

システムプログラム

第12回

創域理工学部 情報計算科学科

松澤 智史

本日の内容

- パスワード認証
- 電子署名

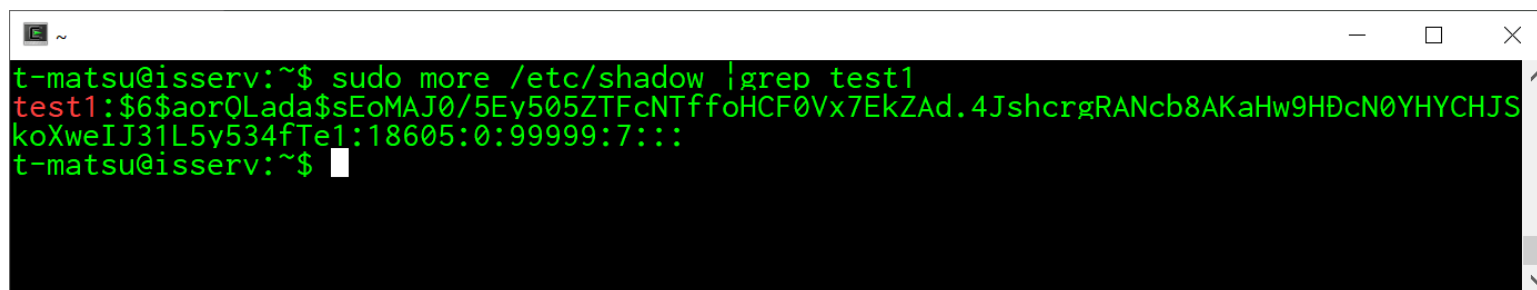
ハッシュ

- 用語
 - キー, ハッシュ関数, ハッシュ値
- 特徴
 - 同じキーから生成されたハッシュ値は必ず同じになる
 - 同じハッシュ値を生成したキーは必ずしも同じとは限らない
 - ハッシュ値からキーを推測するのは困難である



Linuxのパスワード管理

- /etc/passwd
 - ユーザID, グループID, ホーム, パスワード, 使用シェル(デフォルト)
 - パスワード項目がxとなっている場合は以下の/etc/shadowに別記
- /etc/shadow
 - パスワードがハッシュ値として記載
 - 管理者権限でのみ閲覧可能なことが多い



```
t-matsu@isserv:~$ sudo more /etc/shadow |grep test1
test1:$6$aorOLada$sEoMAJ0/5Ey505ZTFcNTffoHCF0Vx7EkZAd.4JshcrgRANcb8AKaHw9HĐcN0YHYCHJS
koXweIJ31L5y534fTe1:18605:0:99999:7:::
t-matsu@isserv:~$
```

- Windowsの場合
 - C:\Windows\System32\config\SAM (ローカルアカウントの場合)

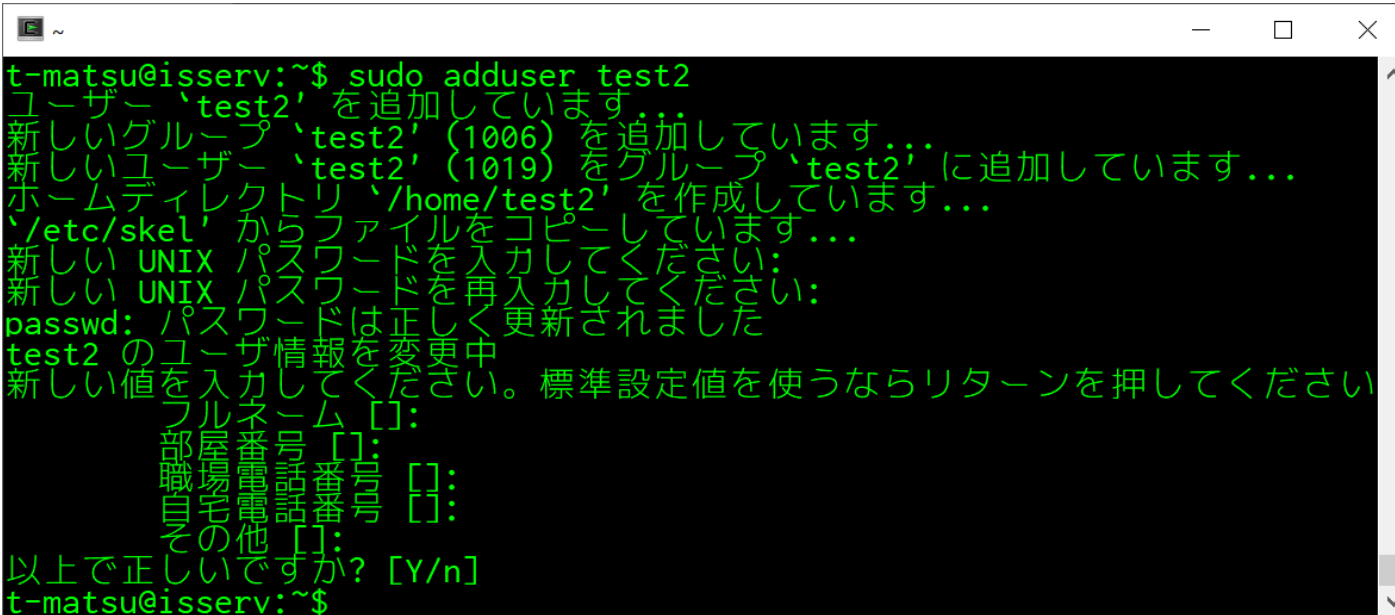
Linuxのパスワード管理(続き)

- /etc/shadowは管理者権限でのみ閲覧可能
- パスワードは平文で保持していない
 - パスワードは暗号化された状態(ハッシュ値)で保持している
 - パスワードファイル奪われてもパスワード文字列の解読は困難
- 文字列も数値なので、ハッシュなどの数値演算処理が可能
 - 第3回講義を参照

実験(ユーザ作成)

