リスク工学特別演習　20021111

暗　　号
（ＤＥＳ暗号・ＲＳＡ暗号・楕円曲線暗号）

025210　中野孝雄　　025215　村田隆一　　025216　左　瑞麟

指導教官　助教授　片岸　一起　　講師　ミャオ　イン

研究のテーマ

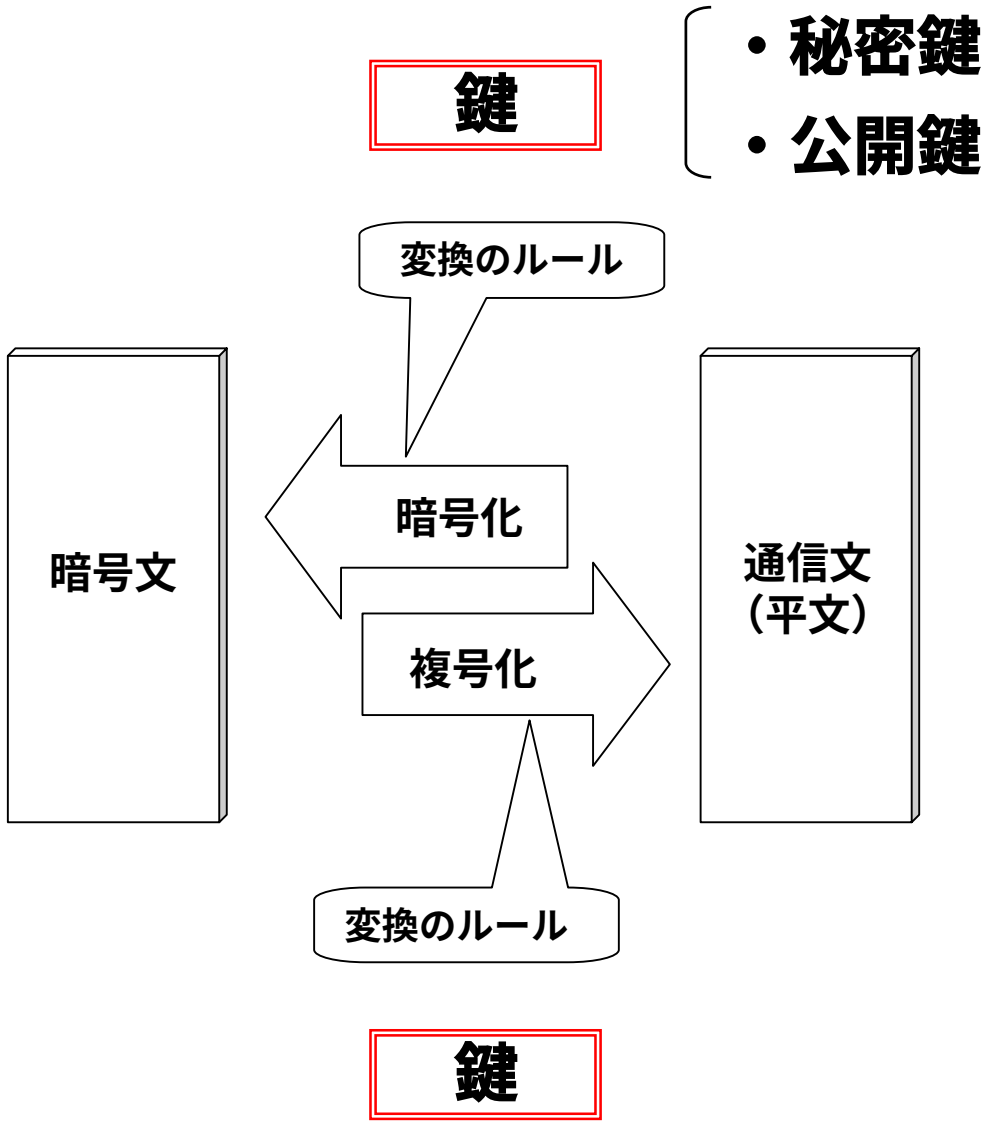
- 秘密鍵暗号と公開鍵暗号の考え方について概観せよ。特にＤＥＳ暗号とＲＡＳ暗号について素人にわかるように説明せよ。

発表の流れ

1：はじめに
2：暗号の概観
3：ＤＥＳ暗号・・・中野
4：ＲＳＡ暗号・・・村田
5：楕円曲線暗号・・・左

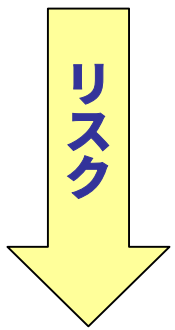
暗号はかつては，軍事や外交などの使用のみに限定されてた。

しかし，近年コンピュータによるネットワークの発達がめざましくなるに伴い，通信の機密を保持するため，ネットークにおいても暗号が使われるようになった。


鍵
• 秘密鍵
• 公開鍵
変換のルール
暗号文
暗号化
複号化
通信文
（平文）
変換のルール
鍵

インターネットを使用して、通信を行う場合、その情報は通信路の途中で、

常に盗聴され悪用される可能性がある。



- 盗聴（秘密鍵）（公開鍵）
- 改ざん（公開鍵）
- なりすまし（公開鍵）

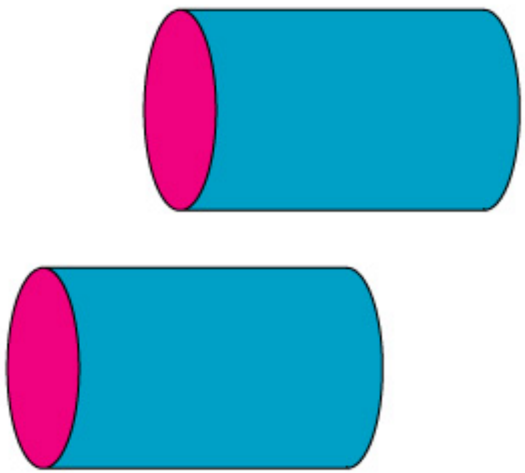
盗聴されても意味が判読できないように

→　暗号化

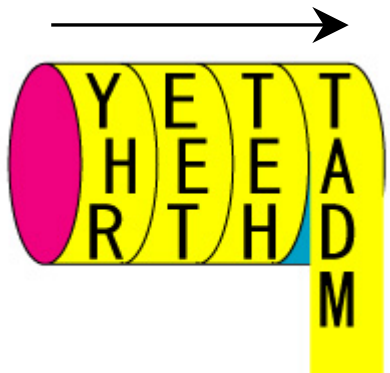
スキュタレー暗号

- 最も古い暗号
- 古代ギリシャで使用

YET THE EARTH DOES MOVE を暗号化



1：同じ太さの棒を2本用意



2：テープを巻き文を書く

3：棒から巻き取り暗号化

シーザー暗号

・現在にも影響を残している暗号
・ジュリアス・シーザーが使用

I LOVE YOU を暗号化

平アルファベット　：A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

暗号アルファベット：D E F G H I J K L M N O P Q R S T U V W X Y Z A B C

I LOVE YOU ⇒ L ORYH BRX

多表式暗号から暗号機、コンピュータ暗号へ

多項式暗号

・単文字換時暗号を基本に変換を複雑に・・・

→現在までの暗号の基本的考えの主流

⇩

暗号機

・世界大戦をきっかけに発展

⇩

コンピュータの暗号

・数学的な原理を用いて複雑な処理で暗号化、復号化

暗号化とは

「秘密通信のため当事者間のみが了解するルールに従って通信文を第三者に理解できない情報に変換すること」である。

“当事者間のみが了解するルール”に

「変換の手法（アルゴリズム（計算の手順））」

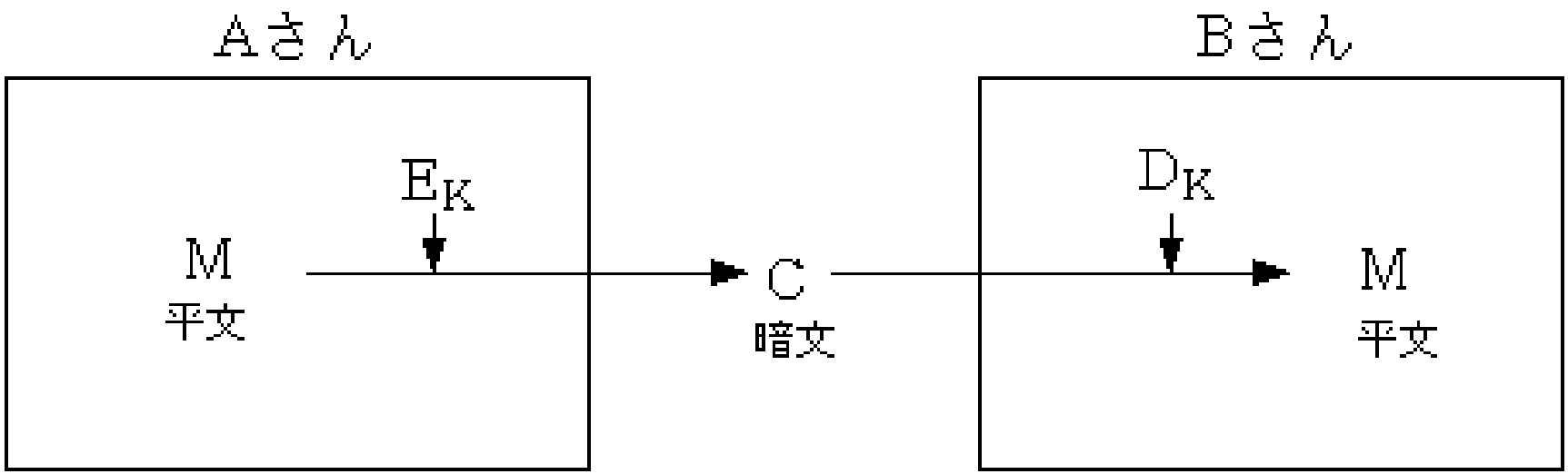
「鍵」という2つの要素が含まれている。

秘密鍵暗号と公開鍵暗号

	アルゴリズム	鍵
秘密鍵暗号	公開	秘密(鍵は1つ)
公開鍵暗号	公開	暗号化鍵は公開 復号化鍵は秘密 (鍵は2つ)

