fgetc, fputc 関数の使い方イメージ

情報通信工学実験B / 電子情報学実験B

電気通信大学 情報理工学域　II類　情報通信工学プログラム/電子情報学プログラム

注意する点: ファイルの入出力を行う fgetc, fputc 関数はデータを 1 バイト単位でしか扱えない

入力ファイル（ひら文）

W	E	W	I	L	L	M	E	E	T	A	T	C	H	O	F	U	S	T	A	T	I	O	N

fgetc　　fgetc

CPU内部

ひら文の10進数表現

87	69	87	73	76	76	77	69	69	84	65	84	67	72	79	70	85	83	84	65	84	73	79	78

RSA暗号化　鍵　$(e,n) = (5,323)$

$C = M^e \ (mod\ n) = M^5 \ (mod\ 323)$

288を表現するには2バイトが必要

1バイトで表現できる

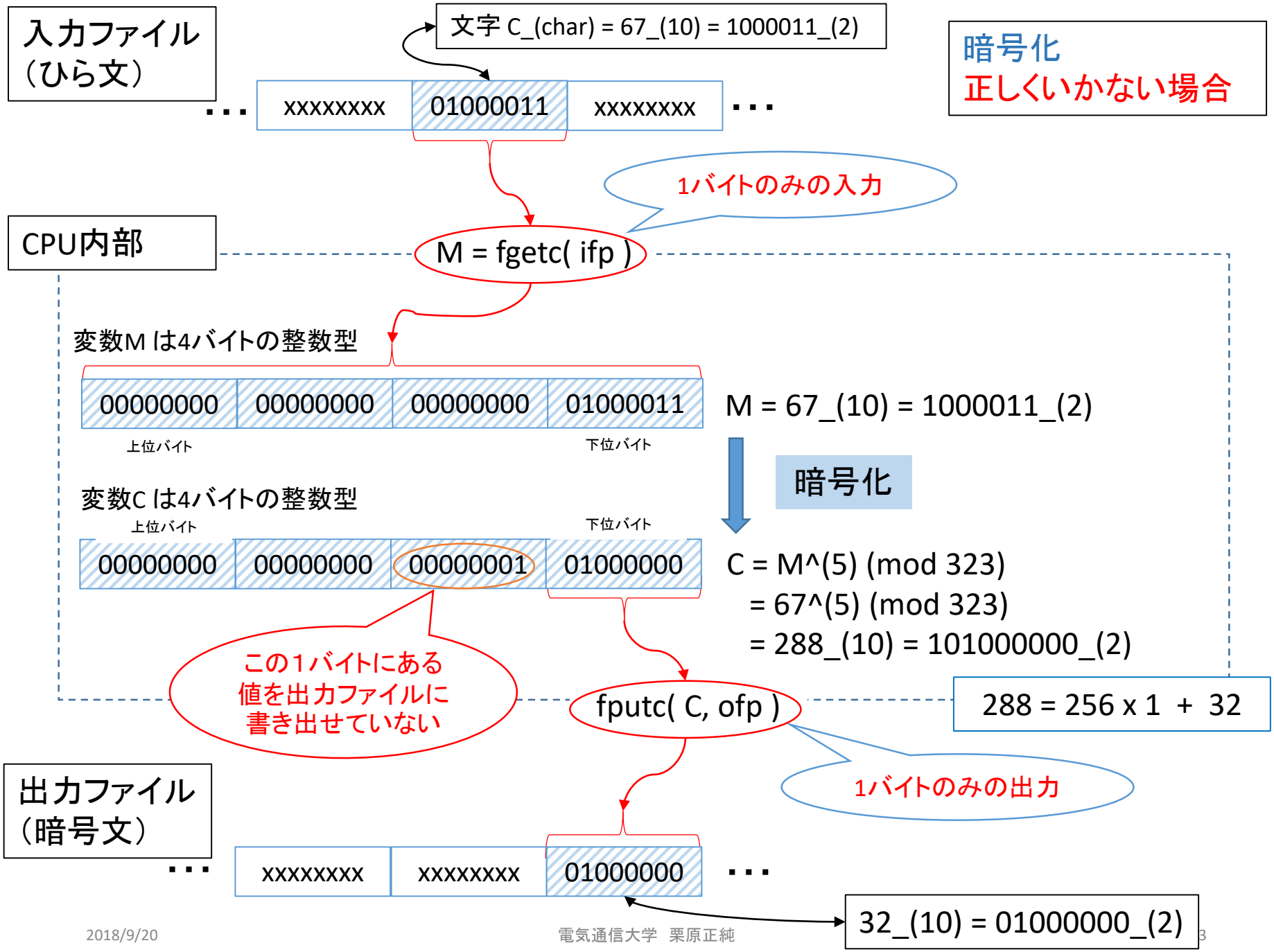
暗号文の10進表現

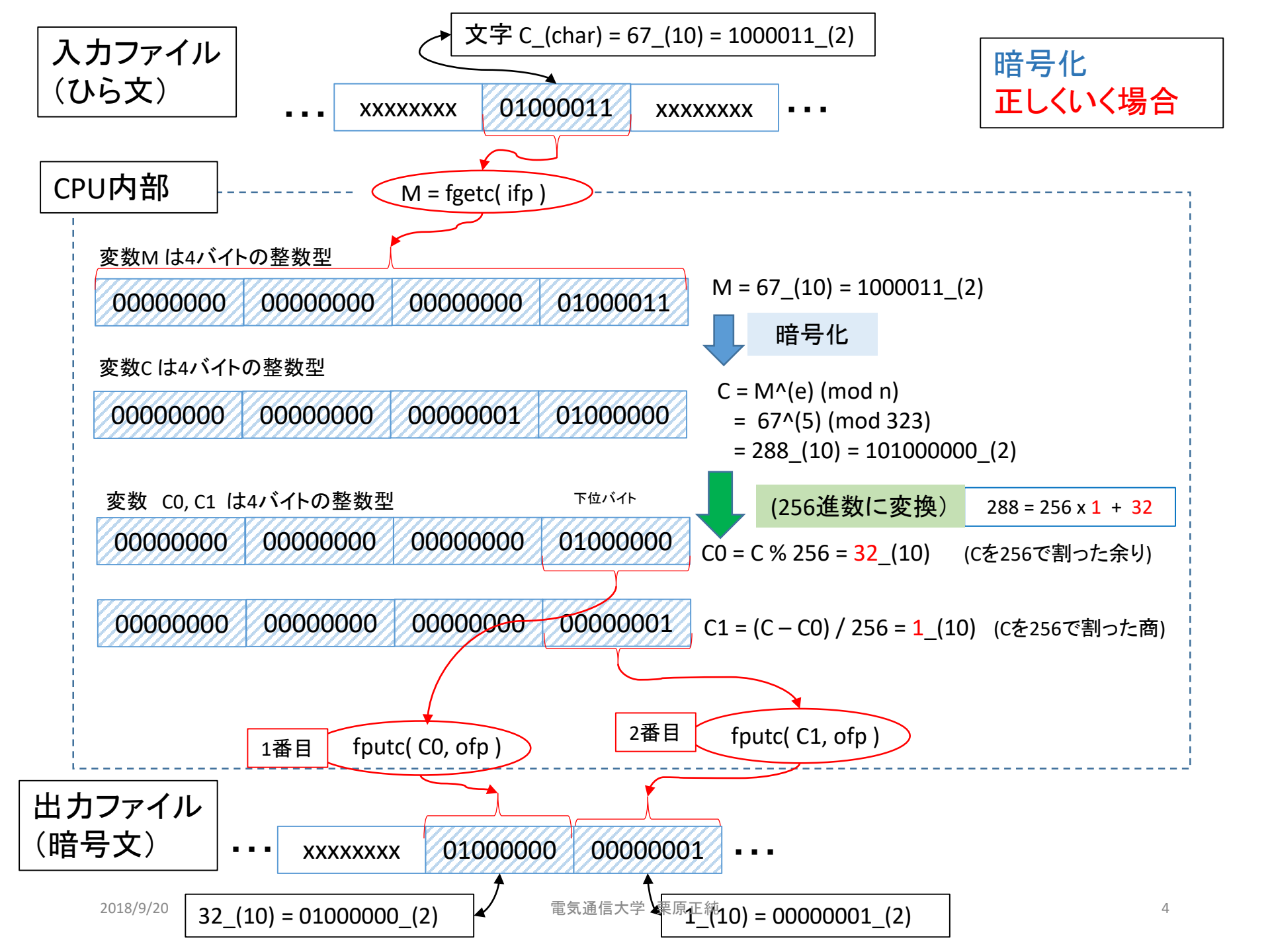
83	103	83	99	247	247	229	103	103	50	12	50	288	21	129	185	187	87	59	12	50	99	129	108