- adduser または useradd コマンド (管理者権限が必要)

test1とtest2というユーザを同じパスワードで作成してみる



```
t-matsu@isserv:~$ sudo adduser test2
ユーザ 'test2' を追加しています...
新しいグループ 'test2' (1006) を追加しています...
新しいユーザ 'test2' (1019) をグループ 'test2' に追加しています...
ホームディレクトリ '/home/test2' を作成しています...
'/etc/skel' からファイルをコピーしています...
新しい UNIX パスワードを入力してください:
新しい UNIX パスワードを再入力してください:
passwd: パスワードは正しく更新されました
test2 のユーザ情報を更新中
新しい値を入力してください。標準設定値を使うならリターンを押してください
フルネーム []:
部屋番号 []:
職場電話番号 []:
自宅電話番号 []:
その他 []:
以上で正しいですか? [Y/n]
t-matsu@isserv:~$
```

実験(パスワードのハッシュ値確認)

- /etc/shadowを見る

```
t-matsu@isserv:~$ sudo more /etc/shadow | grep test
test1:$6$aorQLada$sEoMAJ0/5Ey505ZTFcNTffoHCF0Vx7EkZAd.4JshcrgRANcb8AKaHw9H0cN0YHYCHJSkoXweIJ31L5v534fTe1:18605:0:99999:7:::
test2:$6$eC4mhe/i$G29ir9W2FVeiWjNu1IClRXomwNXBNbaDAyRakt.Z/TqhPaik91GVeA80KdsJ5aag.c/wMRejncAT68Bv0XVNs1:18605:0:99999:7:::
t-matsu@isserv:~$
```

- 異なる文字列が保持されている
- 冒頭の\$ \$で囲まれた文字(aorQLada)をsaltと呼ぶ
- フォーマット
 - \$ハッシュアルゴリズム番号\$salt\$ハッシュ値

salt

ランダムなデータハッシュ化する際に
一方向性関数の入力に加える

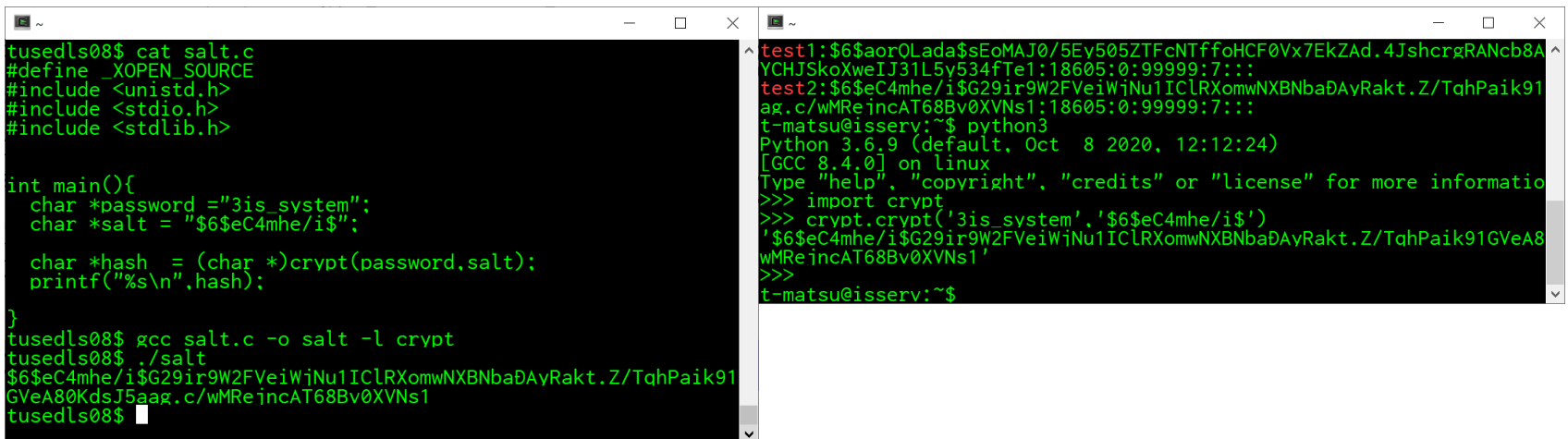
- 確認方法

- man crypt

```
$id$salt$encrypted
DES を使う代わりに、id で使用する暗号化手法を識別し、これがパスワード
文字列の残りの部分を解釈する
方法を決定する。id の値として、以下の値に対応している:
ID | Method
---|-----
1  | MD5
2a | Blowfish (本流の glibc には入っていない:
    | いくつかの Linux ディストリビューションで追加されている)
5  | SHA-256 (glibc 2.7 以降)
6  | SHA-512 (glibc 2.7 以降)
Manual page crypt(3) line 67/103 79% (press h for help or q to quit)
```

Linuxのパスワード管理(続き)