秘密鍵暗号

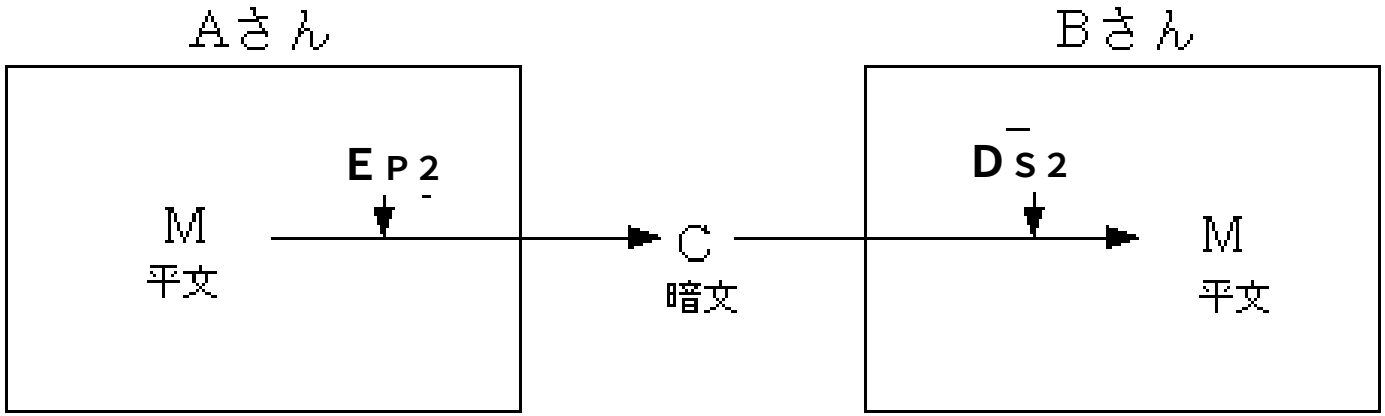
図はAさんからBさんへ共有鍵暗号で暗号化して
メッセージを送信した時のプロセス



E ：組立関数
D ：翻訳関数
K ：鍵

公開鍵暗号

図はAさんからBさんへ公開鍵暗号で暗号化して メッセージを送信した時のプロセス



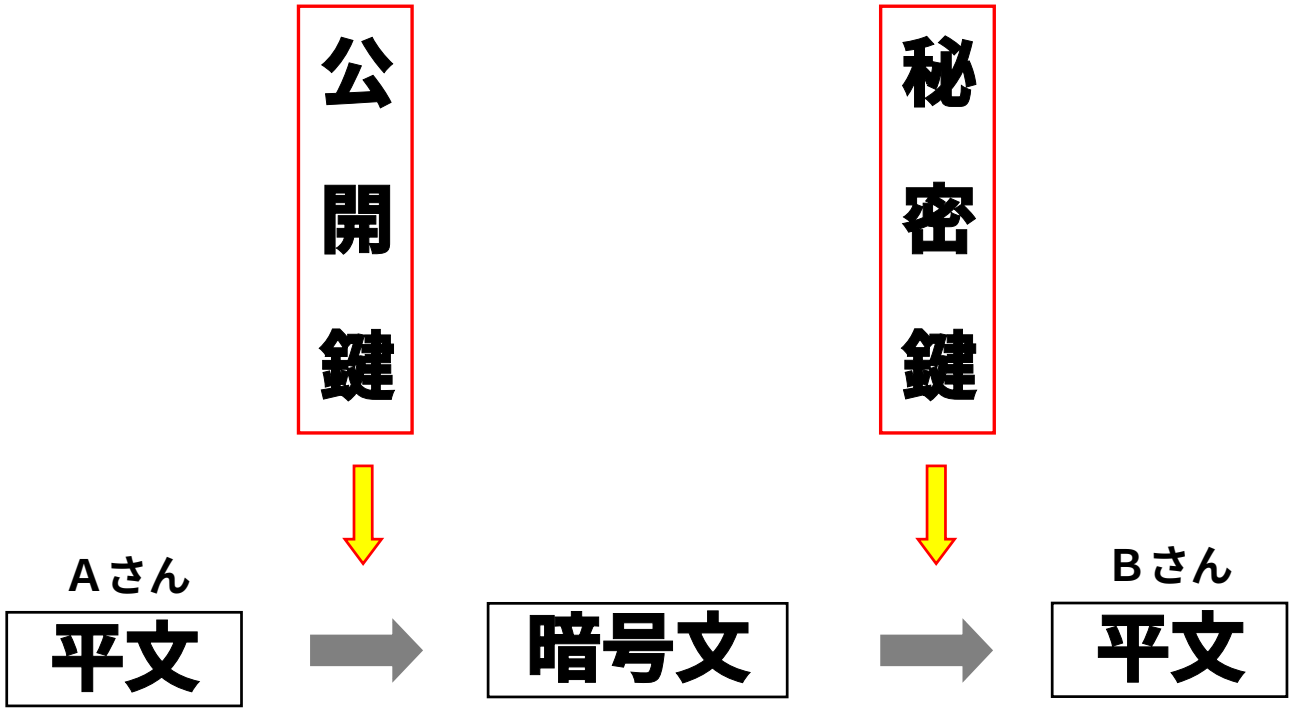
E：組立関数
D：翻訳関数
P２：Bさんの公開鍵
S２：Bさんの秘密鍵

鍵の組み合わせ

Aさん〔Ｐ１（公開鍵）・Ｓ１（秘密鍵）〕
Bさん〔Ｐ２（公開鍵）・Ｓ２（秘密鍵）〕

公開鍵暗号２

鍵の組み合わせ
Ａさん〔Ｐ１（公開鍵）・Ｓ１（秘密鍵）〕
Ｂさん〔Ｐ２（公開鍵）・Ｓ２（秘密鍵）〕



（Ａさん秘密）S1・・・×・・・P1（Ａさん公開）

（Ａさん公開）P1・・・×・・・S1（Ａさん秘密）

（Ｂさん公開）P2・・・◯・・・S2（Ｂさん秘密）

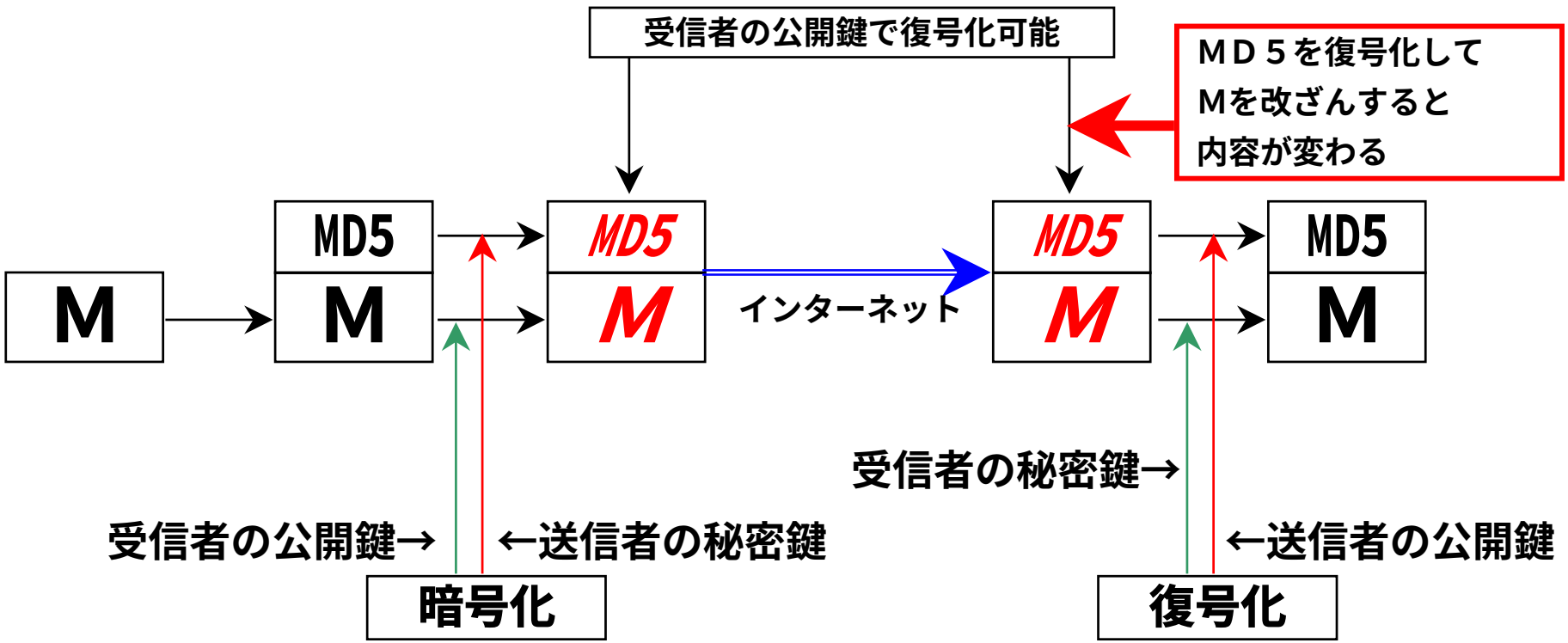
公開鍵暗号３

- 盗聴
- 改ざん
- なりすまし

「改ざん」や「なりすまし」対策に

MD5のメッセージを変えると
Mのメッセージ内容が変わる

ＭＤ５（Message Digestion Ver5)128bit



共通鍵暗号方式と公開鍵暗号方式

共通鍵暗号方式

名称	鍵長	特徴
DES	56bit	1977年米国で標準化。1988年ANSI標準。使用実績が最も多いが、解読されたといわれている。
トリプルDES	56x2 bit	DESのアルゴリズムを異なる２つの鍵で３回実行し鍵長を２倍にする。金融機関で使用され、DESの寿命を伸ばすのに一役買っている。

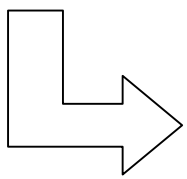
公開鍵暗号方式

名称	鍵長	特徴
RSA	1024bit	RSAはその考案者の三人の頭文字をとったものである。一般向けのセキュリティ機能付きソフトは、ほとんどRSAを使用。
楕円曲線	160bit	楕円曲線と呼ばれる数式によって定義される特殊な加算法で暗号化される。

ビット（bit：binary　digit）

・コンピュータで扱うデータの最小単位
・１つのビットは「０」または「１」
・1つのビットでは０と１の２種類のデータしか表現できない
・ビットの数を増やせば、多くのデータを表現することができる。

１ビットあれば　２つの中から１つを指定（識別）できる。
２ビットあれば　４つの中から１つを指定（識別）できる。
３ビットあれば　８つの中から１つを指定（識別）できる。
・・・・・・（中略）・・・・・・
ｎビットあれば　２ⁿの中から１つを指定（識別）できる。

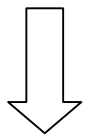


＊３ビットで８ケースを表現＊

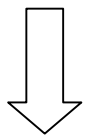
１１１	１が３個
１１０	１が２個
１０１	
０１１	
１００	１が１個
０１０	
００１	
０００	１が０個

８ビットを１バイト

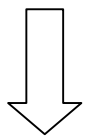
８つの信号（１か０）がひとかたまりになって記憶装置に入っていく。



１バイトは2^8＝２５６個を識別できる情報量



日本語のかな文字、アルファベット、各種符号の合計が
256個に満たないので、1バイトで識別できる。



ＪＩＳ規格では

１	は　００１１０００１	Ａ	は　０１０００００１
イ	は　１０１１００１０	％	は　００１００１０１

ＤＥＳ　共通鍵暗号化方式の代表的な規格

・1970年代はじめ米国政府が政府内のデータの暗号化に関する標準規格を確立しようとしておこなった公募から始まっている。

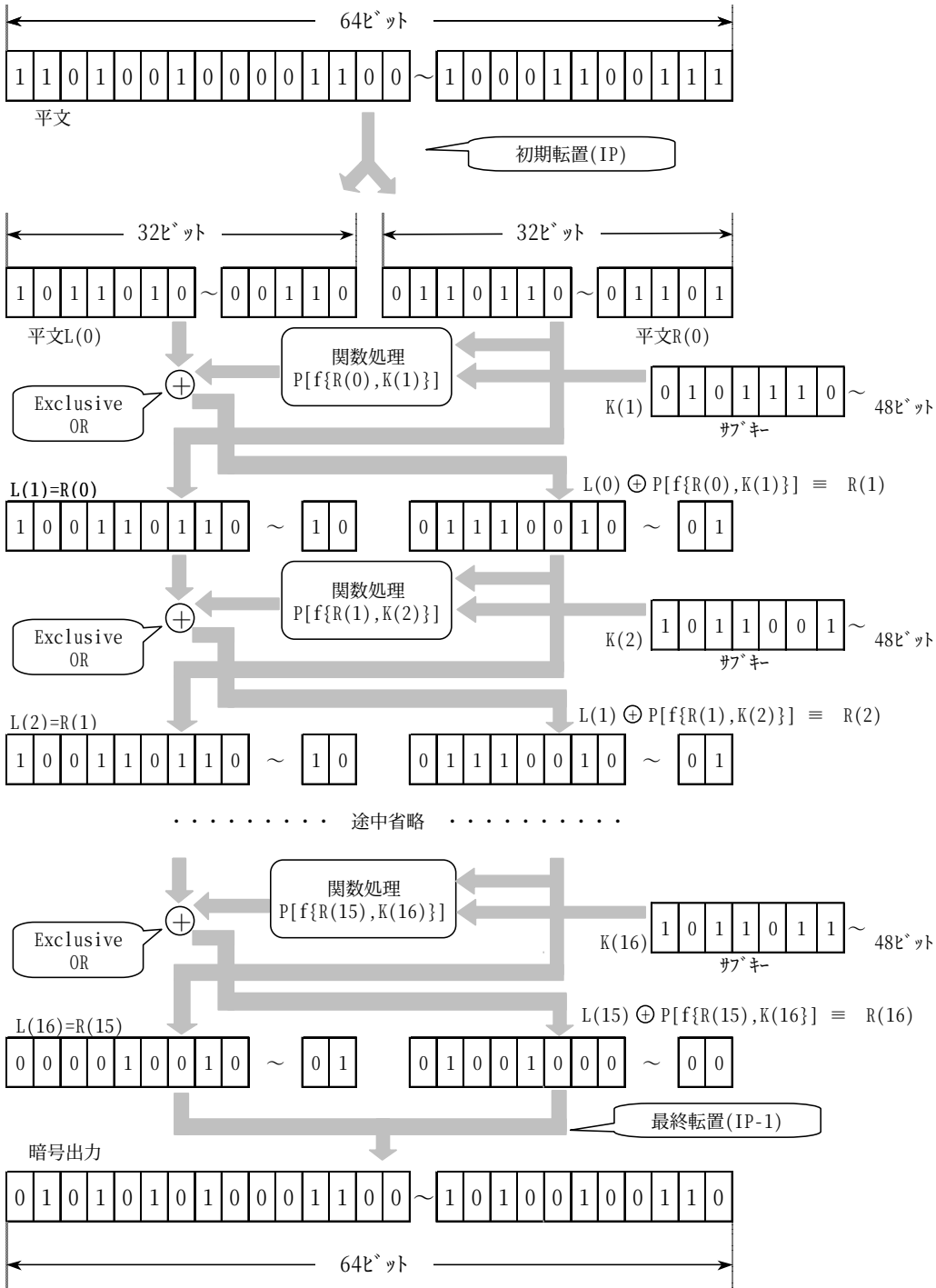
・ＩＢＭ社が開発した暗号化方式を基に、ＮＩＳＴ（米国立標準技術研究所）が規格化し、米国政府が標準暗号化方式として採用した。

・ＤＥＳで使用する**共通鍵は５６ビット**の長さである。
→鍵の長さが強さを（総当りで２の５６乗）

・元の**情報を６４ビット**ずつに区切って、この単位ごとに５６ビットの共通鍵を使って暗号化していく。

・６４ビットは８バイト
アルファベット１文字は１バイトで表されるので
英文を８文字ずつに区切って暗号化していく
暗号文も６４ビットごとに区切られたものとして出力される。

