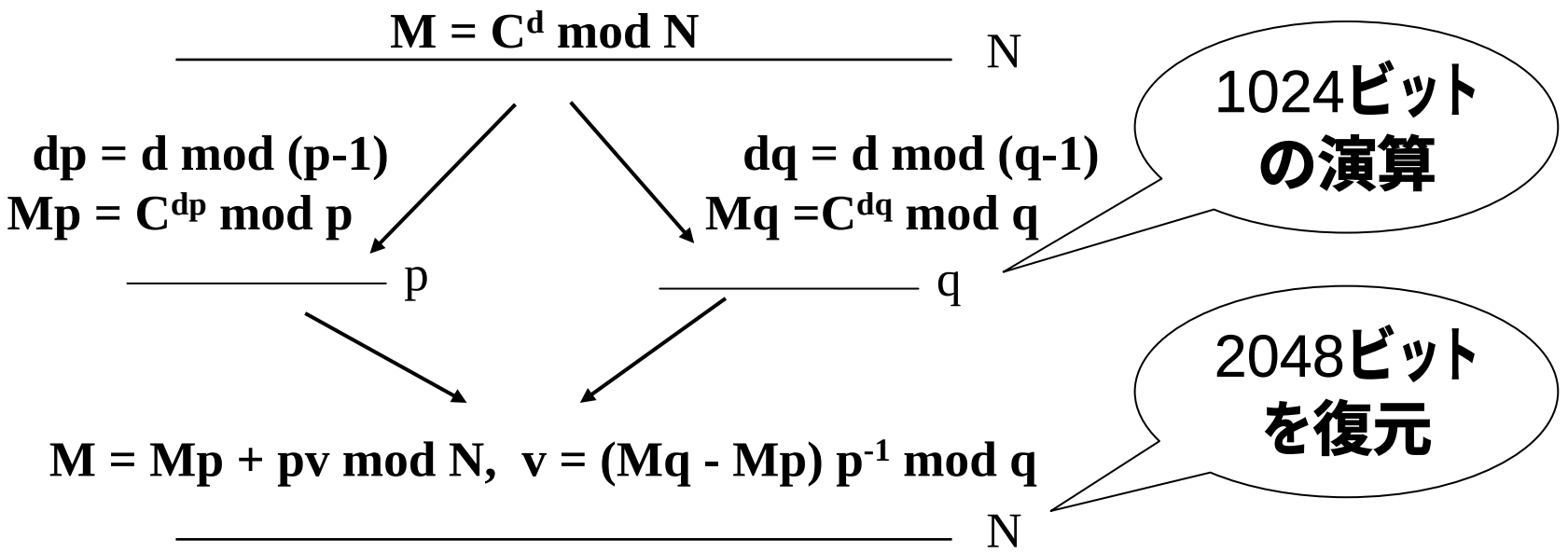
1024ビット剰余乗算器では対応できなくなる？

出典： *1・・・sp800-78

2048ビットRSA：復号へのCRT適用

Decryption: $M = C^d \bmod N(=pq)$

N:2048ビット
p,q: 1024ビット



Nの素因子p,qの値を知っている必要がある

2048ビットRSA：暗号化の場合

$$\textbf{Encryption: } C = M^e \bmod N$$

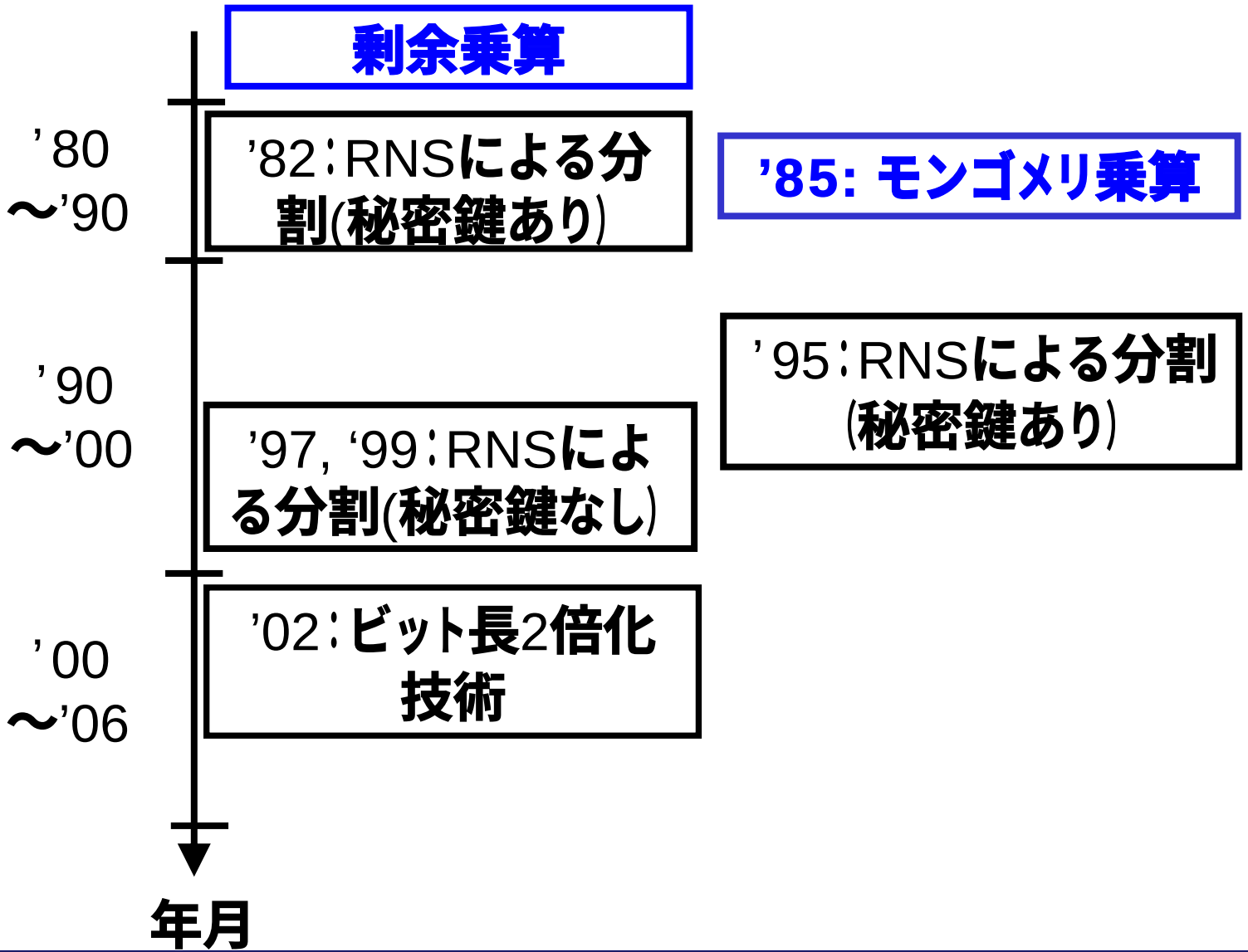
N:2048ビット
p,q: 1024ビット

N=pqだけれども、p,qは秘密鍵
→ 暗号化の場合にp,qは利用できない
→ CRTを用いることはできない
→ 困った！

ご安心を。ビット長2倍化技術があります！

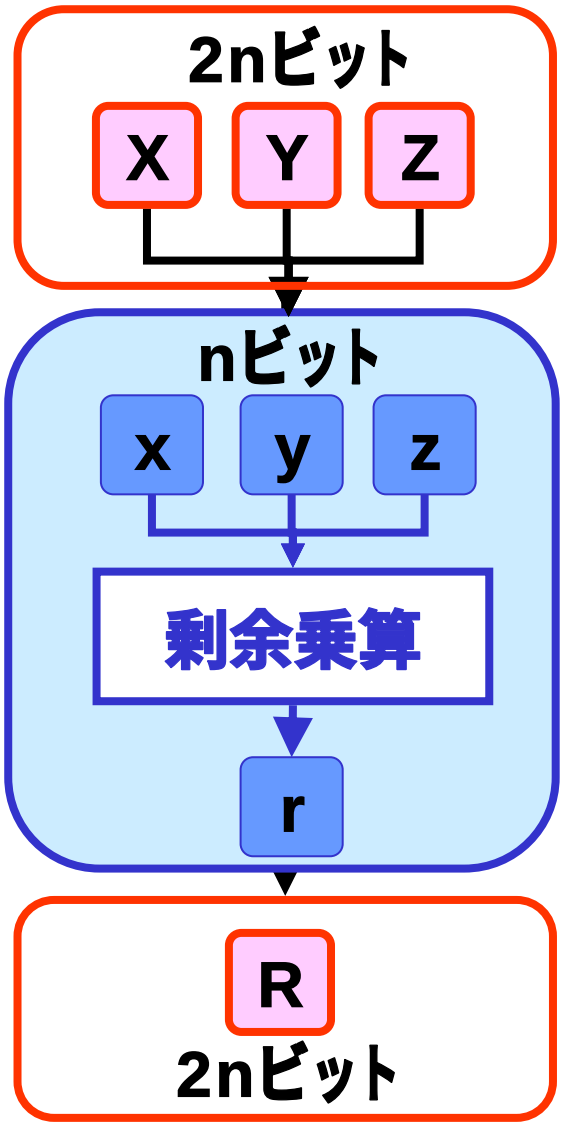
目次

関連研究


剰余乗算
’80
〜’90
’82：RNSによる分割(秘密鍵あり)
’85: モンゴメリ乗算
’90
〜’00
’95：RNSによる分割(秘密鍵あり)
’97, ‘99：RNSによる分割(秘密鍵なし)
’00
〜’06
’02：ビット長2倍化技術
年月

ビット長2倍化技術

- 2nビット整数の剰余乗算
  - R = XY mod Z
    - R, X, Y, Z：2nビット整数
  - n ビット剰余乗算器の利用可