- /etc/shadowにパスワードをハッシュ値として保持
- 実際のパスワード文字列にsaltを結合させて得たハッシュ値を保持
- ハッシュ値を出すライブラリcrypt()が用意されている



```
tusedls08$ cat salt.c
#define _XOPEN_SOURCE
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

int main(){
    char *password = "3is_system";
    char *salt = "$6$eC4mhe/i$";

    char *hash = (char *)crypt(password,salt);
    printf("%s\n",hash);
}

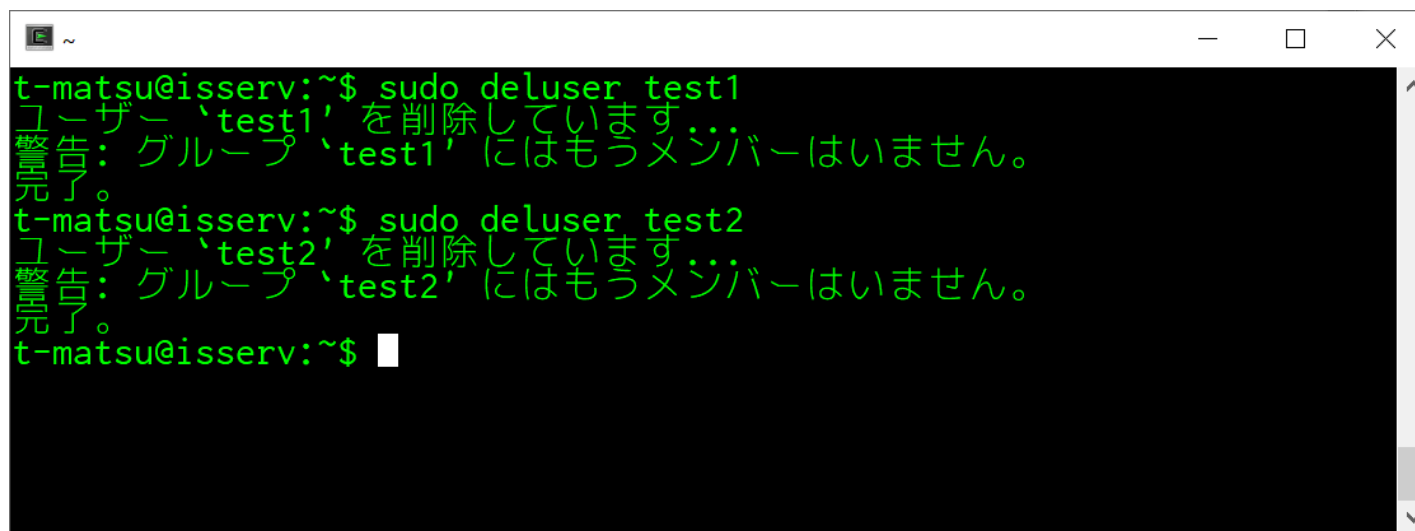
tusedls08$ gcc salt.c -o salt -l crypt
tusedls08$ ./salt
$6$eC4mhe/i$G29ir9W2FVeIWjNu1IClRXomwNXBNbaDAyRakt.Z/TqhPaik91
GVeA80KdsJ5aag.c/wMRejncAT68Bv0XVNs1
tusedls08$
```

```
test1:$6$aor0Lada$sEoMAJ0/5Ey505ZTFcNTffoHCF0Vx7EkZAd.4JshcrgRANcb8A
YCHJSkoXweIJ31L5y534fTe1:18605:0:99999:7:::
test2:$6$eC4mhe/i$G29ir9W2FVeIWjNu1IClRXomwNXBNbaDAyRakt.Z/TqhPaik91
ag.c/wMRejncAT68Bv0XVNs1:18605:0:99999:7:::
t-matsu@isserv:~$ python3
Python 3.6.9 (default, Oct 8 2020, 12:12:24)
[GCC 8.4.0] on linux
Type "help", "copyright", "credits" or "license" for more informatio
>>> import crypt
>>> crypt.crypt('3is_system','$6$eC4mhe/i$')
'$6$eC4mhe/i$G29ir9W2FVeIWjNu1IClRXomwNXBNbaDAyRakt.Z/TqhPaik91GVeA8
wMRejncAT68Bv0XVNs1'
>>>
t-matsu@isserv:~$
```

- 実際に入力されるパスワードのハッシュ値を計算して、
/etc/shadowに記載されている文字列と同じ場合は認証成功とする

後処理

test1 test2ユーザーを実験で作成した場合は削除しておくこと



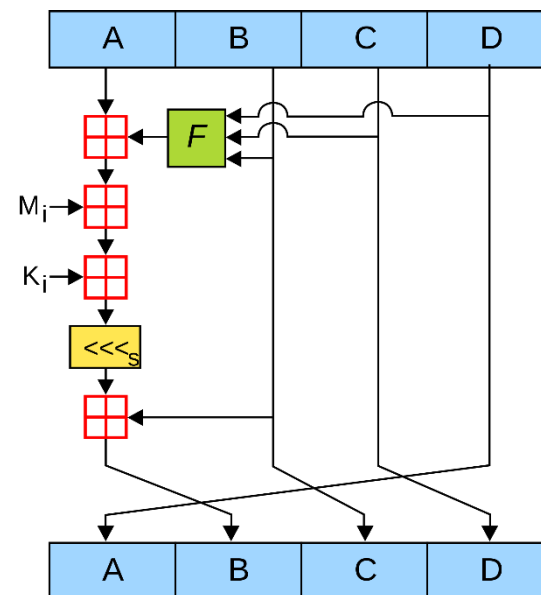
```
t-matsu@isserv:~$ sudo deluser test1
ユーザー 'test1' を削除しています...
警告: グループ 'test1' にはもうメンバーはいません。
完了。
t-matsu@isserv:~$ sudo deluser test2
ユーザー 'test2' を削除しています...
警告: グループ 'test2' にはもうメンバーはいません。
完了。
t-matsu@isserv:~$
```

ハッシュのアルゴリズム

- MD5
 - Message Digest
- Blowfish
- SHA256
 - Secure Hash Algorithm
- SHA512

MD5

- Message Digest algorithm 5
 - 1991年開発
- MD2,MD4の後継
 - MD6もあるがあまり使用されない
- 暗号的ハッシュ関数として必要な強度はない
 - パスワードのハッシュ関数としては使ってはいけない
- 可変長の入力(もちろん文字列でも良い)から128ビット固定のハッシュ値を生成する