暗号化の全体像

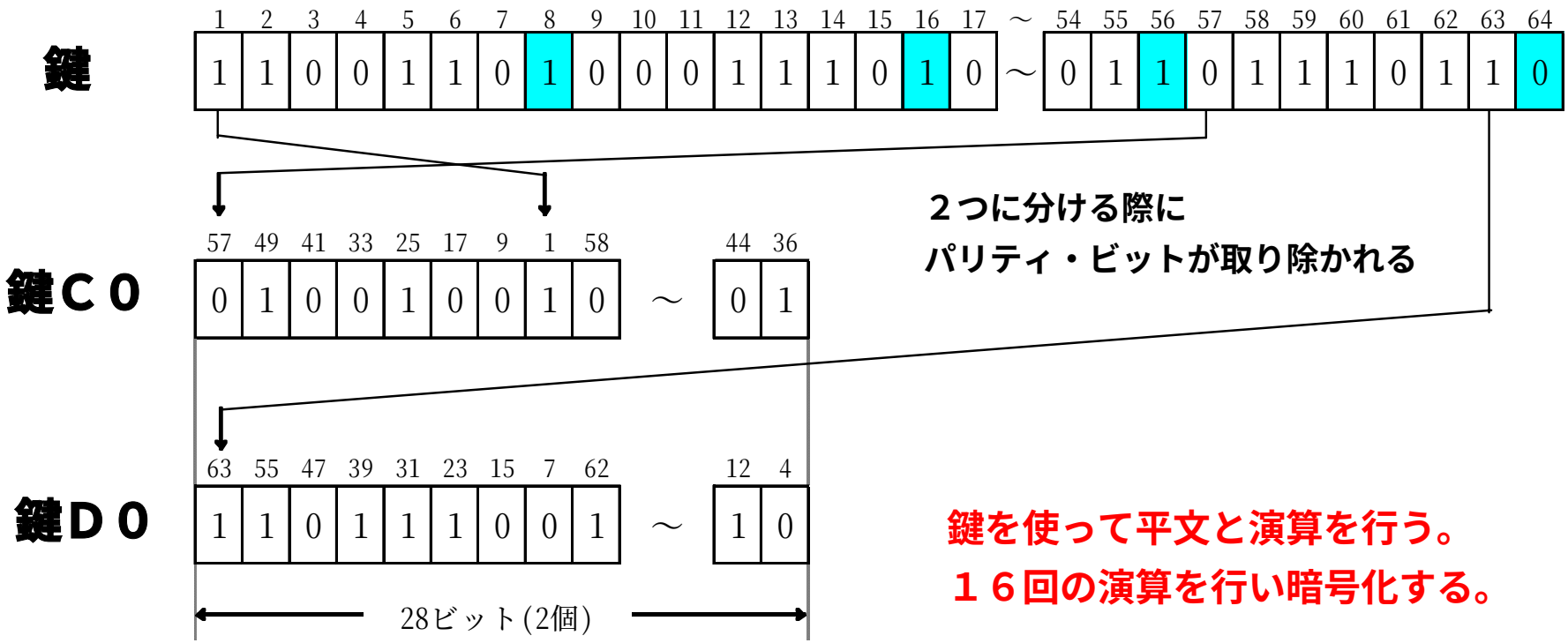

64ﾋﾞｯﾄ
1 1 0 1 0 0 1 0 0 0 0 1 1 0 0 ～ 1 0 0 0 1 1 0 0 1 1 1
平文
初期転置(IP)
32ﾋﾞｯﾄ
32ﾋﾞｯﾄ
1 0 1 1 0 1 0 ～ 0 0 1 1 0
0 1 1 0 1 1 0 ～ 0 1 1 0 1
平文L(0)
平文R(0)
関数処理
P[f{R(0),K(1)}]
Exclusive
OR
K(1)
0 1 0 1 1 1 0 ～ 48ﾋﾞｯﾄ
ｻﾌﾞｷｰ
L(1)=R(0)
L(0) ⊕ P[f{R(0),K(1)}] ≡ R(1)
1 0 0 1 1 0 1 1 0 ～ 1 0
0 1 1 1 0 0 1 0 ～ 0 1
関数処理
P[f{R(1),K(2)}]
Exclusive
OR
K(2)
1 0 1 1 0 0 1 ～ 48ﾋﾞｯﾄ
ｻﾌﾞｷｰ
L(2)=R(1)
L(1) ⊕ P[f{R(1),K(2)}] ≡ R(2)
1 0 0 1 1 0 1 1 0 ～ 1 0
0 1 1 1 0 0 1 0 ～ 0 1
・・・・・・・・・・ 途中省略 ・・・・・・・・・・
関数処理
P[f{R(15),K(16)}]
Exclusive
OR
K(16)
1 0 1 1 0 1 1 ～ 48ﾋﾞｯﾄ
ｻﾌﾞｷｰ
L(16)=R(15)
L(15) ⊕ P[f{R(15),K(16}] ≡ R(16)
0 0 0 0 1 0 0 1 0 ～ 0 1
0 1 0 0 1 0 0 0 ～ 0 0
最終転置(IP-1)
暗号出力
0 1 0 1 0 1 0 1 0 0 0 1 1 0 0 ～ 1 0 1 0 0 1 0 0 1 1 0
64ﾋﾞｯﾄ

暗号化鍵の処理

（ＰＣ－１）

	鍵出力	暗号鍵						
C(0)	1-7	57	49	41	33	25	17	9
	8-14	1	58	50	42	34	26	18
	15-21	10	2	59	51	43	35	27
	22-28	19	11	3	60	52	44	36
D(0)	1-7	63	55	47	39	31	23	15
	8-14	7	62	54	46	38	30	22
	15-21	14	6	61	53	45	37	29
	22-28	21	13	5	28	20	12	4

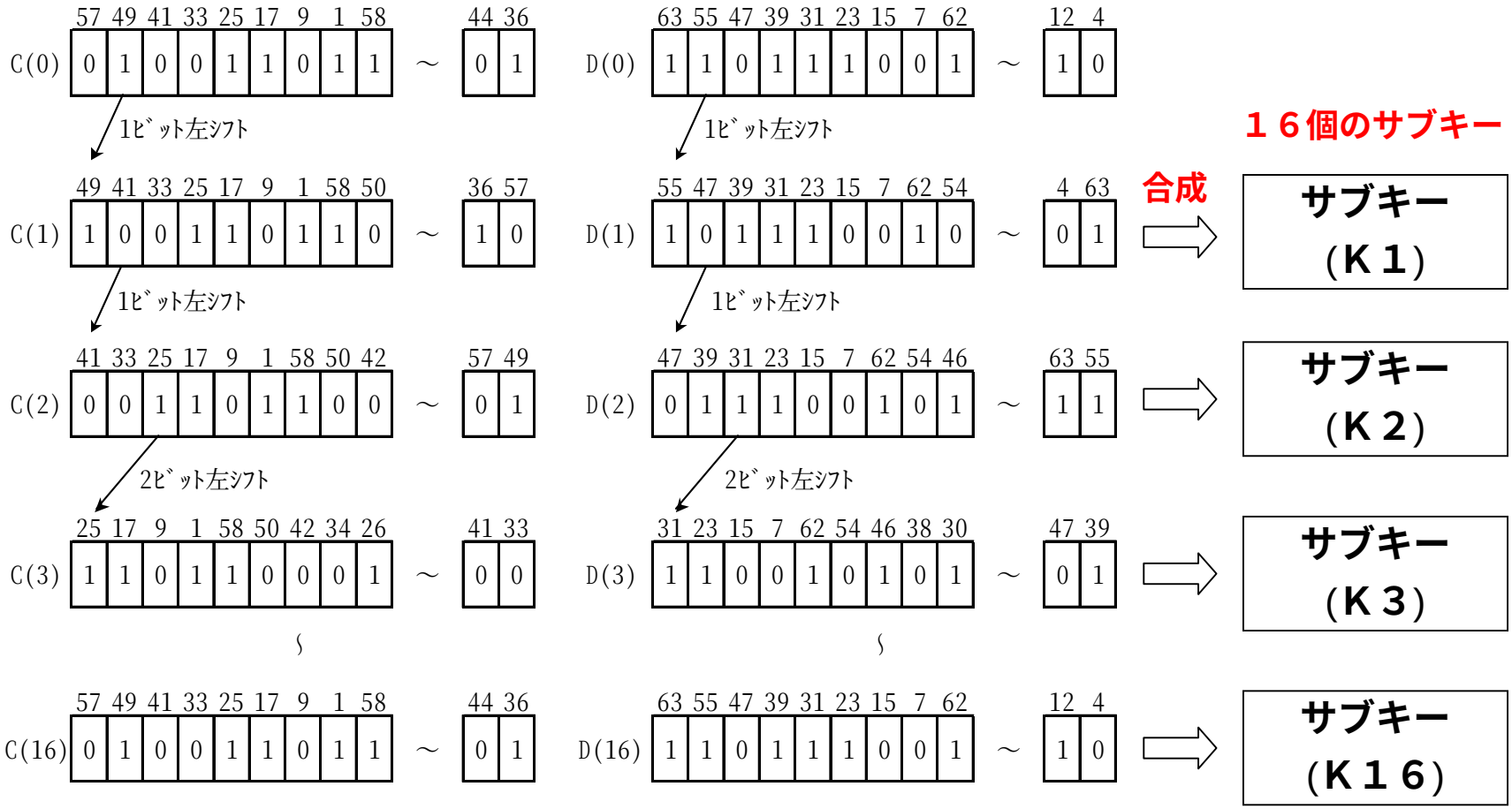
64ビットの鍵から2つの28ビット鍵をつくる選択配置表



鍵を使って平文と演算を行う。

１６回の演算を行い暗号化する。

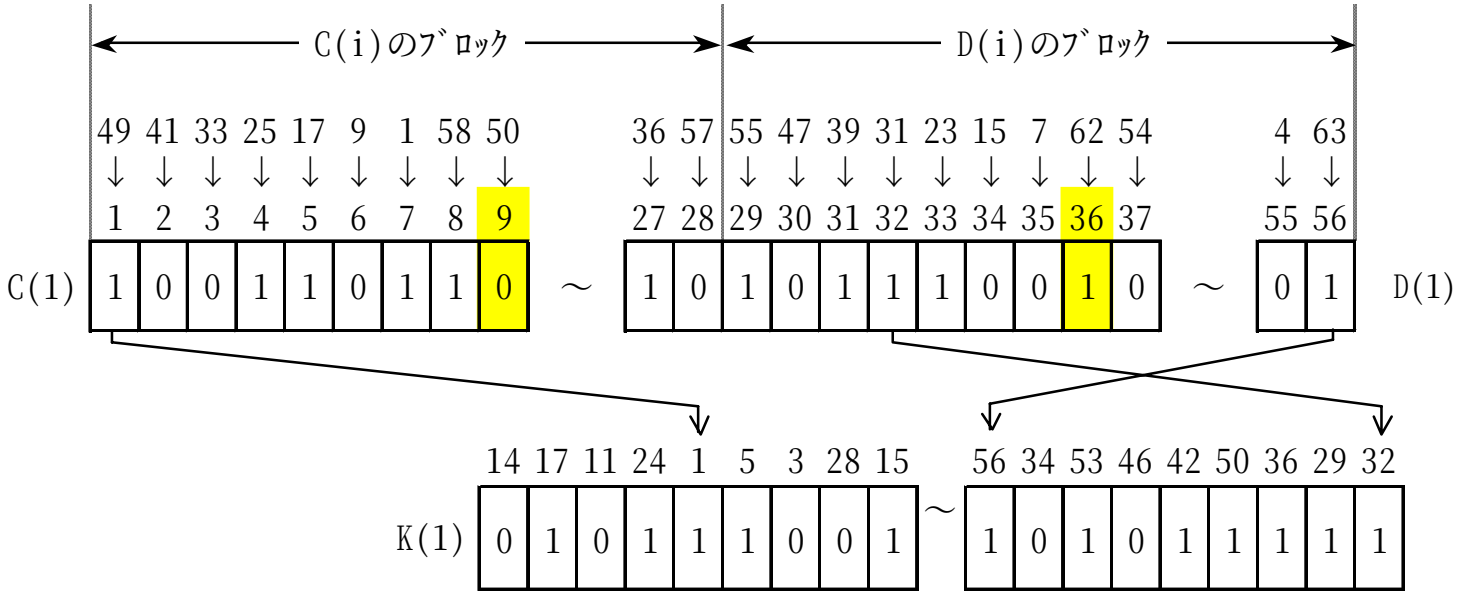
サブキーを作る



段数　i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
シフト数	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

左シフトテーブル

サブキー(Ki)の合成



Ｃ（ｉ）とＤ（ｉ）を合成してサブキーＫ（ｉ）を作成

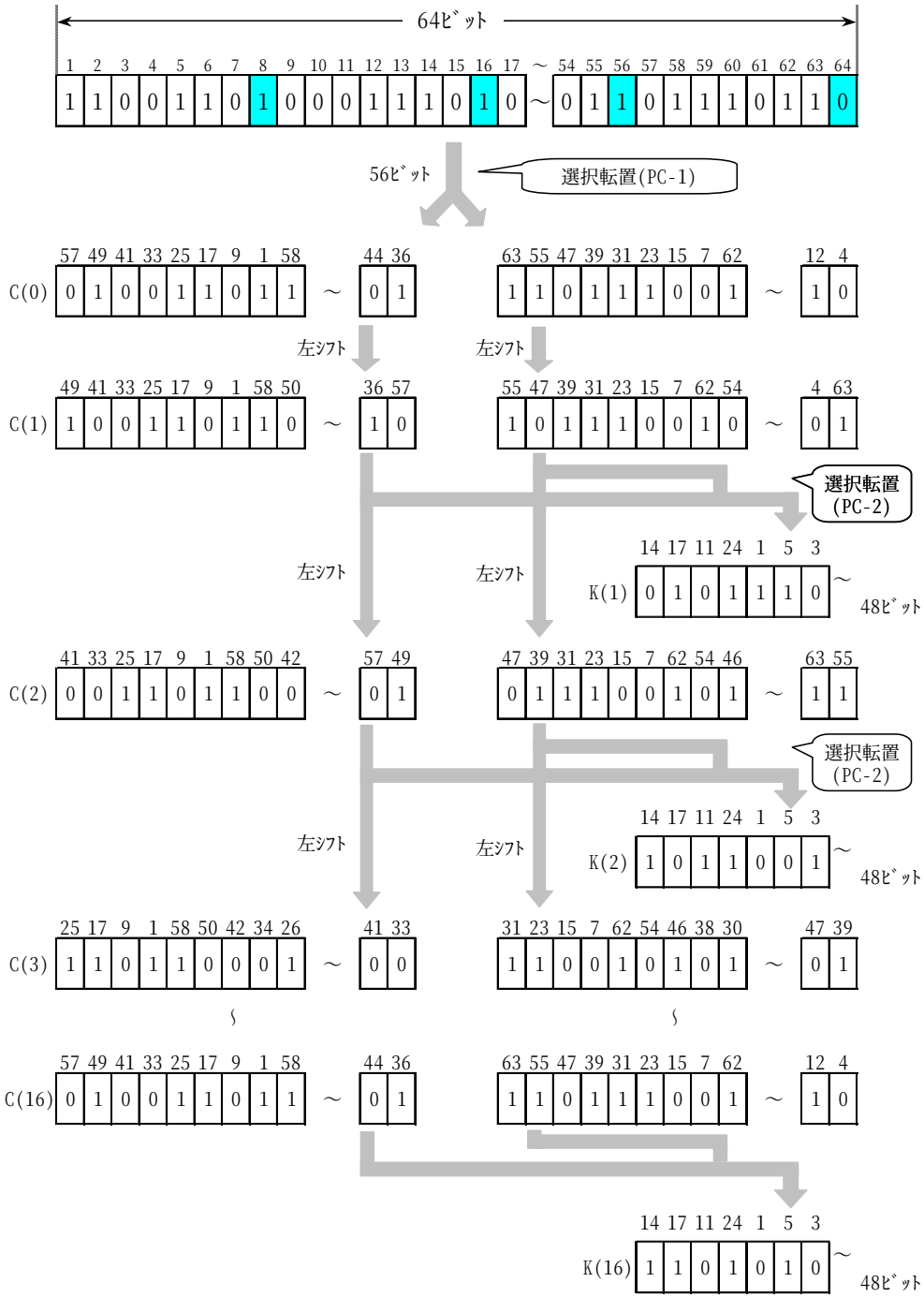
９,18,27,36,45,54は
選択転置の際に削除される。
→48ビットの鍵になる。