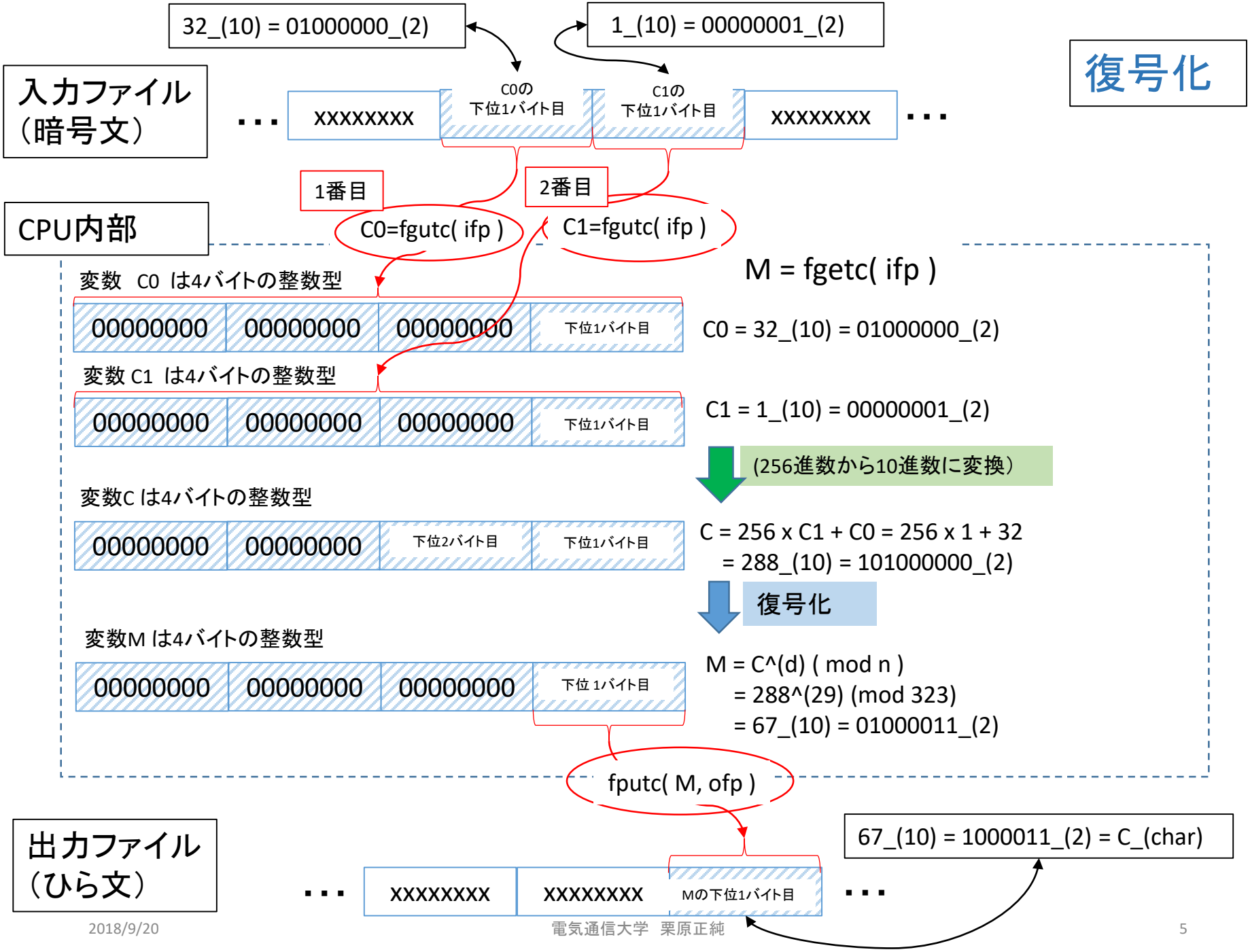
fputc　　fputc

出力ファイル（暗号文）

どのように暗号文をファイルに保存すればよいか？


入力ファイル
（ひら文）
文字 C_(char) = 67_(10) = 1000011_(2)
暗号化
正しくいかない場合
xxxxxxxx
01000011
xxxxxxxx
1バイトのみの入力
CPU内部
M = fgetc(ifp)
変数M は4バイトの整数型
00000000
00000000
00000000
01000011
上位バイト
下位バイト
M = 67_(10) = 1000011_(2)
暗号化
変数C は4バイトの整数型
上位バイト
下位バイト
00000000
00000000
00000001
01000000
C = M^(5) (mod 323)
= 67^(5) (mod 323)
= 288_(10) = 101000000_(2)
この1バイトにある
値を出力ファイルに
書き出せていない
fputc(C, ofp)
288 = 256 x 1 + 32
1バイトのみの出力
出力ファイル
（暗号文）
xxxxxxxx
xxxxxxxx
01000000
32_(10) = 01000000_(2)