    - r, x, y, z ： nビット整数



キーポイント：
$xy = qz + r$ **となる商**q**を復元できれば、通常の筆算による計算が可能となる**

商の復元

入力: x, y, **法**z, **ただし、** 0<= x,y < z
出力: **商**q
ただしn**ビット剰余乗算器は用いてよい**

$r_1 := xy \bmod z$
$r_2 := xy \bmod (z+1)$

$$q := \begin{cases} r_1 - r_2 & \text{if } r_1 >= r_2 \\ r_1 - r_2 + z + 1 & \text{otherwise} \end{cases}$$

証明 $xy = q(z+1)+(r_1-q)$

MulModDiv演算

MulModDiv
入力：x, y, **法**z, **ただし、**0<= x,y < z
出力：**商**q, **剰余**r
(q,r) = MulModDiv(x,y,z)

nビット剰余乗算を2回用いて達成可能

ビット長2倍化

入力: $Z = z_1 2^n + z_0,\ X = x_1 2^n + x_0,\ Y = y_1 2^n + y_0$
出力: $u 2^n + v = XY \bmod Z$

$(q_1, r_1) := \mathrm{MulModDiv}(y_1, 2^n, z_1)$
$(q_2, r_2) := \mathrm{MulModDiv}(q_1, z_0, 2^n)$