（ＰＣ－２）

K(i)出力ﾋﾞｯﾄ	各段のC(ｉ),D(ｉ)入力ﾋﾞｯﾄ					
1-6	14	17	11	24	1	5
7-12	3	28	15	6	21	10
13-18	23	19	12	4	26	8
19-24	16	7	27	20	13	2
25-30	41	52	31	37	47	55
31-36	30	40	51	45	33	48
37-42	44	49	39	56	34	53
43-48	46	42	50	36	29	32

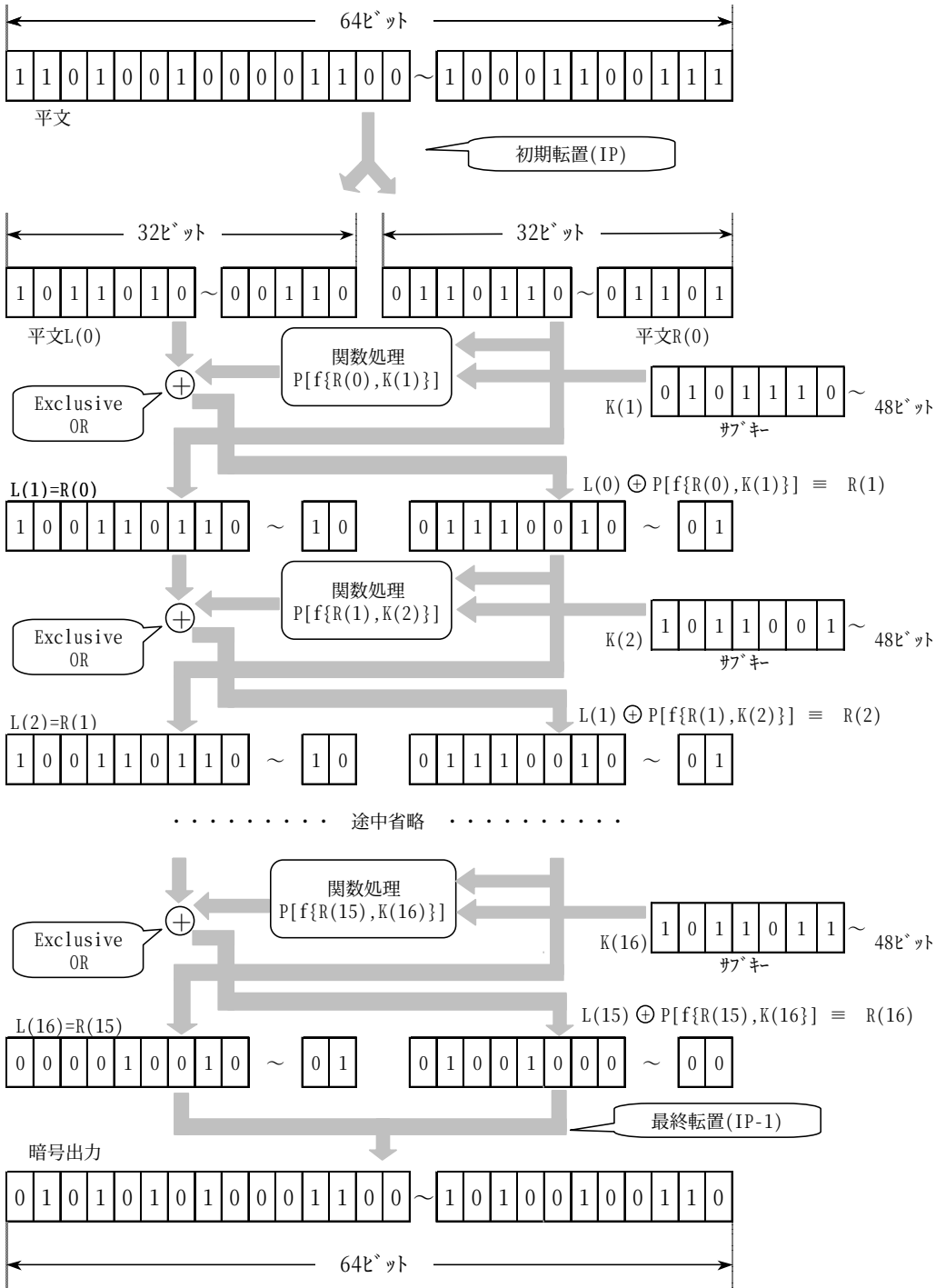
サブキーK(ｉ)の合成に用いる選択配置表

鍵処理のフロー



完成した16個の
サブキーを使って
関数処理をして
暗号化していく

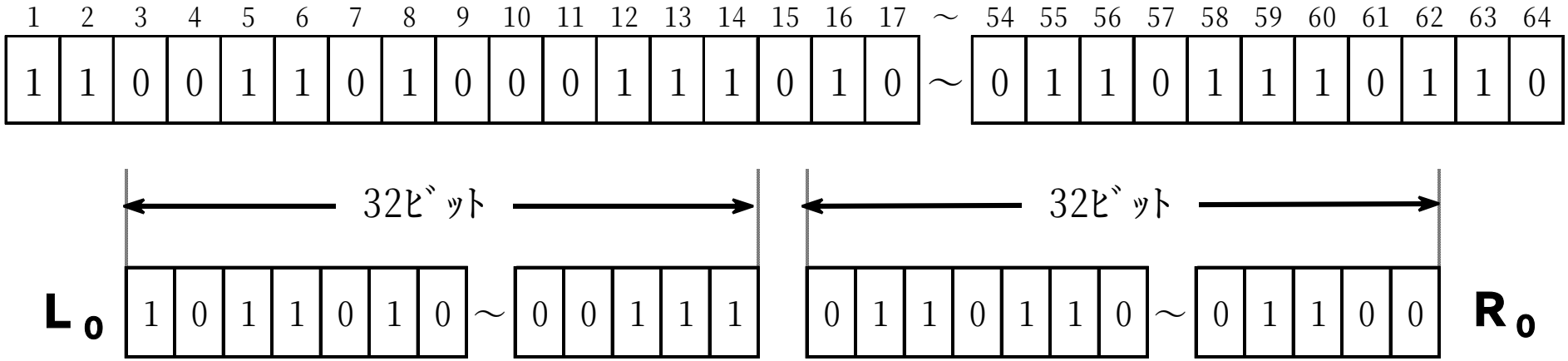
暗号化の全体像


64ビット
1 1 0 1 0 0 1 0 0 0 0 1 1 0 0 ～ 1 0 0 0 1 1 0 0 1 1 1
平文
初期転置(IP)
32ビット
32ビット
1 0 1 1 0 1 0 ～ 0 0 1 1 0
0 1 1 0 1 1 0 ～ 0 1 1 0 1
平文L(0)
平文R(0)
関数処理
P[f{R(0),K(1)}]
Exclusive
OR
K(1)
0 1 0 1 1 1 0 ～ 48ビット
ｻﾌﾞｷｰ
L(1)=R(0)
L(0) ⊕ P[f{R(0),K(1)}] ≡ R(1)
1 0 0 1 1 0 1 1 0 ～ 1 0
0 1 1 1 0 0 1 0 ～ 0 1
関数処理
P[f{R(1),K(2)}]
Exclusive
OR
K(2)
1 0 1 1 0 0 1 ～ 48ビット
ｻﾌﾞｷｰ
L(2)=R(1)
L(1) ⊕ P[f{R(1),K(2)}] ≡ R(2)
1 0 0 1 1 0 1 1 0 ～ 1 0
0 1 1 1 0 0 1 0 ～ 0 1
・・・・・・・・・・ 途中省略 ・・・・・・・・・・
関数処理
P[f{R(15),K(16)}]
Exclusive
OR
K(16)
1 0 1 1 0 1 1 ～ 48ビット
ｻﾌﾞｷｰ
L(16)=R(15)
L(15) ⊕ P[f{R(15),K(16}] ≡ R(16)
0 0 0 0 1 0 0 1 0 ～ 0 1
0 1 0 0 1 0 0 0 ～ 0 0
最終転置(IP-1)
暗号出力
0 1 0 1 0 1 0 1 0 0 0 1 1 0 0 ～ 1 0 1 0 0 1 0 0 1 1 0
64ビット

初期転置IP(Initial Permutation)

入力された平文の文字コードをビット単位で並び替える

64ビットずつ暗号化（８文字）



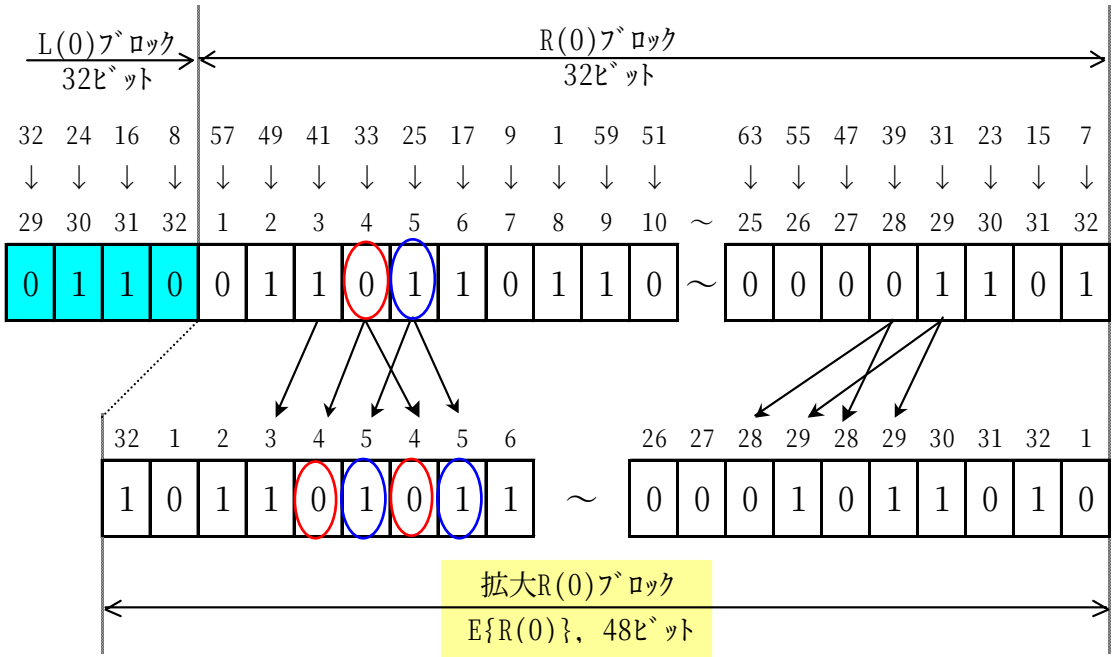
IP								
初期転置出力	文字コード化された平文入力ﾋﾞｯﾄ							
1-8	58	50	42	34	26	18	10	2
9-16	60	52	44	36	28	20	12	4
17-24	62	54	46	38	30	22	14	6
25-32	64	56	48	40	32	24	16	8
33-40	57	49	41	33	25	17	9	1
41-48	59	51	43	35	27	19	11	3
49-56	61	53	45	37	29	21	13	5
57-64	63	55	47	39	31	23	15	7

L0

R0

初期転置の規則表

平文系Ｒ（ｉ）32ﾋﾞｯﾄ拡大処理



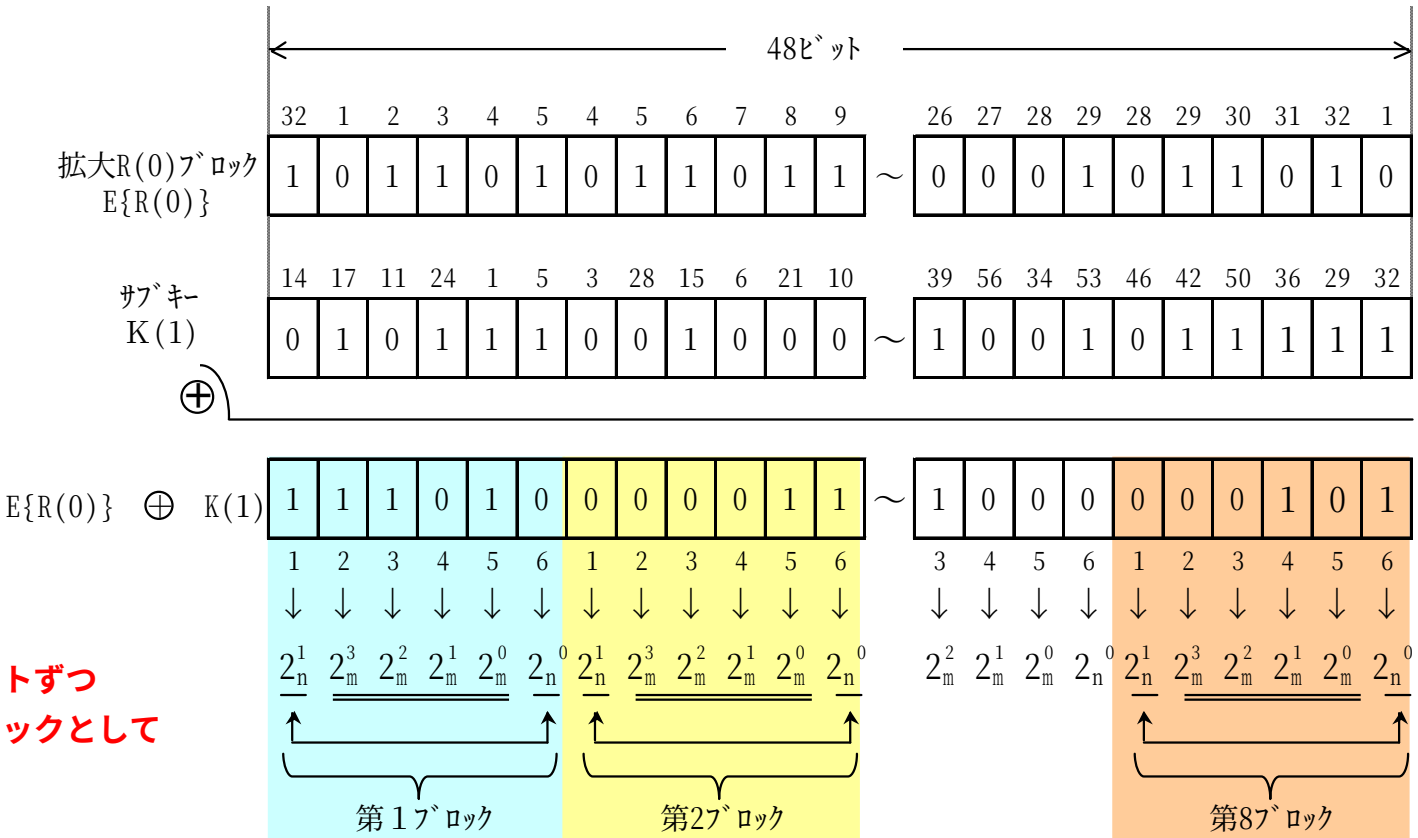
関数処理をサブキー48ビットとの間でおこなうので32ビットを48ビットへ拡大処理する