入力ファイル
（ひら文）
文字 C_(char) = 67_(10) = 1000011_(2)
xxxxxxxx
01000011
xxxxxxxx
暗号化
正しくいく場合
CPU内部
M = fgetc(ifp)
変数M は4バイトの整数型
00000000
00000000
00000000
01000011
M = 67_(10) = 1000011_(2)
暗号化
変数C は4バイトの整数型
00000000
00000000
00000001
01000000
C = M^(e) (mod n)
= 67^(5) (mod 323)
= 288_(10) = 101000000_(2)
変数 C0, C1 は4バイトの整数型
下位バイト
00000000
00000000
00000000
01000000
(256進数に変換)
288 = 256 x 1 + 32
C0 = C % 256 = 32_(10) (Cを256で割った余り)
00000000
00000000
00000000
00000001
C1 = (C – C0) / 256 = 1_(10) (Cを256で割った商)
1番目
fputc(C0, ofp)
2番目
fputc(C1, ofp)
出力ファイル
（暗号文）
xxxxxxxx
01000000
00000001
32_(10) = 01000000_(2)
1_(10) = 00000001_(2)


32_(10) = 01000000_(2)
1_(10) = 00000001_(2)
復号化
入力ファイル
（暗号文）
XXXXXXXX
C0の
下位1バイト目
C1の
下位1バイト目
XXXXXXXX
1番目
2番目
CPU内部
C0=fgutc(ifp)
C1=fgutc(ifp)
M = fgetc(ifp)
変数 C0 は4バイトの整数型
00000000
00000000
00000000
下位1バイト目
C0 = 32_(10) = 01000000_(2)
変数 C1 は4バイトの整数型
00000000
00000000
00000000
下位1バイト目
C1 = 1_(10) = 00000001_(2)
(256進数から10進数に変換)
変数C は4バイトの整数型
00000000
00000000
下位2バイト目
下位1バイト目
C = 256 x C1 + C0 = 256 x 1 + 32
= 288_(10) = 101000000_(2)
復号化
変数M は4バイトの整数型
00000000
00000000
00000000
下位 1バイト目
M = C^(d) (mod n)
= 288^(29) (mod 323)
= 67_(10) = 01000011_(2)
fputc(M, ofp)
出力ファイル
（ひら文）
XXXXXXXX
XXXXXXXX
Mの下位1バイト目
67_(10) = 1000011_(2) = C_(char)

プログラミングのイメージ

暗号化

```
while( (m = fgetc(ifp) ) != EOF ){
   c = cal(m,e,n);                    /* c = m^e ( mod n )   暗号化の計算*/
   c0 = c % 256;
   c1 = (c - c0) / 256;
   fputc(c0, ofp);
   fputc(c1, ofp);
 }
```

復号化

```
while( ( (c0 = fgetc(ifp) ) != EOF) && ( (c1 = fgetc(ifp) ) != EOF) ){
   c = c1*256 + c0;
   m = cal(c,d,n) ;                   /* m = c^d ( mod n ) 復号化の計算*/
   fputc(m, ofp);
 }
```

C言語で int型を利用するときの注意

- C言語では、割り算を表す演算子の前後の数値が int型の整数の場合、計算結果も整数を返す。
- 具体的に、整数 10 と 3 に対し、一般には、10/3 は 3.333… であるが、C言語の場合、計算結果は、3 となってしまう。
- 前頁において、int型の変数 c1 の値を求めるために、c1=(c-c0)/256 と計算していた。しかし、暗号化後の c は　int型の整数であることより、c1=c/256 としても c1=(c-c0)/256 と同じ計算結果を得ることができる。本当にそのようになるか確認せよ。
- 意図しないところで、計算結果が不正確になるので、c1=c/256 という演算の手法は使わない方がよいと思う。
- 整数 10 と実数 3.0 に対し、10/3.0 とした場合の計算結果は、3とはならない。確認せよ。