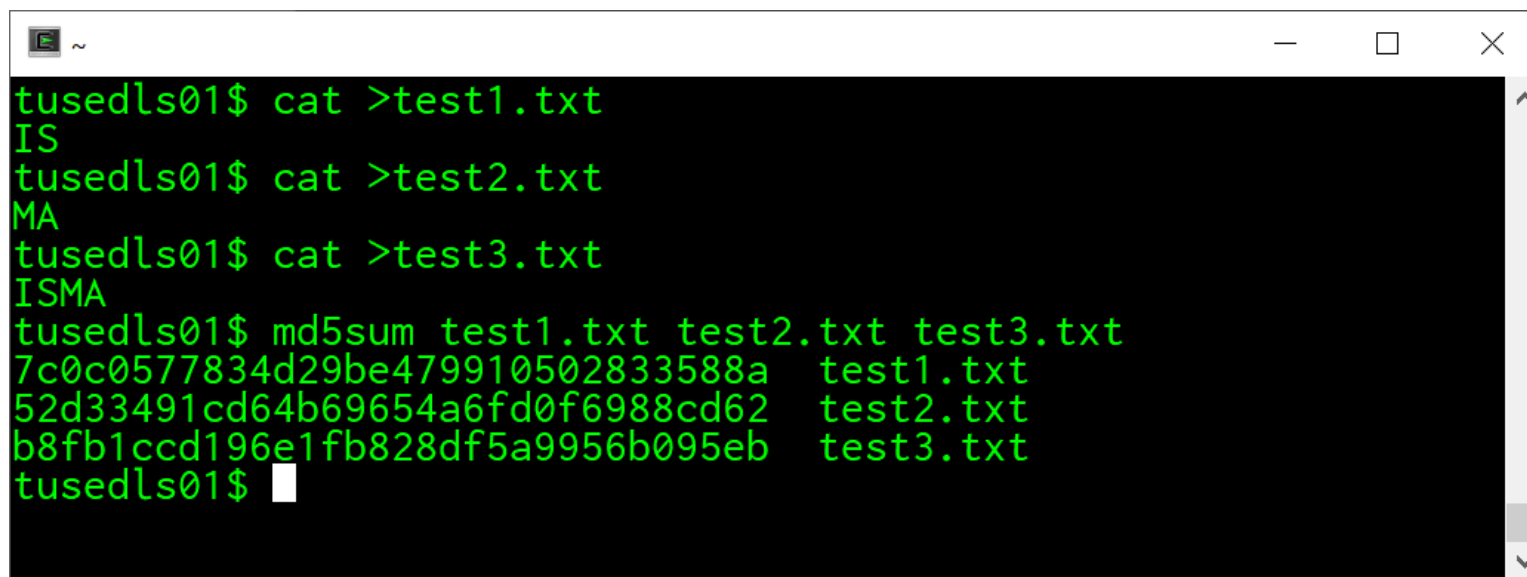


A B C Dは32ビット整数値
Mは入力, Kは定数
この操作を64回

md5sum

\$ md5sum ファイル名

で128ビットのハッシュ値を出力するプログラム

A terminal window with a black background and green text. The window title is '~'. The commands and output are as follows:

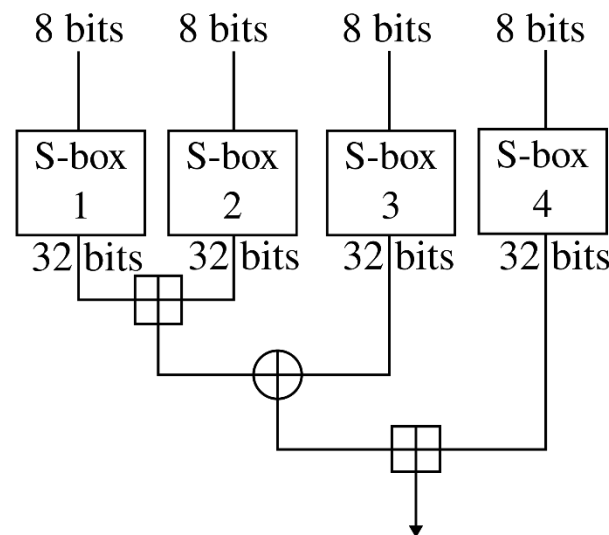
```
tusedls01$ cat >test1.txt
IS
tusedls01$ cat >test2.txt
MA
tusedls01$ cat >test3.txt
ISMA
tusedls01$ md5sum test1.txt test2.txt test3.txt
7c0c0577834d29be479910502833588a  test1.txt
52d33491cd64b69654a6fd0f6988cd62  test2.txt
b8fb1ccd196e1fb828df5a9956b095eb  test3.txt
tusedls01$
```

Windows(PowerShell)は

> certutil -hashfile ファイル名 MD5

Blowfish

- 厳密にはハッシュ関数ではなく暗号方式
 - Blowfish暗号は, 当然復号可能
 - Blowfish暗号はブロック暗号(64ビットブロック)
- Blowfish暗号を基盤としたハッシュ関数
 - 1999年開発(オリジナル)
 - 2a
 - 2x,2y(2011年)
 - 2b(2014年)
- アメリカ国立標準技術研究所(NIST)はBlowfishによるハッシュ関数を承認していない
 - 安全性が計算コストによるもの
 - 最近のLinuxでは標準で使用されない