Ｌブロックの３２ビットに関しては拡大処理を含む関数処理はしない

出力ﾋﾞｯﾄ	Eﾋﾞｯﾄ選択表					
1-6	32	1	2	3	4	5
7-12	4	5	6	7	8	9
13-18	8	9	10	11	12	13
19-24	12	13	14	15	16	17
25-30	16	17	18	19	20	21
31-36	20	21	22	23	24	25
37-42	24	25	26	27	28	29
43-48	28	29	30	31	32	1

拡大配置表(E)

Ｒブロックと次段サブキーを加算



８バイトずつ
１ブロックとして

ExclusiveORの論理による加算

ExclusiveORの論理では、

１＋１＝０・・・答え偶数→０

１＋０＝１・・・答え奇数→１

０＋０＝０・・・答え偶数→０

選択関数表による換字処理へ

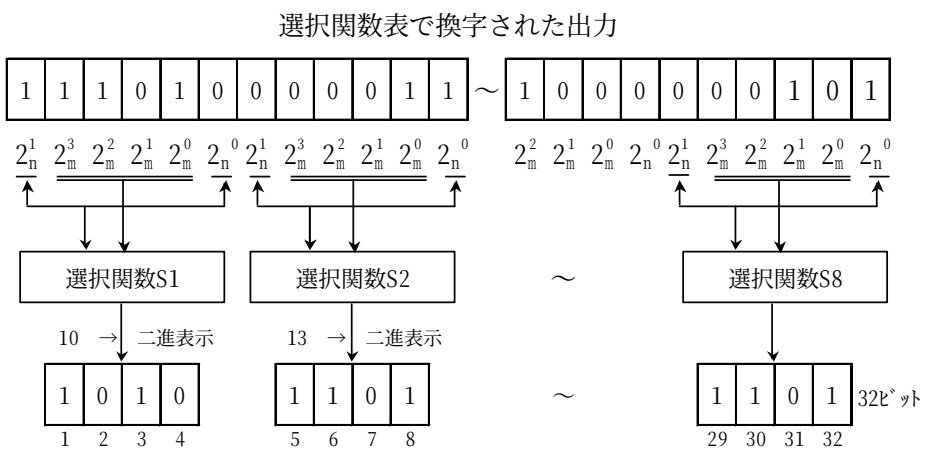
換字選択関数表による換字処理

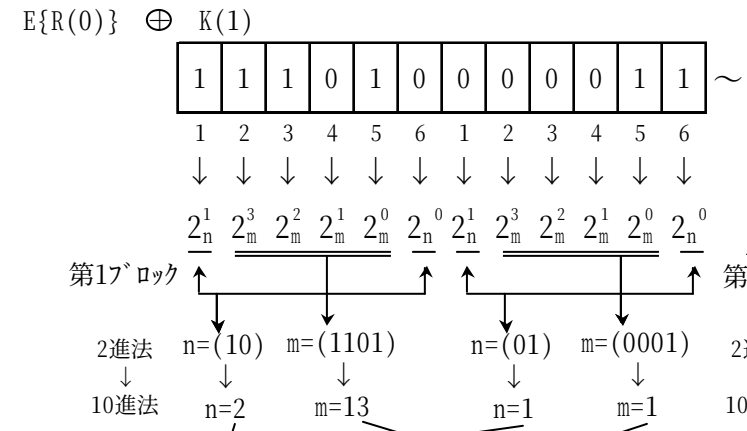
ブロックごとに
換字を実施

２進法表記を１０進法表記にして
列m、行ｎのポインターに

換字された１０進法を
再び２進法表記（４ビット）に

８つのブロックを合成し
３２ビットへ





		列番号m															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ﾌﾞﾛｯｸ1 S1	0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
ﾌﾞﾛｯｸ2 S2	0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
ﾌﾞﾛｯｸ3 S3	0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
ﾌﾞﾛｯｸ4 S4	0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
ﾌﾞﾛｯｸ5 S5	0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

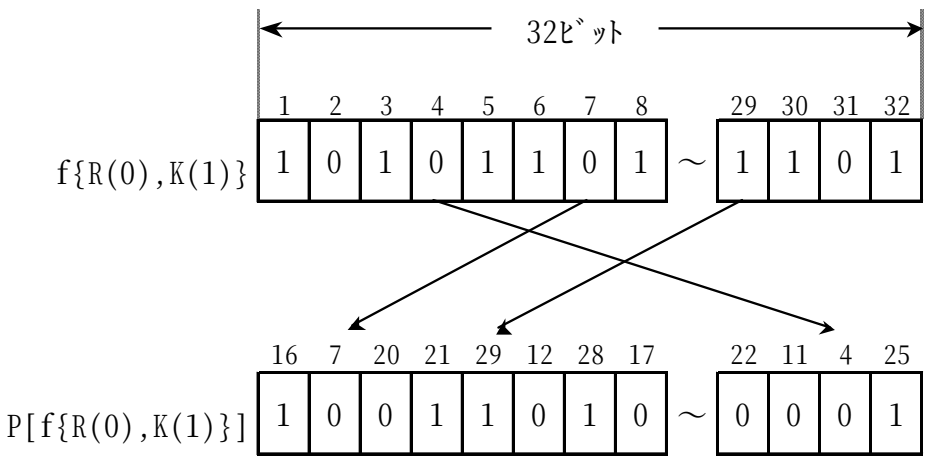
行番号n

関数処理後の転置

転置出力ﾋﾞｯﾄ	入力ﾋﾞｯﾄ			
1-4	16	7	20	21
5-8	29	12	28	17
9-12	1	15	23	26
13-16	5	18	31	10
17-20	2	8	24	14
21-24	32	27	3	9
25-28	19	13	30	6
29-32	22	11	4	25

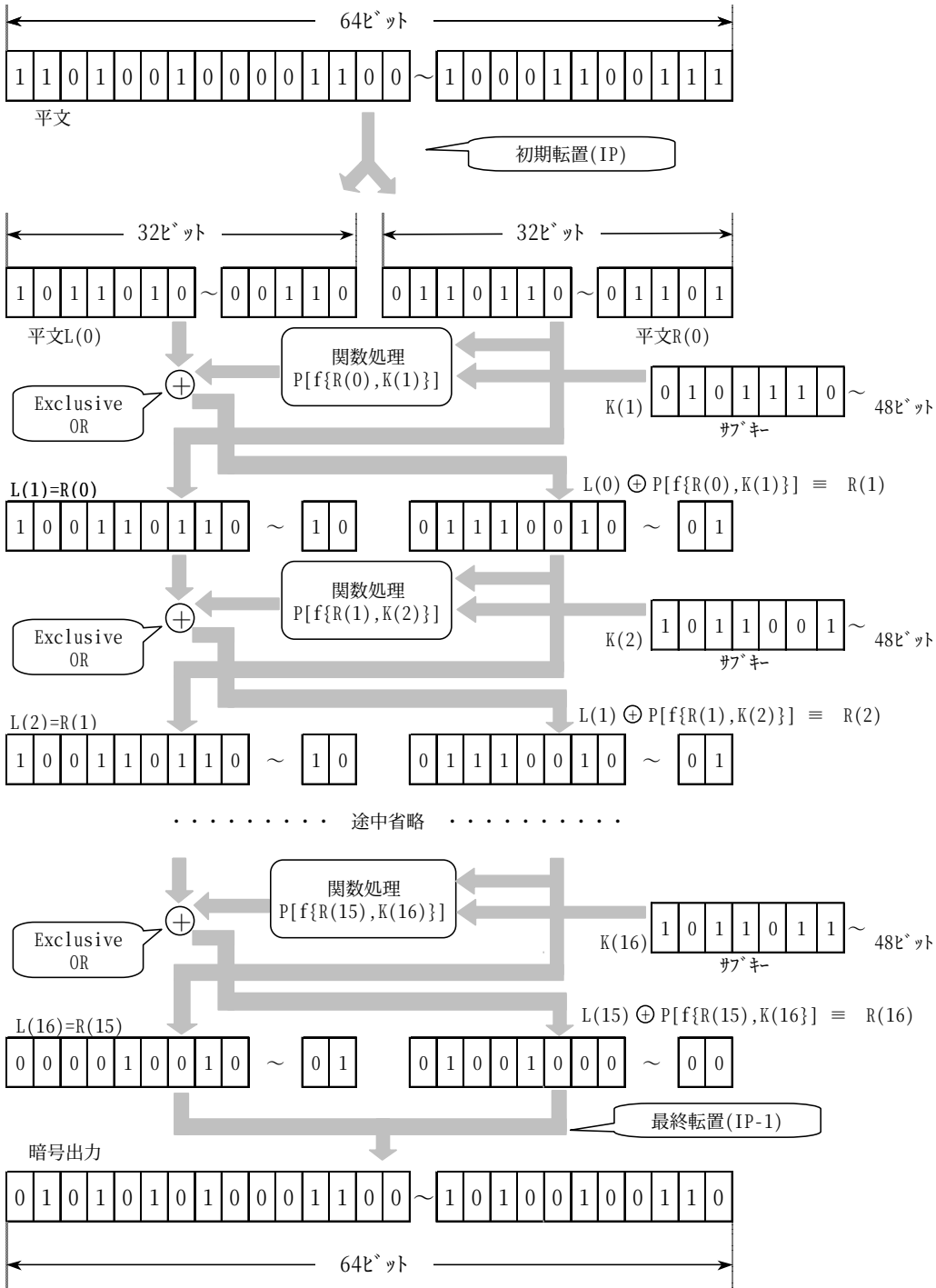
選択関数処理後の転置表（Ｐ転置）

３２ビットに関数処理されたものを
転置表で再び転置

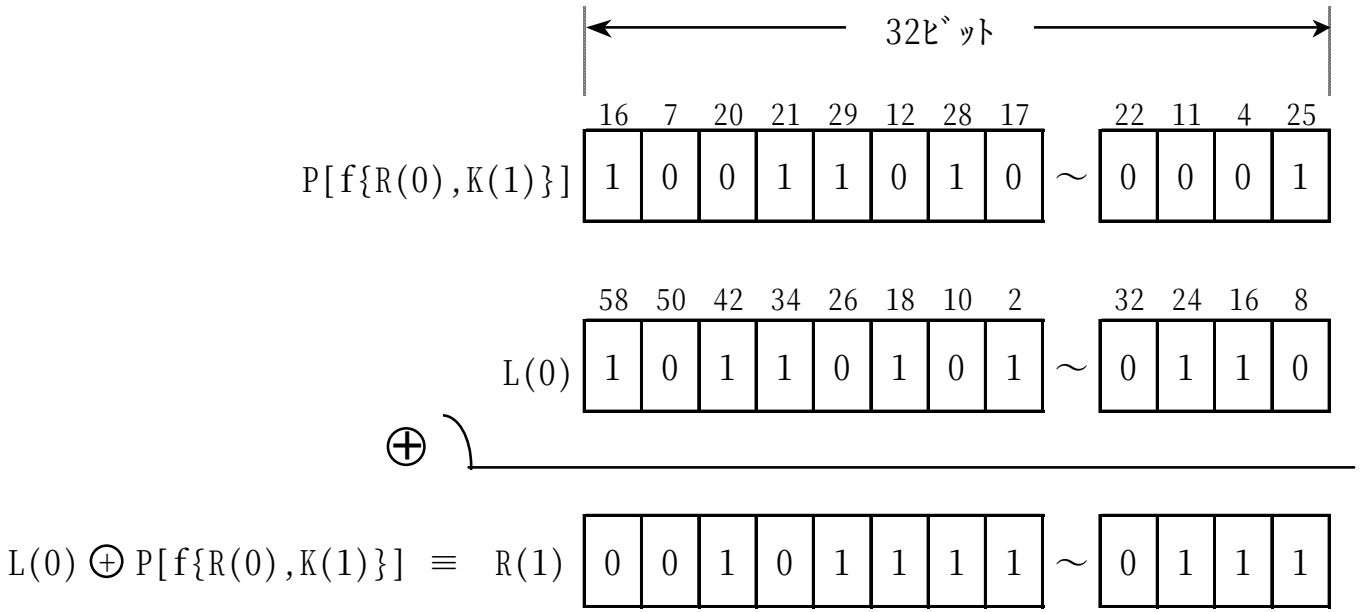


選択関数処理後の転置

暗号化の全体像


64ビット
1 1 0 1 0 0 1 0 0 0 0 1 1 0 0 ～ 1 0 0 0 1 1 0 0 1 1 1
平文
初期転置(IP)
32ビット
32ビット
1 0 1 1 0 1 0 ～ 0 0 1 1 0
0 1 1 0 1 1 0 ～ 0 1 1 0 1
平文L(0)
平文R(0)
関数処理
P[f{R(0),K(1)}]
Exclusive
OR
K(1)
0 1 0 1 1 1 0 ～ 48ビット
ｻﾌﾞｷｰ
L(1)=R(0)
L(0) ⊕ P[f{R(0),K(1)}] ≡ R(1)
1 0 0 1 1 0 1 1 0 ～ 1 0
0 1 1 1 0 0 1 0 ～ 0 1
関数処理
P[f{R(1),K(2)}]
Exclusive
OR
K(2)
1 0 1 1 0 0 1 ～ 48ビット
ｻﾌﾞｷｰ
L(2)=R(1)
L(1) ⊕ P[f{R(1),K(2)}] ≡ R(2)
1 0 0 1 1 0 1 1 0 ～ 1 0
0 1 1 1 0 0 1 0 ～ 0 1
・・・・・・・・・・ 途中省略 ・・・・・・・・・・
関数処理
P[f{R(15),K(16)}]
Exclusive
OR
K(16)
1 0 1 1 0 1 1 ～ 48ビット
ｻﾌﾞｷｰ
L(16)=R(15)
L(15) ⊕ P[f{R(15),K(16}] ≡ R(16)
0 0 0 0 1 0 0 1 0 ～ 0 1
0 1 0 0 1 0 0 0 ～ 0 0
最終転置(IP-1)
暗号出力
0 1 0 1 0 1 0 1 0 0 0 1 1 0 0 ～ 1 0 1 0 0 1 0 0 1 1 0
64ビット