$(q_3, r_3) := \mathrm{MulModDiv}(x_1, r_1 - q_2 + y_0, z_1)$
$(q_4, r_4) := \mathrm{MulModDiv}(x_0, y_1, z_1)$
$(q_5, r_5) := \mathrm{MulModDiv}(q_3 + q_4, z_0, 2^n)$
$(q_6, r_6) := \mathrm{MulModDiv}(x_1, r_2, 2^n)$
$(q_7, r_7) := \mathrm{MulModDiv}(x_0, y_0, 2^n)$
$u := r_3 + r_4 - q_5 - q_6 + q_7$
$v := r_7 - r_6 - r_5$

ビット長2倍化 (証明)

証明

$(x_1 2^n + x_0)(y_1 2^n + y_0)$**を展開し、**

MulModDiv**による変換と、**

$z_1 2^n = -z_0 \bmod Z$**の関係を用いて簡略化する**

$XY = x_1 y_1 2^n 2^n + \ldots$

$= x_1(q_1 z_1 + r_1)2^n + \ldots$ ← (q_1, r_1)=MulModDiv$(y_1, 2^n, z_1)$

$= x_1(-q_1 z_0 + r_1 2^n) + \ldots \bmod Z$ ← $z_1 2^n = -z_0 \bmod Z$

…

$= u2^n + v$

効率性 (演算回数)