SHA256 SHA512

- SHA-1

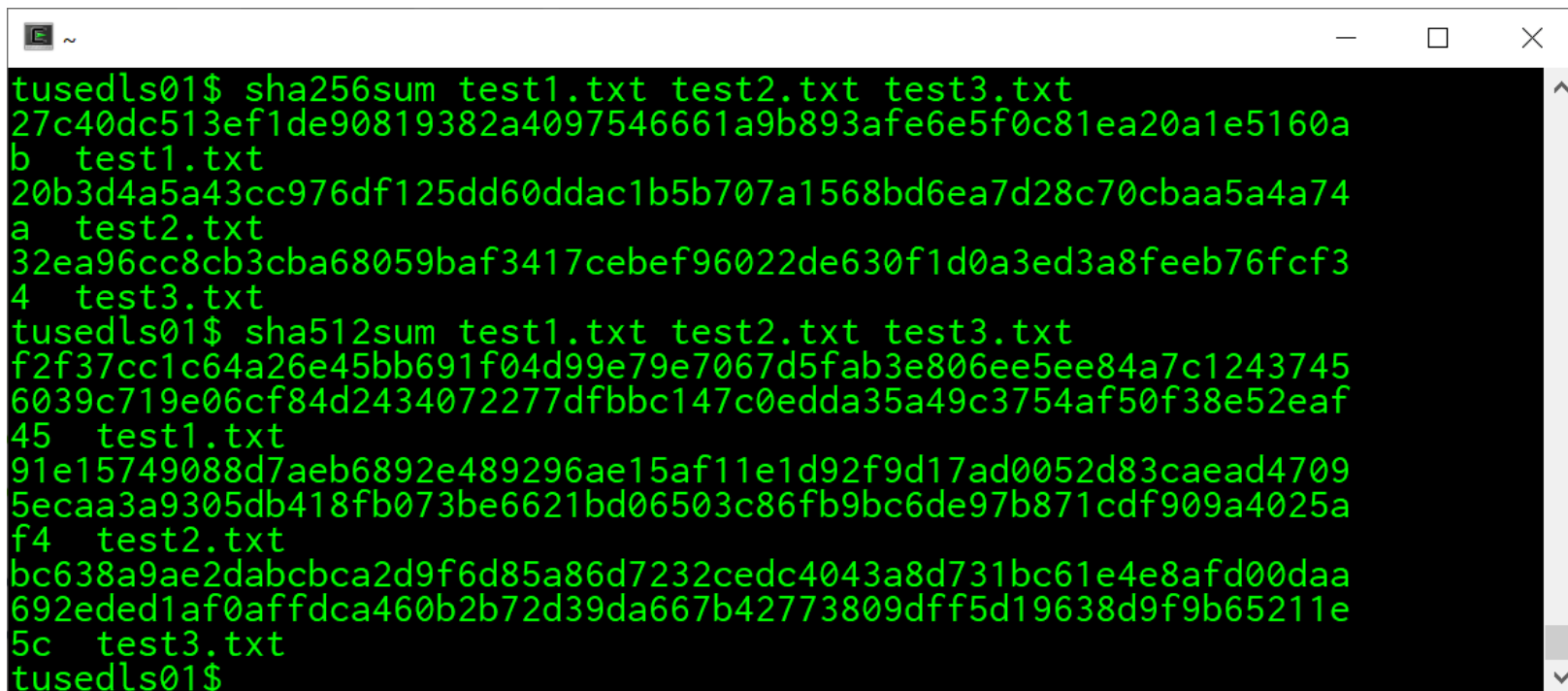
- 1995年から米国標準
- 2004年に欠陥発表
- NISTは2010年までに廃止を申請
- GoogleやMicrosoftなども2017年までに廃止を表明

- SHA-2

- SHA256, SHA512などがある(224,256,384,512)
- SHA256は32ビット, SHA512は64ビットを1ワードとして扱う
 - アルゴリズムは基本的に同じ(初期値や繰り返し回数などは異なる)
- 2010年 NISTがSHA-1からの移行を推奨

sha256sum sha512sum

md5sumと使い方は同じ

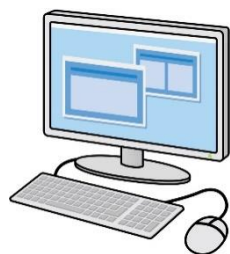
A terminal window with a black background and green text. The window title is '~'. It shows two commands being executed: 'sha256sum test1.txt test2.txt test3.txt' and 'sha512sum test1.txt test2.txt test3.txt'. The outputs are long hexadecimal strings, each followed by the filename. The first command's output is: '27c40dc513ef1de90819382a4097546661a9b893afe6e5f0c81ea20a1e5160a b test1.txt', '20b3d4a5a43cc976df125dd60ddac1b5b707a1568bd6ea7d28c70cbaa5a4a74 a test2.txt', and '32ea96cc8cb3cba68059baf3417cebef96022de630f1d0a3ed3a8feeb76fcf3 4 test3.txt'. The second command's output is: 'f2f37cc1c64a26e45bb691f04d99e79e7067d5fab3e806ee5ee84a7c1243745 6039c719e06cf84d2434072277dfbbc147c0edda35a49c3754af50f38e52eaf 45 test1.txt', '91e15749088d7aeb6892e489296ae15af11e1d92f9d17ad0052d83caead4709 5ecaa3a9305db418fb073be6621bd06503c86fb9bc6de97b871cdf909a4025a f4 test2.txt', and 'bc638a9ae2dabcbca2d9f6d85a86d7232cedc4043a8d731bc61e4e8afd00daa 692eded1af0affdca460b2b72d39da667b42773809dff5d19638d9f9b65211e 5c test3.txt'. The prompt 'tusedls01\$' is visible at the end of each command line.

```
tusedls01$ sha256sum test1.txt test2.txt test3.txt
27c40dc513ef1de90819382a4097546661a9b893afe6e5f0c81ea20a1e5160a
b test1.txt
20b3d4a5a43cc976df125dd60ddac1b5b707a1568bd6ea7d28c70cbaa5a4a74
a test2.txt
32ea96cc8cb3cba68059baf3417cebef96022de630f1d0a3ed3a8feeb76fcf3
4 test3.txt
tusedls01$ sha512sum test1.txt test2.txt test3.txt
f2f37cc1c64a26e45bb691f04d99e79e7067d5fab3e806ee5ee84a7c1243745
6039c719e06cf84d2434072277dfbbc147c0edda35a49c3754af50f38e52eaf
45 test1.txt
91e15749088d7aeb6892e489296ae15af11e1d92f9d17ad0052d83caead4709
5ecaa3a9305db418fb073be6621bd06503c86fb9bc6de97b871cdf909a4025a
f4 test2.txt
bc638a9ae2dabcbca2d9f6d85a86d7232cedc4043a8d731bc61e4e8afd00daa
692eded1af0affdca460b2b72d39da667b42773809dff5d19638d9f9b65211e
5c test3.txt
tusedls01$
```