Ｌブロックのデータ加算



Ｌブロックデータの加算

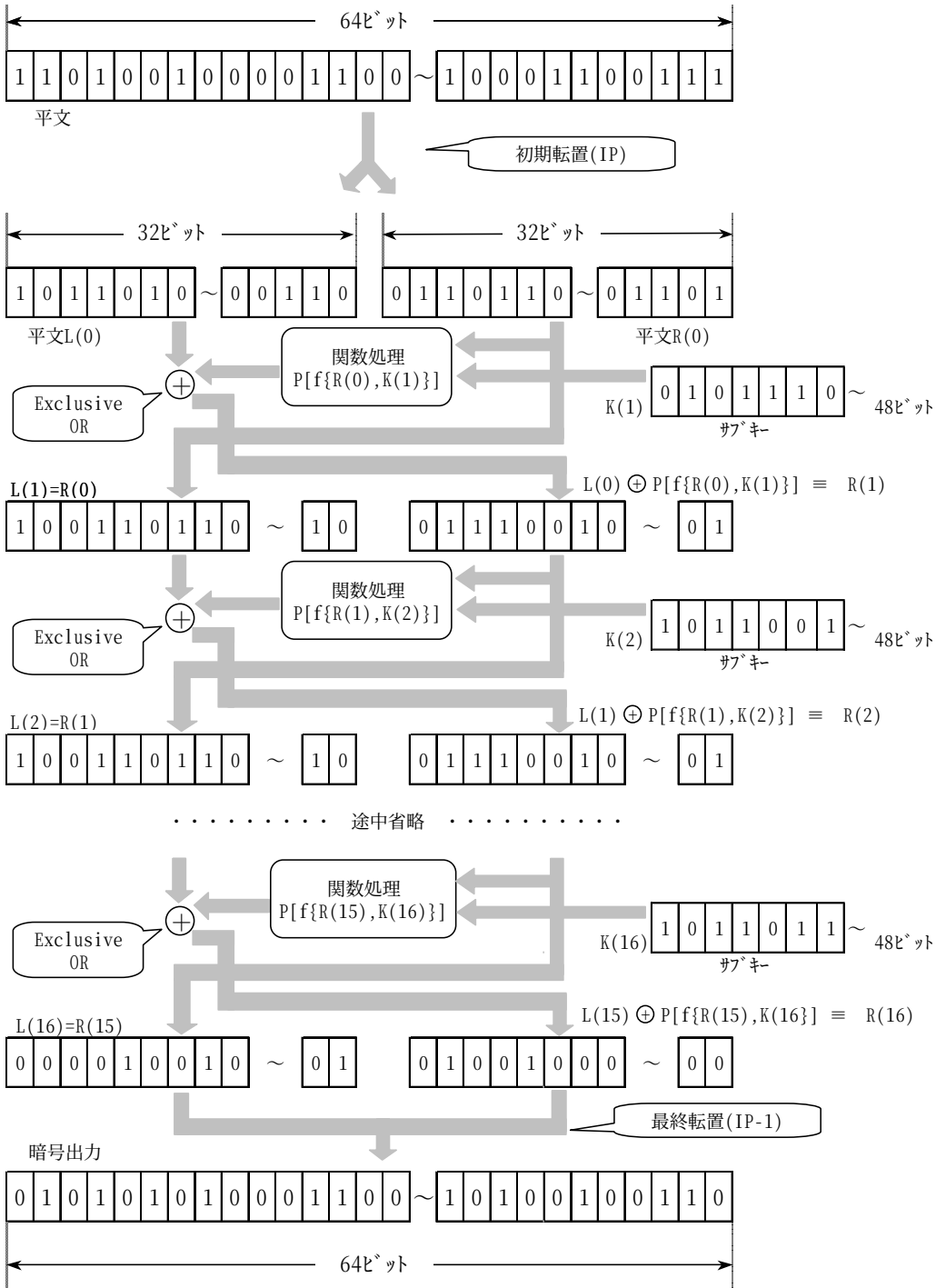
転置表によって転置されたものと
Ｌブロックのデータを
ExclusiveORの論理により加算

ExclusiveOR**の論理では、**

１＋１＝０・・・答え偶数→０

１＋０＝１・・・答え奇数→１

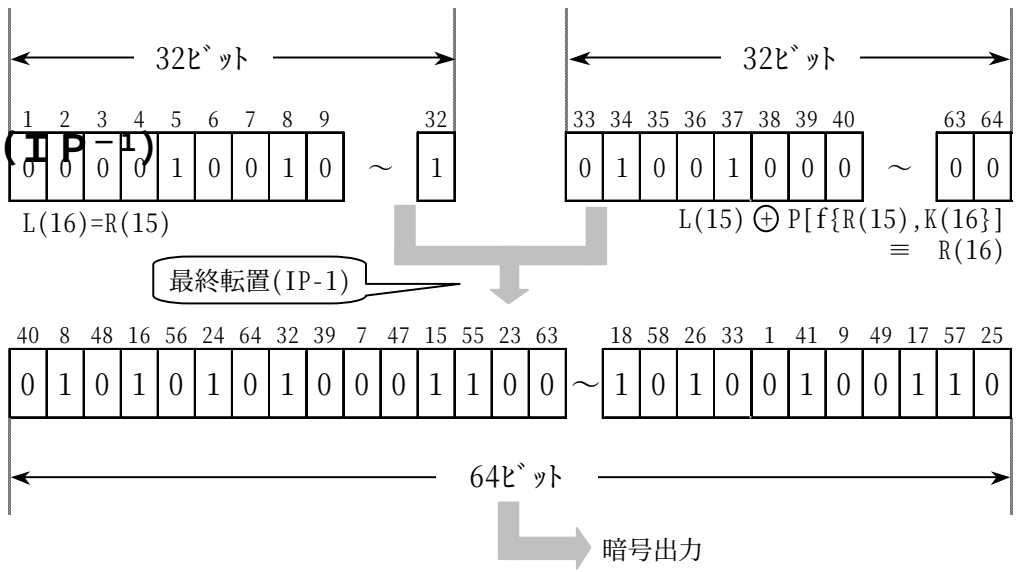
暗号化の全体像


64ビット
1 1 0 1 0 0 1 0 0 0 0 1 1 0 0 ～ 1 0 0 0 1 1 0 0 1 1 1
平文
初期転置(IP)
32ビット
32ビット
1 0 1 1 0 1 0 ～ 0 0 1 1 0
0 1 1 0 1 1 0 ～ 0 1 1 0 1
平文L(0)
平文R(0)
関数処理
P[f{R(0),K(1)}]
Exclusive
OR
K(1)
0 1 0 1 1 1 0 ～ 48ビット
ｻﾌﾞｷｰ
L(1)=R(0)
L(0) ⊕ P[f{R(0),K(1)}] ≡ R(1)
1 0 0 1 1 0 1 1 0 ～ 1 0
0 1 1 1 0 0 1 0 ～ 0 1
関数処理
P[f{R(1),K(2)}]
Exclusive
OR
K(2)
1 0 1 1 0 0 1 ～ 48ビット
ｻﾌﾞｷｰ
L(2)=R(1)
L(1) ⊕ P[f{R(1),K(2)}] ≡ R(2)
1 0 0 1 1 0 1 1 0 ～ 1 0
0 1 1 1 0 0 1 0 ～ 0 1
・・・・・・・・・・ 途中省略 ・・・・・・・・・・
関数処理
P[f{R(15),K(16)}]
Exclusive
OR
K(16)
1 0 1 1 0 1 1 ～ 48ビット
ｻﾌﾞｷｰ
L(16)=R(15)
L(15) ⊕ P[f{R(15),K(16}] ≡ R(16)
0 0 0 0 1 0 0 1 0 ～ 0 1
0 1 0 0 1 0 0 0 ～ 0 0
最終転置(IP-1)
暗号出力
0 1 0 1 0 1 0 1 0 0 0 1 1 0 0 ～ 1 0 1 0 0 1 0 0 1 1 0
64ビット

最終転置

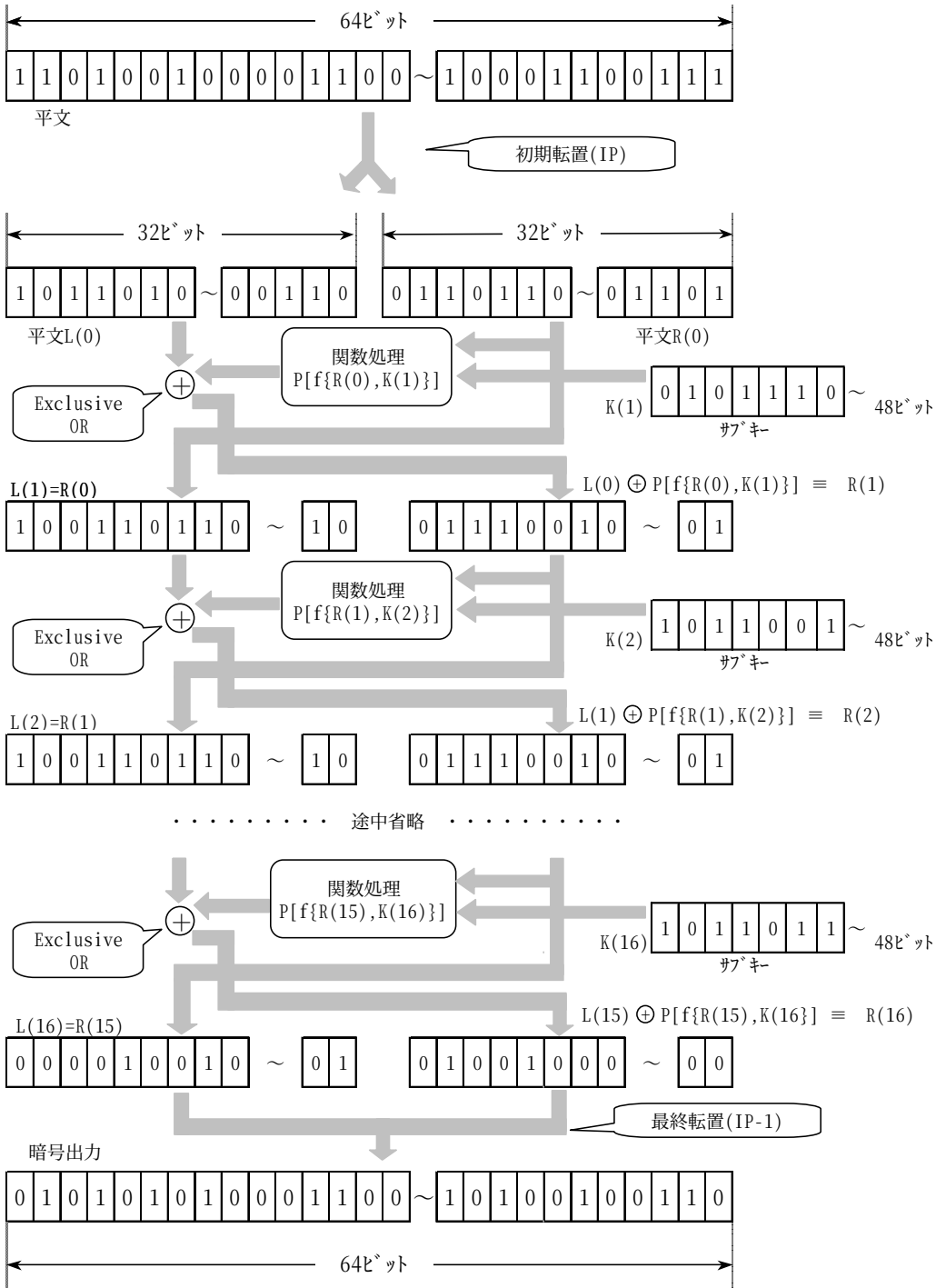
暗号文出力	最終転置入力ﾋﾞｯﾄ							
1-8	40	8	48	16	56	24	64	32
9-16	39	7	47	15	55	23	63	31
17-24	38	6	46	14	54	22	62	30
25-32	37	5	45	13	53	21	61	29
33-40	36	4	44	12	52	20	60	28
41-48	35	3	43	11	51	19	59	27
49-56	34	2	42	10	50	18	58	26
57-64	33	1	41	9	49	17	57	25

最終転置表 (IP⁻¹)