MulModDiv演算の呼出し回数：7回

→ nビット剰余乗算の呼出し回数：14回

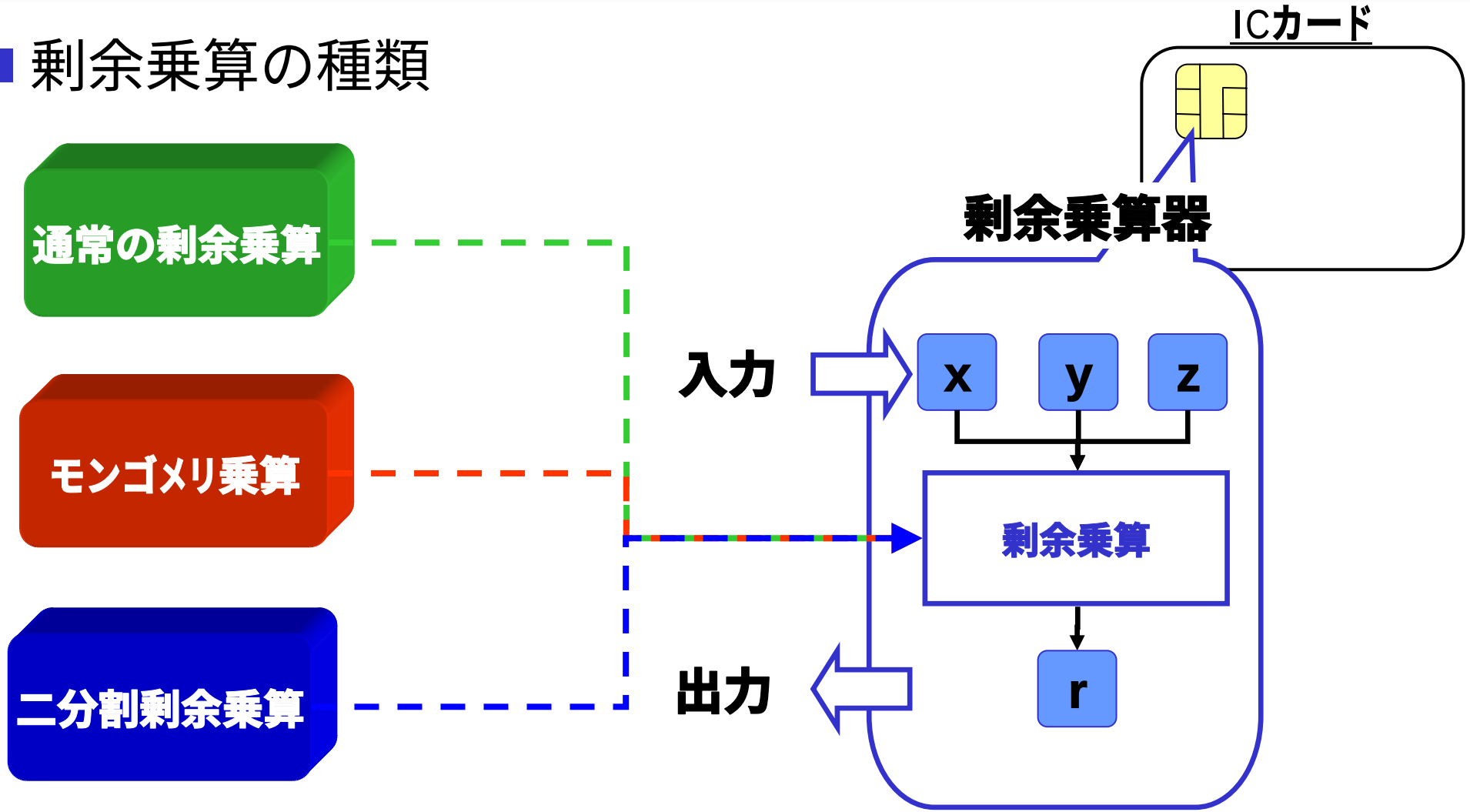
→ 2nビットRSAの暗号化を、
約240回のnビット剰余乗算で実現

現実的な時間で実行できます

目次

色々な剰余乗算

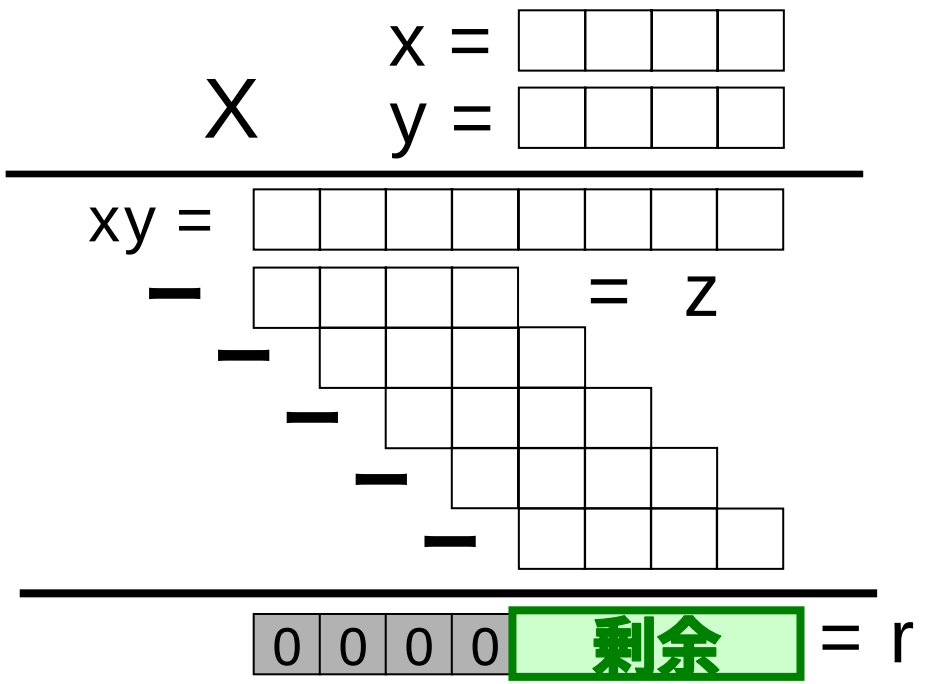
- 剰余乗算の種類



モンゴメリ乗算