Windows(PowerShell)は

> certutil -hashfile ファイル名 SHA256

ネットワーク経由のパスワード認証

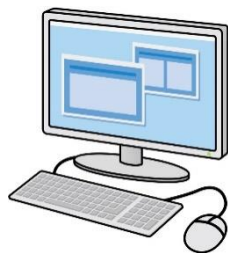


パスワードのハッシュ「sEoMAJ0/5....」

毎回同じハッシュ値を送る場合
同じハッシュ値を第3者が使うことでなりすまし可能

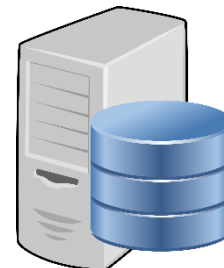


もう工夫



毎回異なる適当な文字列

「送られてきた文字列+パスワード」のハッシュ



ワンタイムパスワード認証という

パスワード以外の使われ方

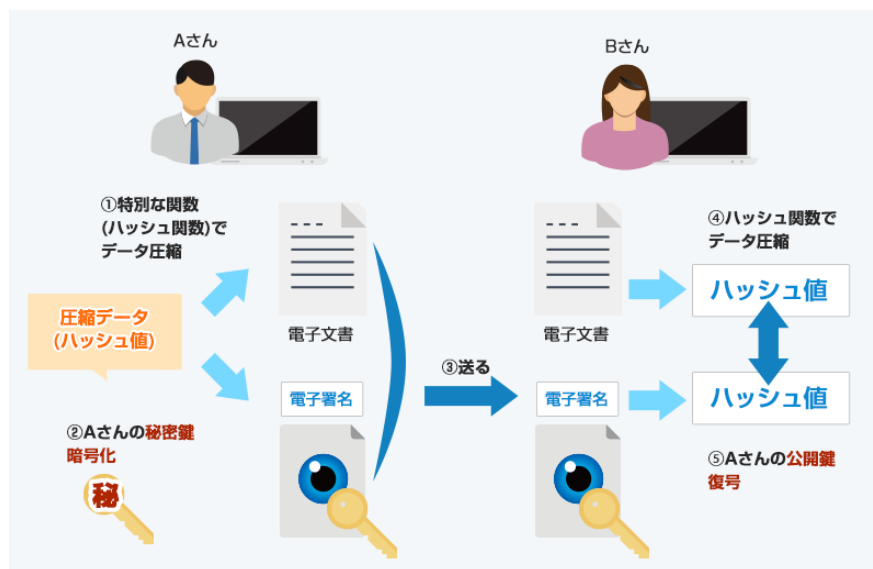
- ファイル内部の欠落チェック(真正性チェック)
 - 安全性強度はそれほど問題にならない

ubuntu-ja-20.04.1-desktop-amd64.iso (ISOイメージ) (md5sum: 3f9e1b8180b4b24eeffadef450878d6a)

ubuntu-ja-20.04.1-desktop-amd64.iso.torrent (Torrentファイル) (md5sum: 0a0a21b8e258146539d46bdda15d089d)

- 検索時などのキー
 - 連想配列

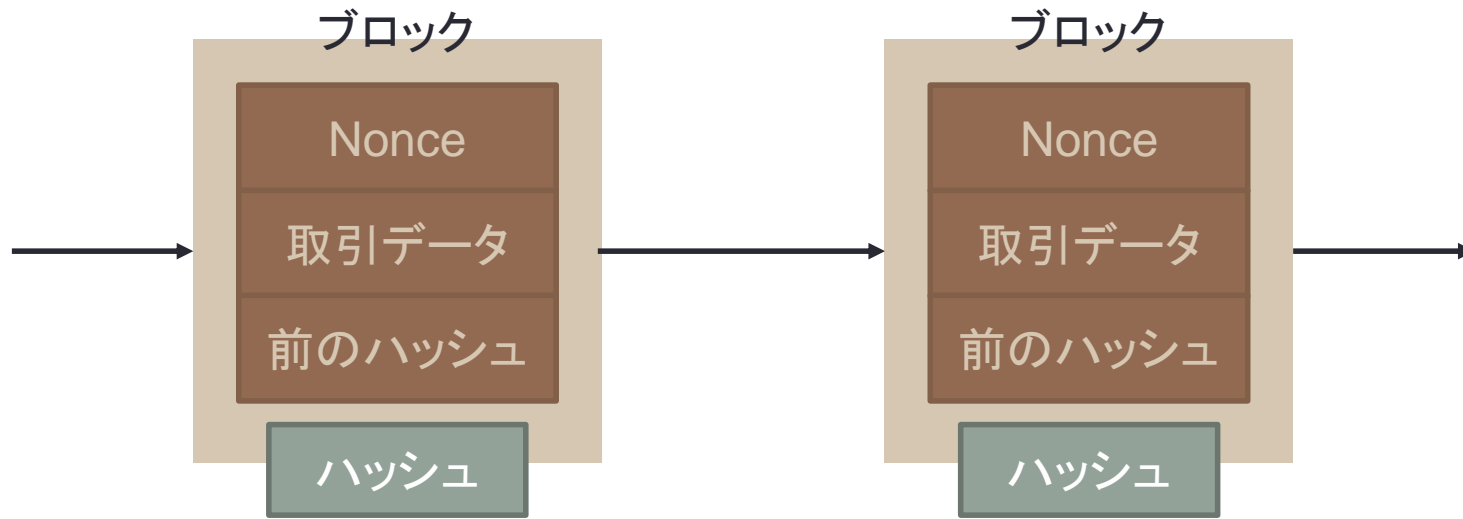
- 署名
 - 公開鍵暗号と併用
 - TLS(ssh, https..)
 - ブロックチェーン
 - etc



引用元: Global Sign

ブロックチェーン

- ハッシュとリストの応用技術
 - ハッシュ値における制限
 - Nonceの計算(探索)
 - 探索行為をマイニングと呼ぶ



電子署名

- 公開鍵暗号とハッシュを利用
- 公開鍵暗号は、秘密鍵、公開鍵の二種類の鍵を使う
 - 公開鍵で暗号化した暗号文は秘密鍵で復号
 - 秘密鍵で暗号化した暗号文(暗号文とは呼ばない)は公開鍵で復号
- 印鑑レス(印鑑不要論)の現在の時勢において注目を集める

余談: RSA暗号(公開鍵暗号)

- 素数 p と q を選ぶ, $n = p * q$ とする(p と q は互いに素である)
- $\gcd(e, (p - 1)(q - 1)) = 1$ となる e を選ぶ
- $ed = 1 \bmod (p - 1)(q - 1)$ となる d を選ぶ
- (n, e) を公開鍵とする