最終転置と暗号出力

暗号化の全体像


64ビット
1 1 0 1 0 0 1 0 0 0 0 1 1 0 0 ～ 1 0 0 0 1 1 0 0 1 1 1
平文
初期転置(IP)
32ビット
32ビット
1 0 1 1 0 1 0 ～ 0 0 1 1 0
0 1 1 0 1 1 0 ～ 0 1 1 0 1
平文L(0)
平文R(0)
関数処理
P[f{R(0),K(1)}]
Exclusive
OR
K(1)
0 1 0 1 1 1 0 ～ 48ビット
ｻﾌﾞｷｰ
L(1)=R(0)
L(0) ⊕ P[f{R(0),K(1)}] ≡ R(1)
1 0 0 1 1 0 1 1 0 ～ 1 0
0 1 1 1 0 0 1 0 ～ 0 1
関数処理
P[f{R(1),K(2)}]
Exclusive
OR
K(2)
1 0 1 1 0 0 1 ～ 48ビット
ｻﾌﾞｷｰ
L(2)=R(1)
L(1) ⊕ P[f{R(1),K(2)}] ≡ R(2)
1 0 0 1 1 0 1 1 0 ～ 1 0
0 1 1 1 0 0 1 0 ～ 0 1
・・・・・・・・・・ 途中省略 ・・・・・・・・・・
関数処理
P[f{R(15),K(16)}]
Exclusive
OR
K(16)
1 0 1 1 0 1 1 ～ 48ビット
ｻﾌﾞｷｰ
L(16)=R(15)
L(15) ⊕ P[f{R(15),K(16}] ≡ R(16)
0 0 0 0 1 0 0 1 0 ～ 0 1
0 1 0 0 1 0 0 0 ～ 0 0
最終転置(IP-1)
暗号出力
0 1 0 1 0 1 0 1 0 0 0 1 1 0 0 ～ 1 0 1 0 0 1 0 0 1 1 0
64ビット

RSA暗号

- 1977年にRivest，Shamir，Adlemanによって実現された世界初の公開鍵暗号
- 素因数分解の困難性とオイラーの定理を利用している
- 特許が2000年に切れているため，この技術を自由に応用できる

整数論の基礎

- 合同式
  - 公開鍵暗号系の基礎になっている
- 可換群
  - RSA暗号の鍵生成, 暗号化, 復号化に関連
- 剰余類
  - RSA暗号の鍵生成, 暗号化, 復号化に関連
- オイラーの定理
  - RSA暗号の暗号化, 復号化で使用

合同式

ある整数a, bと正の整数nに対して,
$a - b$がnの倍数であるとき,
aとbは法nに関して合同であるといい,

$$a \equiv b \pmod{n}$$

と表す．このような式を合同式という．

合同式の例(1)

法5の場合 $12-(-8)=20$ は5の倍数なので,

$$12 \equiv -8 \pmod 5$$

合同式のもう1つの意味

整数a, bを正の整数nで割ったときの商をq_a, q_b，余りをr_a, r_bとすると
次式が成り立つ.

$$a = q_a n + r_a$$

$$b = q_b n + r_b$$

このとき，$r_a = r_b$ならば,

$$a \equiv b \pmod{n}$$

合同式の例(2)

$a = 12,\ b = -8,\ n = 5$ とすると,

$$12 = 2 \times 5 + 2$$
$$-8 = (-2) \times 5 + 2$$

となるので,

$$12 \equiv -8 \pmod 5$$

ある整数aを

正の整数nで割った余りをrとする.

$$\downarrow$$

$$r = a \bmod n$$

$$\downarrow$$

$$a \bmod n = b \bmod n$$

$$\downarrow$$

$$a \equiv b \pmod{n}$$

可換群

ある集合Gとある演算$\star$に対して，
集合G内で演算$\star$とその逆の演算が
自由に行えるとき
$(G, \star)$は可換群である

可換群の例

$$(\mathcal{R}, +), \quad (\mathcal{R}, \times)$$

可換群ではない例

$$(\mathcal{N}, +), \quad (\mathcal{N}, \times)$$

剰余類

ある正の整数$n \geq 1$によって決定される集合$\{0, 1, \cdots, n-1\}$を

$$\mathcal{Z}_n = \{0, 1, \cdots, n-1\}$$

と表す．これを剰余類全体の集合という．

剰余類の加算と乗算

任意の$a, b \in \mathbb{Z}_n$に対して，
加算，乗算を次のように定義する．

加算 $(a + b) \bmod n$

乗算 $(a \times b) \bmod n$

加算の例

$\mathbb{Z}_4$の加算表

$+$	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

乗算の例

$\mathbb{Z}_4$の乗算表

$\times$	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	0	2
3	0	3	2	1

$(\mathbb{Z}_n, +)$は可換群になる.
しかし，$(\mathbb{Z}_n, \times)$はnの値によって
逆演算が不可能となり，
可換群にならないことがある.

逆演算とは？

$a \star b$の逆演算が行えるとき

bの逆元が存在する

$(\mathcal{R}, \times)$ の場合を考える.

乗算　　$3 \times 4 = 12$

逆演算　　$3 \div 4 = \frac{3}{4}$

↓

4 の逆元は $\frac{1}{4}$ である.

$(\mathcal{R}, \times)$の場合，

一般に，$a, b \in \mathcal{R}$に対して，

bがaの逆元であるとき，

$$a \times b = 1$$

が成り立つ．

$(\mathbb{Z}_n, \times)$の場合，
ある$a, b \in \mathbb{Z}_n$にたいして，
bがaの乗法の逆元であるとき，
$ab \equiv 1 \pmod{n}$が成り立つ．

乗法の逆元の例

$\mathbb{Z}_4$ の場合

x	1	2	3
x^{-1}	1	−	3

$\mathbb{Z}_7$ の場合

x	1	2	3	4	5	6
x^{-1}	1	4	5	2	3	6

法nの乗法の逆元の存在

$\mathbb{Z}_n$の元xに対して,

$\mathsf{gcd}(x, n) = 1$

が成り立つとき，xは乗法の逆元を持つ.

乗法の逆元は
拡張されたユークリッドの互除法で
簡単に求められる.

既約剰余類

$\mathbb{Z}_n$の元xで,
$\gcd(x, n) = 1$をみたすものの集合を
$\mathbb{Z}_n^*$と表し,
既約剰余類全体の集合という.

既約剰余類の例

法が９の場合

$$\mathbb{Z}_9 = \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$$

$$\mathbb{Z}_9^* = \{1, 2, 4, 5, 7, 8\}$$

既約剰余類の元の数

$\mathbb{Z}_n^*$の元の数を$\phi(n)$と表す.
$\phi(n)$はオイラーの関数と呼ばれ,
nが互いに異なる素数p, qの積であるとき
$\phi(n) = (p-1)(q-1)$ が成り立つ.

オイラーの定理

任意の$x \in \mathbb{Z}_n^*$に対して

$$x^{\phi(n)} \equiv 1 \pmod{n}$$

が成立する

オイラーの定理の変形(1)

kを任意の正の整数とすると，任意の$x \in \mathbb{Z}_n^*$に対して，

$$x^{k\phi(n)+1} \equiv x \pmod{n}$$

が成り立つ.

オイラーの定理の変形(2)

$$ab = k\phi(n) + 1$$

$$\downarrow$$

$$ab - 1 = k\phi(n)$$

$$\downarrow$$

$$ab \equiv 1 \pmod{\phi(n)}$$

オイラーの定理の変形(3)

任意の $x \in \mathbb{Z}_n^*$ に対して，

$$x^{ab} \equiv x \pmod{n}$$

が成り立つ．ただし，$ab \equiv 1 \pmod{\phi(n)}$

オイラーの定理の特別な場合

nが互いに異なる素数p, qの積であるとき
任意の$x \in \mathbb{Z}_n$に対して,

$$x^{ab} \equiv x \pmod{n}$$

が成り立つ．ただし，$ab \equiv 1 \pmod{\phi(n)}$

数学のまとめ(1)

ある整数aを

正の整数nで割った余りをrとする.

↓

$$r = a \bmod n$$

数学のまとめ(2)

$\mathcal{Z}_n^*$の任意の元xは,

- $\gcd(x, n) = 1$を満たす.
- $xy \equiv 1 \pmod{n}$を満たす$y \in \mathcal{Z}_n^*$が存在する.

数学のまとめ(3)

$\mathbb{Z}_n^*$の元の個数は$\phi(n)$と表す.

nが互いに異なる素数p, qの積なら,

$\phi(n) = (p-1)(q-1)$

数学のまとめ(4)

nが互いに異なる素数p, qの積であるとき
任意の$x \in \mathbb{Z}_n$に対して,

$$x^{ab} \equiv x \pmod{n}$$

が成り立つ．ただし，$ab \equiv 1 \pmod{\phi(n)}$

鍵の生成(1)

利用者Ａは次のようにして，
公開鍵と秘密鍵をつくる．

１．互いに異なる素数p, qを選ぶ

２．$n = pq$を求める

３．$\phi(n) = (p-1)(q-1)$を求める

鍵の生成(2)

４．$\phi(n)$ と互いに素である数 e を選ぶ

５．$ed \equiv 1 \pmod{\phi(n)}$ となる
　d を求める

以上から，公開鍵 e, n，秘密鍵 d を得る．
また，$p, q, \phi(n)$ も公開してはならない．

暗号化

利用者Bは平文xをAに送る場合
次のようにして暗号化する.

平文$x(0 \leq x < n)$に対して,

$$y = x^e \bmod n$$

となる暗号文yを求める

復号化

利用者Aは暗号文yを
次のようにして復号化する.

暗号文yに対して

$$x = y^d \bmod n$$
$$= x^{ed} \bmod n$$

となる平文xを求める

RSA暗号の用途(1)

- **認証**
  - **公開鍵を作った人以外が, 対になる秘密鍵を作ることができないので, これによって本人であることを確認することができる.**

RSA暗号の用途(2)

- **秘密鍵暗号の秘密鍵の配送**
  - RSA暗号では安全のためにpとqをそれぞれ100桁程度に選ばなければならない．そのために鍵の長さが長くなり，暗号化，復号化に時間がかかる．RSAとDESを比較するとほぼ1000倍の時間がかかる．そこで，RSA暗号で秘密鍵暗号の秘密鍵を暗号化して送り，実際の平文は秘密鍵暗号で暗号化するといった方式がとられる．

RSA暗号の安全性

i. 暗号文を解読するためには
　秘密鍵 d を知る必要がある
ii. d を知るためには $\phi(n)$ が必要
iii. $\phi(n)$ は p, q から
　求めなければならない

nからp, qを求めることを考えられるが,
実際のRSA暗号ではnは非常に大きな数になり,
素因数分解することは難しい.
つまり，p, qからnを求めることはやさしいが,
nからp, qを求めることは難しいという
一方向性によってRSAの安全性は保障される.

- **実際**
  - RSA**暗号を解読は素因数分解することと同程度難しいらしい，としかいえない**
- **理由**
  - **秘密鍵**d**を求めずに暗号文**y**から平文**x**を得る簡単な方法が存在しないことも，素因数分解** n=pq**を知らずに**d**を求める方法が存在しないことも証明されていない**

楕円曲線暗号方式

- 1985年にKoblitz氏とMiller氏がほぼ同時に考案した公開鍵型の暗号方式。
- 楕円曲線と呼ばれる数式によって定義される特殊な加算法に基づいて暗号化、復号を行う方式。
- 解読の困難さは楕円曲線上の離散対数問題を解くのと同程度と言われる。

整数論（体）

（F、＋、＊）

1．Fは＋に関しては可換群となる。

$$\forall\ a,b \in F,\quad a+b=b+a$$

2．Fは＊に関して＋の単位元以外の要素に対して可換群となる。

3．＋と＊に関して分配率を満たす。

$$a*(b+c)=a*b+a*c$$

整数論 (有限体と原始元)

- 体の集合が有限であるとき、有限体と言う。
- 整数の集合Z_pに対してpが素数であるとき、体になる。

$$Z_7 = \{0, 1, 2, 3, 4, 5, 6\}$$

- 有限体Z_pについて$Z_p^* = Z_p \setminus \{0\}$ は乗法群として巡回群である。

$$\exists\, \alpha \in Z_p^*;\ Z_p^* = \{1, \alpha, \alpha^2, \ldots, \alpha^{p-2}\}$$