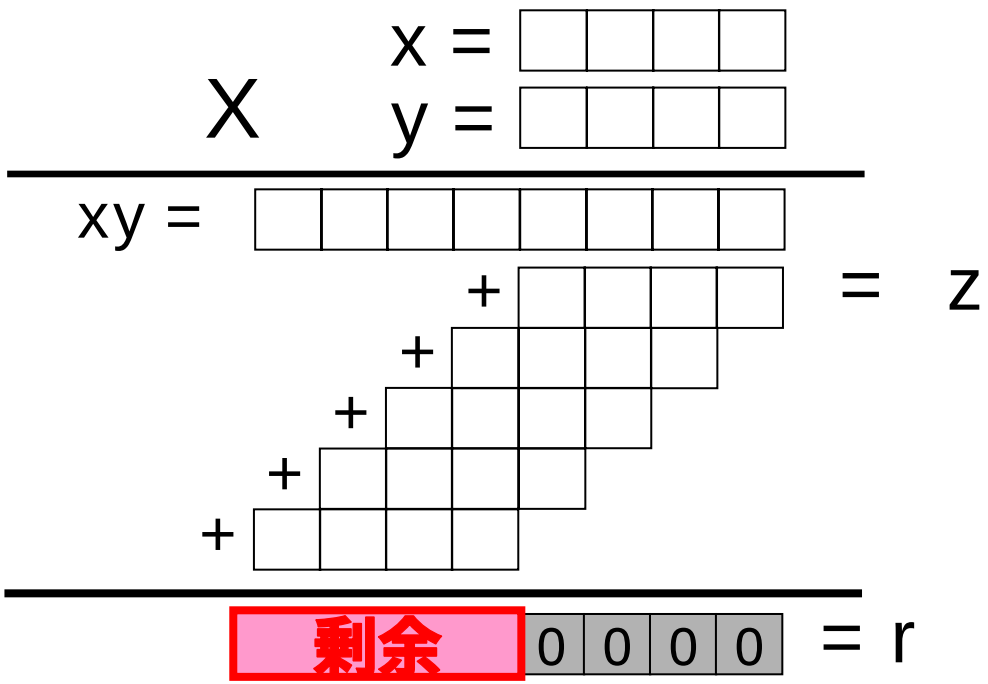
xy mod z

通常の剰余乗算

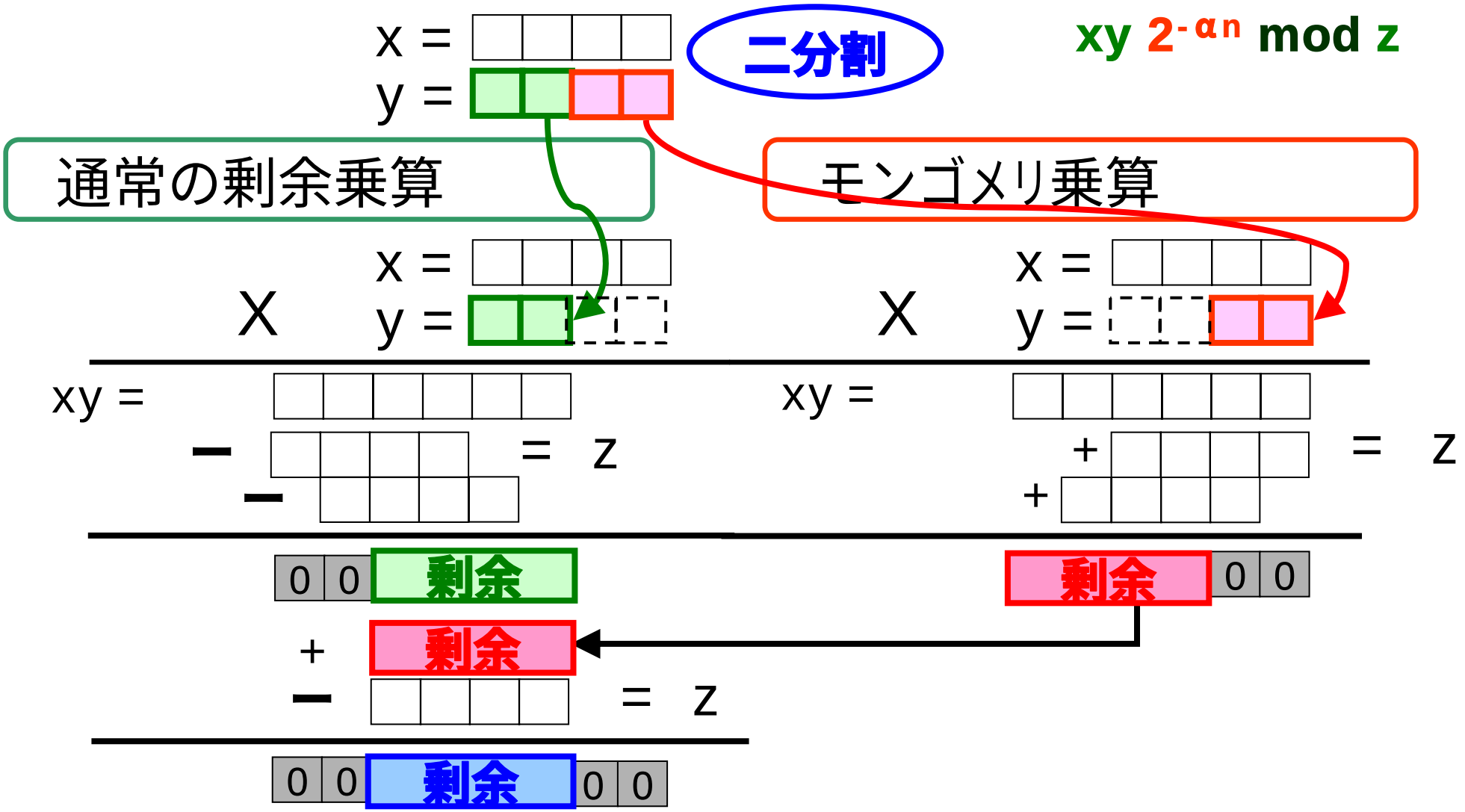


xy 2^{-n} mod z

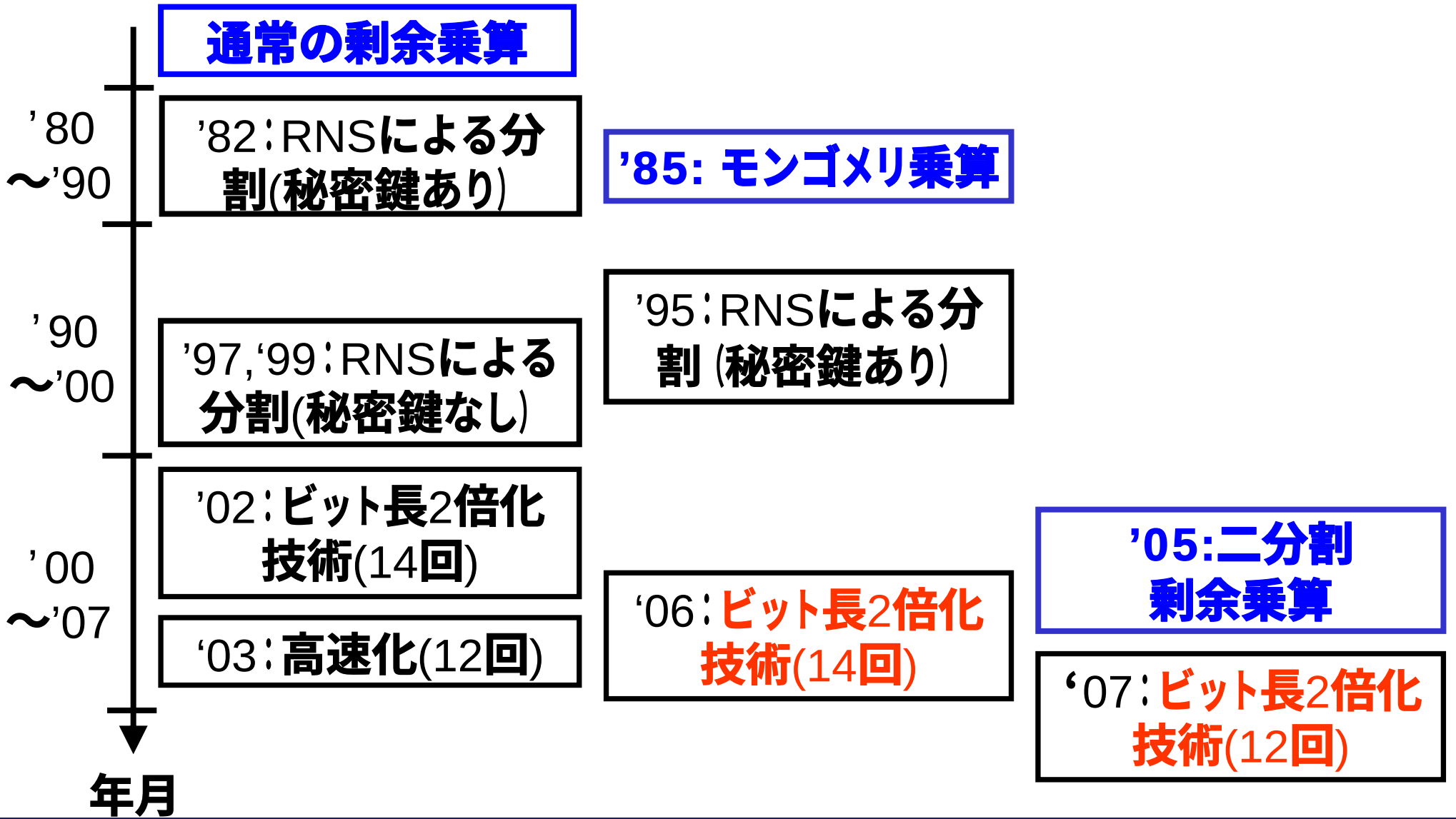
モンゴメリ乗算



二分割剰余乗算


x =
y =
二分割
xy 2^-αn mod z
通常の剰余乗算
モンゴメリ乗算
X
x =
y =
xy =
−
= z
−
0 0
剰余
+
剰余
−
= z
0 0
剰余
0 0
X
x =
y =
xy =
+
= z
+
剰余
0 0

関連研究


通常の剰余乗算
’80
〜’90
’82：RNSによる分割(秘密鍵あり)
’85: モンゴメリ乗算
’90
〜’00
’97,‘99：RNSによる分割(秘密鍵なし)
’95：RNSによる分割 (秘密鍵あり)
’00
〜’07
’02：ビット長2倍化技術(14回)
‘03：高速化(12回)
‘06：ビット長2倍化技術(14回)
’05:二分割剰余乗算
‘07：ビット長2倍化技術(12回)
年月

まとめ

- ビット長2倍化技術の紹介
  - nビットの剰余乗算器を用いて2nビットの剰余乗算を実現
  - 法の因数分解は用いない
  - 主にRSAの暗号化に適用

- 各種拡張方式の紹介
  - 高速化
  - モンゴメリ乗算
  - 二分割剰余乗算

    → 7月にオーストラリアで開催される国際会議で発表します