- 平文 $m \in Z_p^*$ において暗号文 c を

$$c = m^e \bmod n \quad \text{と計算する}$$

- 暗号文 $c \in Z_p^*$ において暗号文 m を

$$m = c^d \bmod n \quad \text{と計算する}$$

$$\text{つまり } m^{ed} \bmod n = m$$

やってみよう: 秘密・公開鍵生成

秘密鍵

```
tusedls01$ openssl genrsa 1024 > privatekey.pem
Generating RSA private key, 1024 bit long modulus
.....+++++
.....+++++
e is 65537 (0x10001)
tusedls01$ cat privatekey.pem
-----BEGIN RSA PRIVATE KEY-----
MIICX0IBAAKBgQC/VhHMwLNWp+4t0IxVy6kmBfrSuwEiDndluRjvOhr93w5Nr+0O
ZrOqtpdoHm8hH1FjJhb504+iz4ui51k/3qV8rmEOZR6PaFJo2Fm1WlvTBmTejo0F
rFGegYBEWeGo0/mL2CXqzn5L0uy6lfo/hZGb6A8Jrh2d4LhKffHIm0JvvyIDAQAB
AoGBAIFVoLqTF6bf5P07IGFuRWxaZm3D0s0+mx67u8sYLqg8+GIzHyqnAZPH0Rik
WeI91Tn8KVNdA96+BFmdoIq36YvsKth5KW3gynefn8biIWzxC7rOGx0G20BVY5lS
gy5aNmrB0m0i3rG/Ep9SxrGZ0TFY/sK8ZUGIF7Pfa+bPZgphAkEA/YXm5CfLgqg0
U+ZjOqSKVZU17BQyTmfadd9K8clMrOCNMkmwsr3YwMMF8wHkWXcVn4ThIXSuBiIQ
rB7efMietwJBAME0oRbawVd1jmYUpl4UKncE/HZtNCLiROmAq/680FH091t6SD7
oKeVjASBGvqNuw8BaS8oxooR2IKOPNHZY40CO0Dnex6RzeAc4VOLub16qwgAEqHU
cTBptJucqpAkanePMwS3n0jQe1Vw0+EXu++C1sgUb+Cbiozm3q24asscxs/LAkAr
BIzZiduCnaoXEIH6cIM0/Rg5RhdXH4o/k2ZtyHJhFns90jsWHPCoX6PNXPGk1+n0
bYH9/hY2p8KNT5A7KSwBAkBNL3BL8ix79NVp+wrYtgVeJu8SW0qRFS0VQyXj6KrG
0/XC63FgywKs/fK2py+VGEeW0JgtTsgSna8FJZneS4g6
-----END RSA PRIVATE KEY-----
tusedls01$
```

備考

m^e の e は65537固定値が
よく使われる

公開鍵

```
tusedls01$ openssl rsa -in privatekey.pem -pubout -out publickey.pem
writing RSA key
tusedls01$ cat publickey.pem
-----BEGIN PUBLIC KEY-----
MIGfMA0GCsGgSIb3D0EBAQUAA4GNA0CBiOKBgQC/VhHMwLNWp+4t0IxVy6kmBfrS
uwEiDndluRjvOhr93w5Nr+0OZrOqtpdoHm8hH1FjJhb504+iz4ui51k/3qV8rmEO
ZR6PaFJo2Fm1WlvTBmTejo0FrFGegYBEWeGo0/mL2CXqzn5L0uy6lfo/hZGb6A8J
rh2d4LhKffHIm0JvvyIDAQAB
-----END PUBLIC KEY-----
tusedls01$
```

やってみよう:暗号化と復号

暗号化

```
tusedls01$ openssl rsautl -encrypt -pubin -inkey publickey.pem -in test1.txt -out test1.enc
tusedls01$ hexdump test1.enc
00000000 690a d82f e7a2 4efc bd1b 7240 a0f2 ff56
00000010 c7b0 06af b47d f4db 330e 95e0 547e 7194
00000020 3f74 362d 8021 f1ec 8801 0197 c746 bdd5
00000030 aa36 3f5e a020 8834 68d1 da1b c888 4abd
00000040 9008 841f bb18 6c8b 7c9a a2e9 0766 ffa1
00000050 e6b7 5633 1bcd b00d cb28 70fe ff8a 7f8c
00000060 5035 cbae 134c 4c8a d676 03e4 9302 a378
00000070 568b c243 7ab8 14aa 1506 1048 5e94 bfa4
00000080
tusedls01$
```

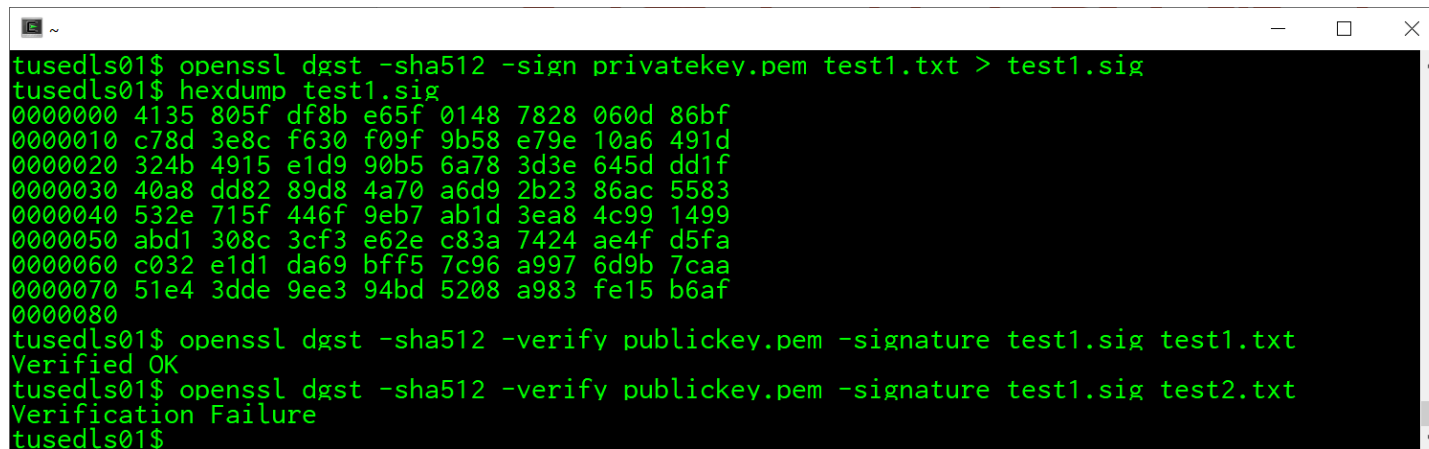
復号

```
tusedls01$ openssl rsautl -decrypt -inkey privatekey.pem -in test1.enc
IS
tusedls01$
```