$$Z_7^* = \{1, 2, 3, 4, 5, 6\} = \{3^6, 3^2, 3^1, 3^4, 3^5, 3^3\}$$

- αはZ_pの原始元と呼ばれる。

有限体 (Z_P) 上の楕円曲線

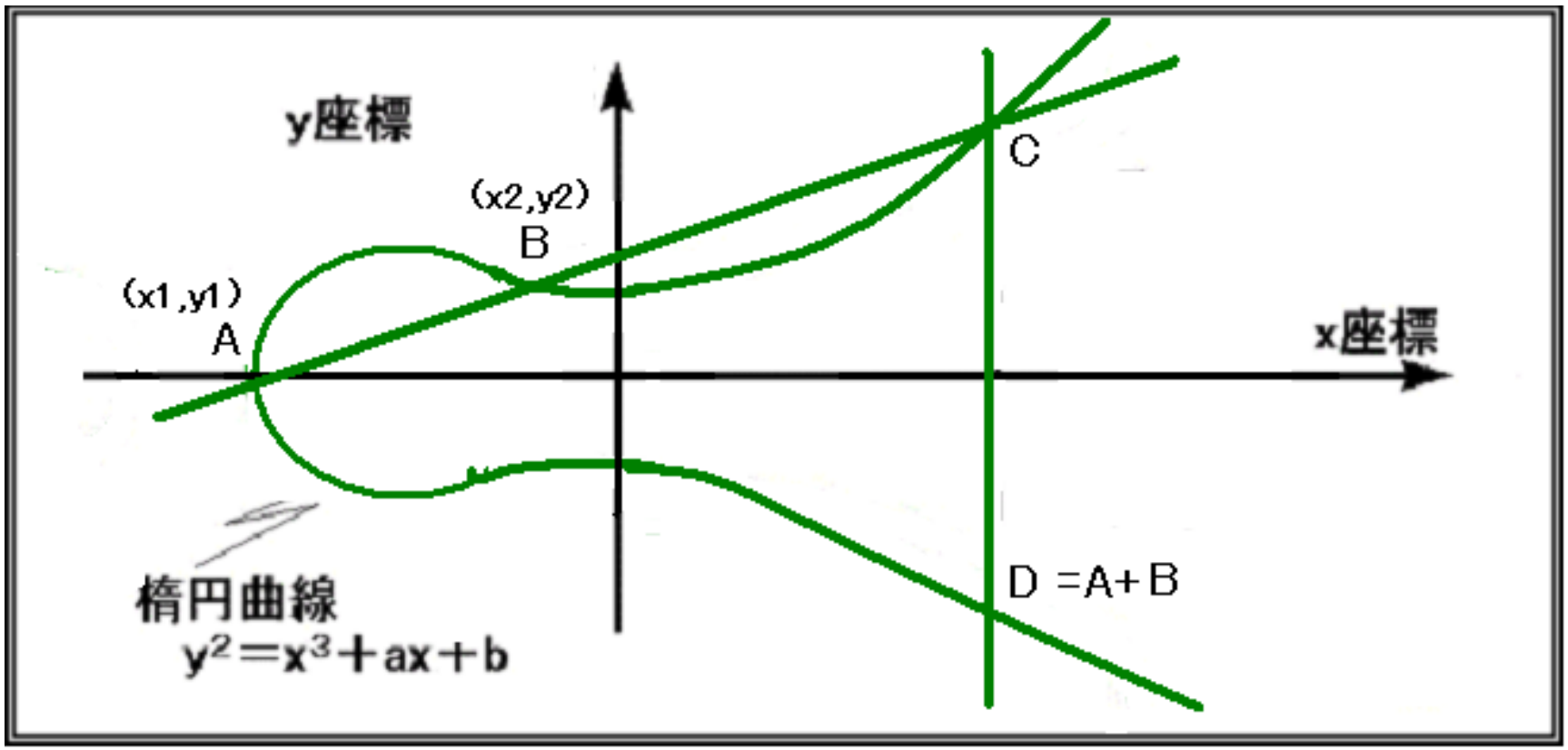
- P (＞3の素数)、$y^2 \equiv x^3+ax+b \pmod p$
- $a,b \in Z_P$, $4a^3+27b^2 \neq 0 \pmod p$
- 点集合は

$$(x、y) \in Z_P * Z_P$$

無限遠点：O

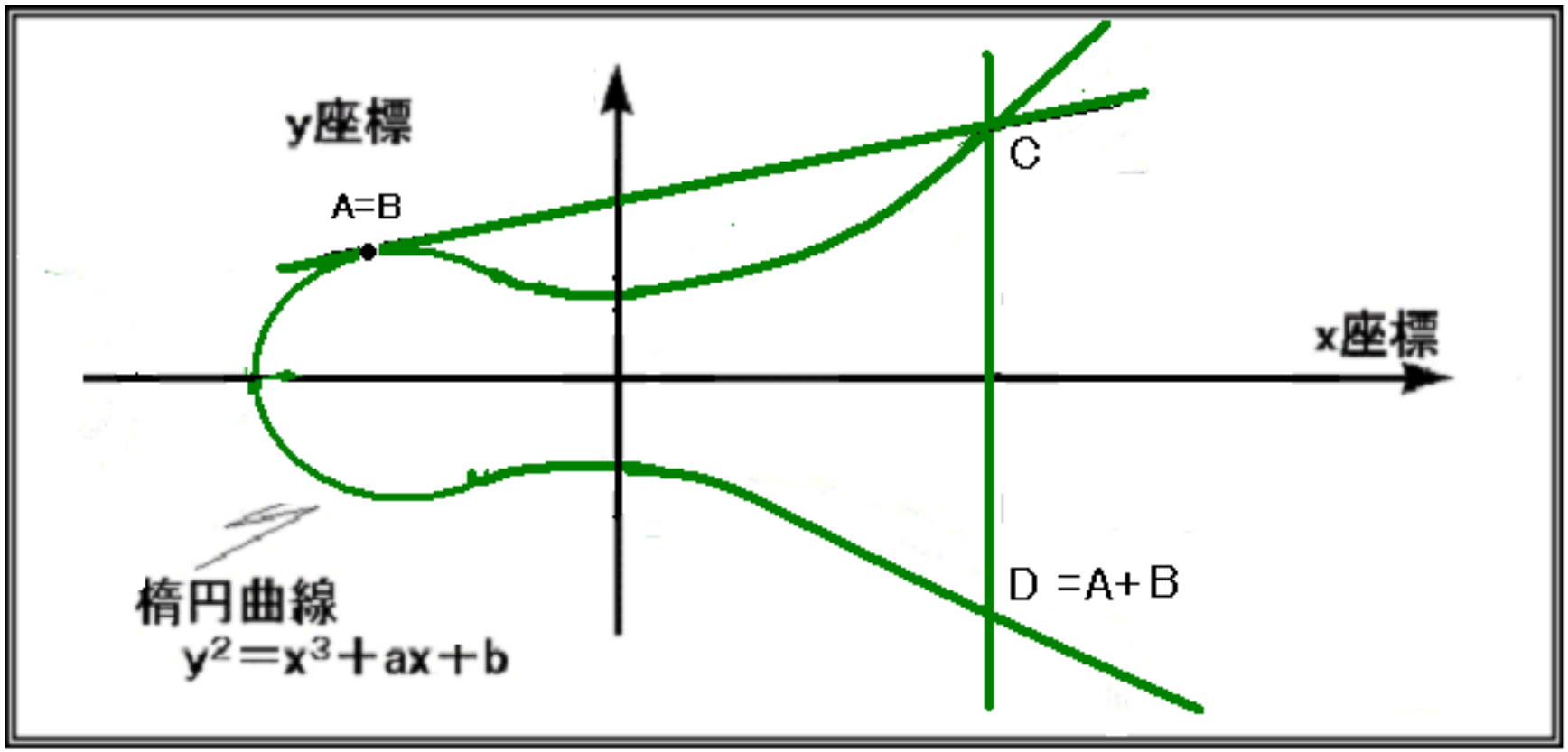
楕円曲線上の加算 (1)

$X_1 \neq X_2$



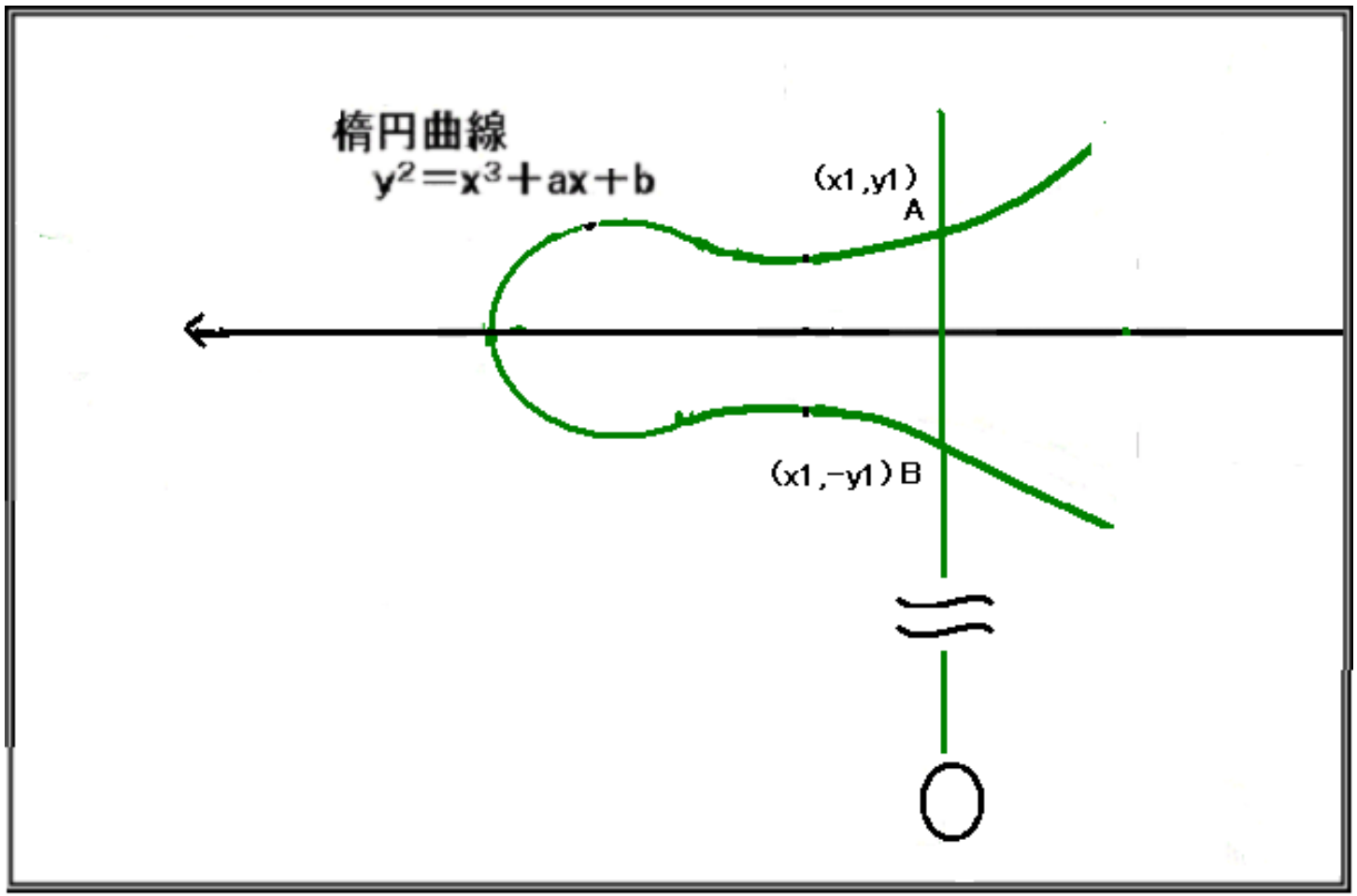
楕円曲線上の加算 (2)

A=B



楕円曲線上の加算 (3)

B=－A



$(\{E、O\}, +)$ は可換群となる

1. $P+Q=R \in E$
2. $\forall P \in E$ 単位元：O

   $P+O=O+P=P$
3. $\forall P \in E, \exists P' \in E,$

   S.T $P+P'=P'+P=O$
4. $\forall P,Q,R \in E$ $P+(Q+R)=(P+Q)+R$
5. $\forall P,Q \in E$ $P+Q=Q+P$

- **E:** $y^2 \equiv x^3+ax+b \pmod p$
- **P=(x_1,y_1), Q=(x_2,y_2), P,Q ∈{E,O}**
- **P+Q=(x_3,y_3) とする**
- **$x_3=\lambda^2-x_1-x_2$ $y_3=\lambda(x_1-x_3)-y_1$**
- **$\lambda=(y_2-y_1)/(x_2-x_1)$, if $P \neq Q$**
- **$\lambda=(3x_1^2+a)/2y_1$, if P = Q**

楕円曲線上の離散対数問題

- 素数pを定めたとき楕円曲線Eにおける原始元αとβが与えられたときに対して、$a\alpha = \beta$ であるような唯一の整数a を求めること。

- Pをうまく選べば、この問題は難しいと考えられる。

Example

- $Y^2 = x^3 + x + 1$ Z_5
- $\alpha = (0, 1)$
- $2\ \alpha = (4, 2)$
- $a\ \alpha = (2, 1)$ a=?

楕円曲線暗号系

- 受信者Bは予め暗号化鍵を生成して公開して、送信者Aはその公開鍵を用いて平文Mを暗号化し、その暗号文を受け取ったBは自分の秘密鍵で復号化する。

暗号化鍵の生成

- 受信者BはE上の点α（α∈E）を選び、
- β＝aα　を計算し、
- P、α、βを暗号化鍵として公開し、
- aは秘密鍵として保持しておく。

暗号化

- 平文Mを楕円曲線上の点xに対応させ、
- 送信者Aは乱数kを発生させ、

$$y_1 = k\alpha$$

$$y_2 = x + k\beta$$

を計算し

- (y_1, y_2)を暗号文としてBに送る。

復号

- 受信者Bは秘密鍵aを用いて、
- $D_K(Y)=y_2-ay_1$　を計算し、Xを得る。
- XをMに対応させ、平文Mに戻る。

復号

- $D_K(Y)=y_2-ay_1=x+k\beta-ak\alpha$

  $= x+ka\alpha-ak\alpha$

  $= x$

安全性について

- 公開鍵P、 α、 βを用いて外部者Cが
  (y_1, y_2)を平文Mに復号するためには、 βからaを求めるか($\beta = a\alpha$)、もしくはy2から乱数kを求めるか($y_2 = x + k\beta$)をしなければならないが、いずれも楕円曲線上の離散対数問題を解かなければならないので、復号化は事実上できない。

楕円曲線暗号とRSA暗号

- 1024ビットの鍵を使うRSA暗号と同程度の安全性を、楕円曲線暗号では160ビットで実現することができる。
- 暗号化と復号はRSA暗号に比べて約10倍高速である。

109ビットの楕円曲線暗号が解読された

- 40カ国1300人
- 9500台のコンピュータ
  (Unix 2/3,Windows 1/3)
- 4ヶ月間
- RSAの６００ビット以上に相当
- 単体４５０MH_Zマシンの場合：５００年以上

楕円曲線暗号系の魅力

- 短い鍵で高い安全性が確保でき、計算も高速に行うことができる。
- 普段利用しているパソコンでも十分高速に動作する。

結論

- 共通鍵暗号系のスピードが速いが、鍵を安全に相手に送る手段がない。
- 公開鍵暗号系の安全性が高いが、スピードが遅い。
- 公開鍵暗号系は共通鍵の配送に使う、平文の暗号化と復号は共通鍵暗号系で。
- 安全性の要求が高い、記憶容量が少ないのは　楕円曲線暗号系で(Master cardなど)。