ファイルを コード (16進数) で表示するには？

(ファイルは、1byteの数の列)

dump.c

```
#include <stdio.h>
int main(){
    int c;
    c=getchar();
    while(c!=EOF){
        printf("%x ",c);
        c=getchar();
    }
}
```

16進数で表示

%d 10進数で

```
while((c=getchar())
    !=EOF){
printf("%x ",c)
}
```

```
cc dump.c ↵
./a.out < ファイル名
```

```
cc ␣ dump.c ␣ -o ␣ dump
./dump < ファイル名
```

a.outでなくて、dumpが作られる。

```
ls ␣ -l
```

option

```
ls ␣ -l ␣ *.c
```

← .cでおわるファイル

*は任意の文字列

R(統計)用には %d で10進表示

./dump < c-text1.txt > t.txt.

t.txtをRが読めるように.

a ← c(35, 27, ……)

とする。

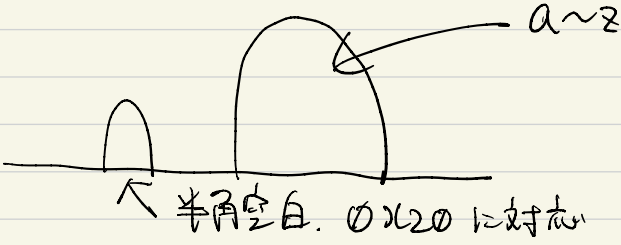
右辺を左辺に代入.

ベクトル (35, 27, …) をつくる.

hist(a)　aのヒストグラムをえがく

q() ⏎　Saveと聞かれたら n.　(No)

終了.



アスキーコード表 検索.

② 256とおり全部ためす. のもあり.

§2. RSA 暗号

G：有限アーベル群.

m：位数. つまり #G.

例. $(\mathbb{Z}/5\mathbb{Z})^\times = \{1,2,3,4\}$

	1	2	3	4
1	1	2	3	4
2	2	4	1	3
3	3	1	4	2
4	4	3	2	1

定理 $\forall a \in G$, $a^m = e$ ← 単位元.

(i) $G = \{a_1, \cdots, a_m\}$

$G' = \{aa_1, aa_2, \cdots, aa_m\}$

$G = G'$ となる.

(ii) $i \neq j \Rightarrow aa_i \neq aa_j$

$aa_i = aa_j$ に、a^{-1} をかけると、$a_i = a_j$ ∴ $i = j$ //

$G = G'$ かつ可換なので.

$$a_1 \cdots a_m = (aa_1)(aa_2)\cdots(aa_m)$$

$$= a^m (a_1 \cdots a_m)$$

$(a_1 \cdots a_m)^{-1}$ をかけると

$$e = a^m \quad //$$

例. p: 素数 $(\mathbb{Z}/p\mathbb{Z})^\times = \{1, 2, \cdots, p-1\} = G$

有限アーベル群.

$\forall a \in \mathbb{Z} \setminus p\mathbb{Z}$ に対して

$$a^{p-1} \equiv 1 \mod p$$

フェルマーの小定理

RSA暗号とは.

5 p_1, p_2 7 異なる素数.

2進で2048ケタ以上位.

36 $n = p_1 p_2$, 13 e は $(p_1-1)(p_2-1)$ と素な数.

↑ 公開 ↑ 公開

$ed \equiv 1 \mod (p_1-1)(p_2-1)$ となる d を求める。

つまり 不定方程式

$ed + y(p_1-1)(p_2-1) = 1$ と解く。

$d \cdot 13 + y \cdot 24 = 1$

d をひみつかぎ。

公開しないのは、p_1, p_2 と d。

公開は、n と e ← 公開かぎ。

◎ 大きい数の素因数分解は、時間がかかる。

∴ n から、p_1, p_2 を求めるのは "できない"。

暗号化 $0 \le A < n$. A を暗号化するには、

$B = A^e \% n$ 公開かぎ e で e 乗して、mod n $\%n$ はnで わった余り。

複号化 $C = B^d \% n$ ひみつかぎ d で d 乗して mod n。

Prop. $C = A$.

$C \equiv (A^e)^d \mod n$.

ここで、$ed + y(p_1-1)(p_2-1) = 1$ を使うと、

$A^{ed} = A\, A^{(p_1-1)(p_2-1)(-y)}$

$\equiv 1$

Thより、 $A^{p_1-1} \equiv 1 \mod p_1$

$A^{p_2-1} \equiv 1 \mod p_2$

$\therefore A^{(p_1-1)(p_2-1)(-y)} \equiv 1 \mod p_1$

$\equiv 1 \mod p_2$

$A^{(p_1-1)(p_2-1)(-y)} - 1$ は p_1 でも p_2 でもわりきれる。

p_1, p_2 は異なる素数なので、これは n でわりきれる。

$\therefore A^{(p_1-1)(p_2-1)(-y)} \equiv 1 \mod n$ //

数十, 数百バイトを、まとめて A として、RSA を使う。

例. abc → コード → 0x61, 0x62, 0x63

↓

0x616263 ← これを A に。

② 電子署名にも使える。 私の公開かぎを、信頼できる方法ではいふ。

文例 "takayama 2022.04.21" を、私のひみつかぎで暗号化して、ファイルに署名として追加。 ← ファイルのサイズ、....

② 本人確認にも使える。

"まようはくもり"(ひみつのことば)を、私の公開かぎで暗号化して私におくる。

私は、ひみつかぎで、復号化して、いまおっしゃったのは "まようはくもり" ですね。と答える。

例. https

webページが本物だ、を、上の本人確認のしくみで、保証.

公開かぎ等は、ぴりせんなどの、会社が配布.

---

gpg␣--gen-key　公開かぎ ひみつかぎ 生成.

- に、短い形式
-- 長い形式のoption

◎ オレオレ証明なので、メールのアドレスも本物でなくてよい.

gpg --list-keys. 自分のもってる公開かぎのリストを表示.

gpg --list-secret-keys. ひみつかぎのリスト

かぎは ~~フォルダ~~ .gnupg の下にあり. ls -a

gpg␣-r␣メールのアドレス␣-ea␣test.jpg　暗号化したいファイル名.

test.jpg.asc が生成される.

emacs␣test.jpg.asc ⓔ で中身をみる.

gpg␣test.jpg.asc　復号化.

test.jpg が生成される.

ひみつかぎ、公開かぎ を export. (他のコピー用)

import する仕組みあり ← 自分のシステムに登録.

◎ ひみつかぎ なくしたら、どうしようもない.

課題　1. gpgを使ってみる.

2. レポート1をやりはじめる.

(3. 月の準備.)　　11:13. breakout room.

補足.(負べき)

$a^m = e$ なら、

$a^{-1} = a^{m-1}$

よって $(a^{-1})^n = (a^{m-1})^n = a^{(m-1)n}$

終了です.

質問ある人は、音声ONでまいて下さい.