RSA暗号に興味のある方は

<http://www.is.noda.tus.ac.jp/~t-matsu/3is/RSA.java> (学内のみ)

やってみよう: 電子署名



```
tusedls01$ openssl dgst -sha512 -sign privatekey.pem test1.txt > test1.sig
tusedls01$ hexdump test1.sig
00000000 4135 805f df8b e65f 0148 7828 060d 86bf
00000100 c78d 3e8c f630 f09f 9b58 e79e 10a6 491d
00000200 324b 4915 e1d9 90b5 6a78 3d3e 645d dd1f
00000300 40a8 dd82 89d8 4a70 a6d9 2b23 86ac 5583
00000400 532e 715f 446f 9eb7 ab1d 3ea8 4c99 1499
00000500 abd1 308c 3cf3 e62e c83a 7424 ae4f d5fa
00000600 c032 e1d1 da69 bff5 7c96 a997 6d9b 7caa
00000700 51e4 3dde 9ee3 94bd 5208 a983 fe15 b6af
00000800
tusedls01$ openssl dgst -sha512 -verify publickey.pem -signature test1.sig test1.txt
Verified OK
tusedls01$ openssl dgst -sha512 -verify publickey.pem -signature test1.sig test2.txt
Verification Failure
tusedls01$
```

test1.txt は verified OK

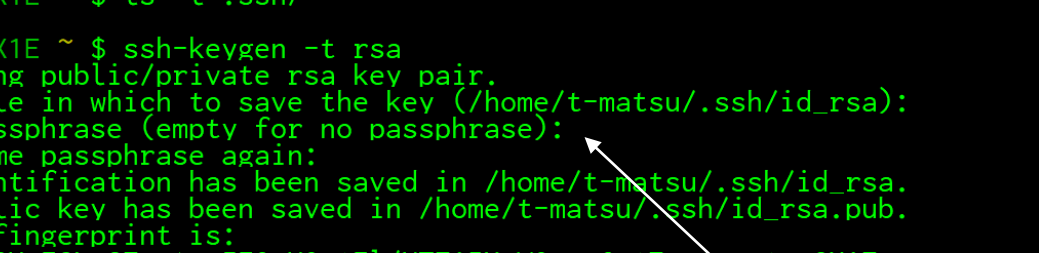
test2.txt は Failure

公開鍵認証

- Linuxのリモートログイン
 - パスワード認証
 - どこからでも(パスワードを知っていれば)ログインできる
 - パスワードを知られてしまえば第3者もログインも可能
 - 公開鍵認証
 - 秘密鍵を持っているPCからのみログイン可能
 - パスワード不要
 - セキュリティはパスワード認証より上
 - 電子署名のアルゴリズムを使う
 - 秘密鍵を持っていることが本人確認となる(電子署名と同じ)

やってみよう: 公開鍵認証

接続元(自身のPC)で鍵生成



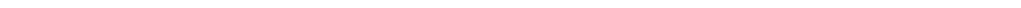
```
t-matsu@X1E ~ $ ls -l .ssh/
合計 0
t-matsu@X1E ~ $ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/t-matsu/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/t-matsu/.ssh/id_rsa.
Your public key has been saved in /home/t-matsu/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:4PUv7GLqSEosA+p5F0mU9vt7l/NTE1IKyW9mzw6vtEw t-matsu@X1E
The key's randomart image is:
+---[RSA 3072]-----+
|
| ..          +
| +          +
| . o o .   o o
|   = o .   B .
|   + S . + +
| o .   o   . . =
| + o o .   o o E =
| . + . . . oo = + . o
| oo .. o + o . o o
+---[SHA256]-----+
t-matsu@X1E ~ $
```

パスフレーズは空で良い

空でない場合は秘密鍵への
アクセスパスフレーズとなる

パズフレーズは空で良い

空でない場合は秘密鍵への
アクセスパスフレーズとなる



A terminal window showing the command `ls .ssh/` being executed. The output is `id_rsa id_rsa.pub`, indicating that the private and public keys have been successfully generated. The prompt is `t-matsu@X1E ~ $`.

秘密鍵と公開鍵(.pub)の完成

やってみよう: 公開鍵認証

公開鍵を接続先(tusedt000.ed.tus.ac.jp)へコピー

```
t-matsu@X1E ~ $  
t-matsu@X1E ~ $ scp /home/t-matsu/.ssh/id_rsa.pub j-matsu@tusedt000.ed.tus.ac.jp:~/  
t-matsu@X1E ~ $
```

接続先(tusedt000)で公開鍵をauthorized_keysに追加

```
tusedls15$ cat id_rsa.pub >> ~/.ssh/authorized_keys  
注意:  
>> は追記  
> は上書き
```

一応パーミッションの確認(自分のみ読み書き可能)

```
tusedls15$ ls -l .ssh/authorized_keys  
-rw----- 1 j-matsu teach 2360 12月 10 14:11 .ssh/authorized_keys  
tusedls15$
```

ログアウトしてもう一度ログイン(パスワード無しでログインできればOK)

```
tusedls15$ ログアウト  
Connection to tusedt000.ed.tus.ac.jp closed.  
t-matsu@X1E ~ $ ssh j-matsu@tusedt000.ed.tus.ac.jp  
Last login: Thu Dec 10 14:35:59 2020 from nodars028231.is-fw.noda.tus.ac.jp  
tusedls15$
```

まとめ

ハッシュ

- MD5
- SHA-1, SHA-2(SHA256,512)
- パスワード認証, 電子署名, etc, さまざまなケースで活用

質問あればどうぞ