暗号について

SecPrj Intro-phase

暗号とは

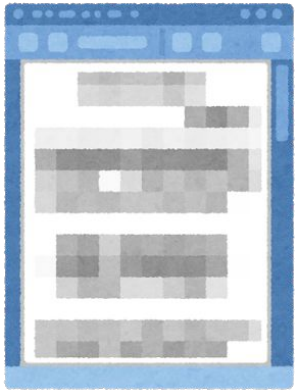
暗号とは、ある情報を特定の決まった人しか読めないように一定の手順に基づいて無意味な文字や符号の列に置き換えたもの。情報の伝送や記録、保存の際、第三者に盗み見られたり**改竄**されないようにするために作成される。

暗号（cryptograph）とは - IT用語辞典 e-Words https://e-words.jp/w/%E6%9A%97%E5%8F%B7.html

平文と暗号

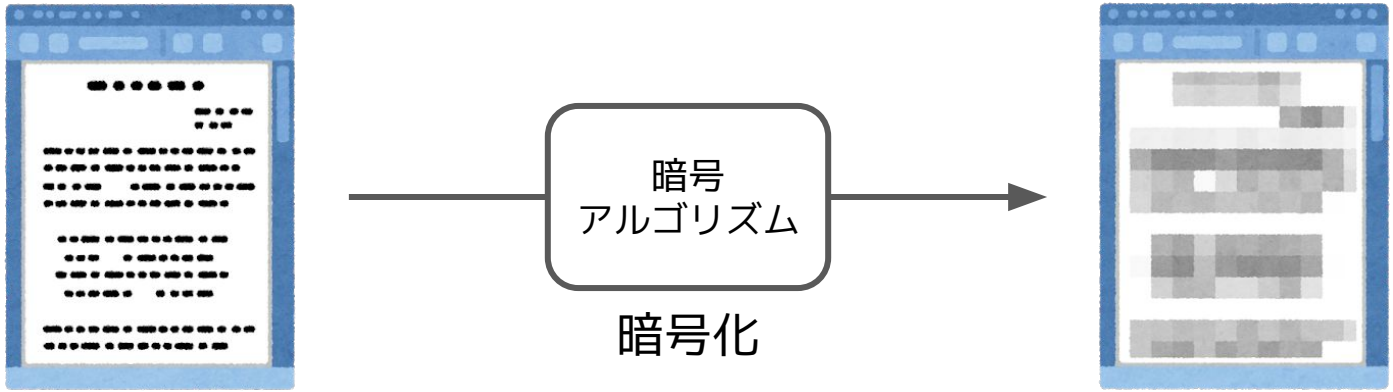
平文

暗号化



暗号文

平文と暗号



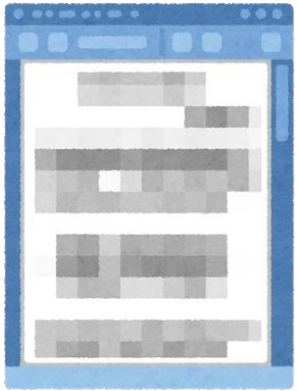
平文と暗号

暗号鍵

平文

暗号
アルゴリズム

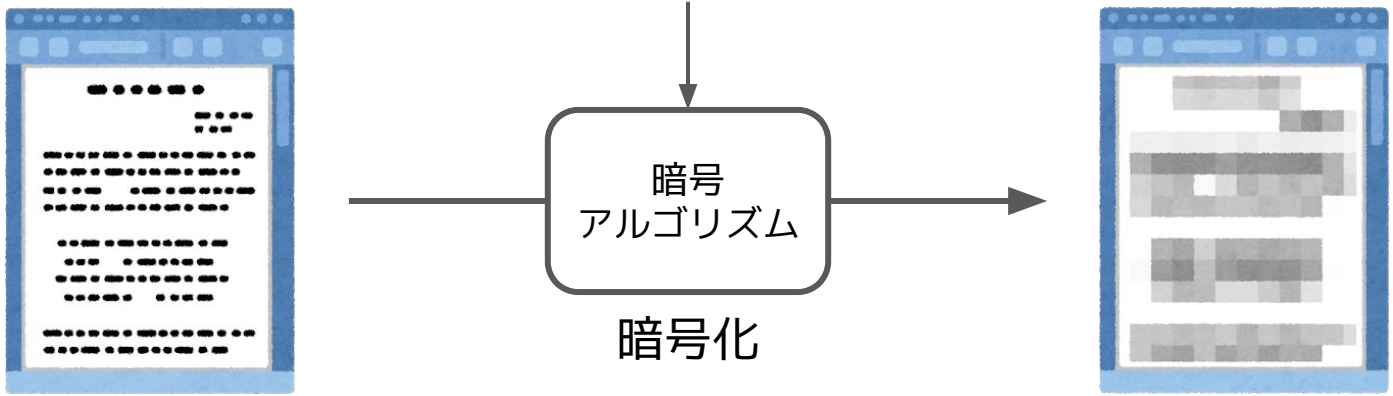
暗号化



暗号文

平文と暗号

暗号鍵



暗号化

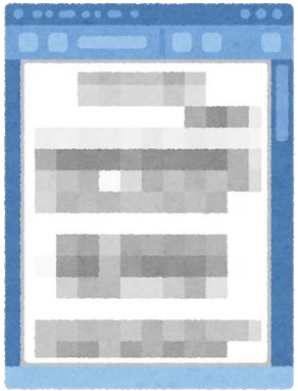
平文

暗号文

暗号化アルゴリズムと暗号鍵によって暗号化する

暗号化と復号

暗号鍵



暗号文

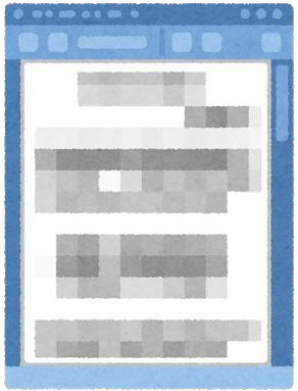
復号
アルゴリズム

復号

平文

暗号化と復号

暗号鍵



暗号文

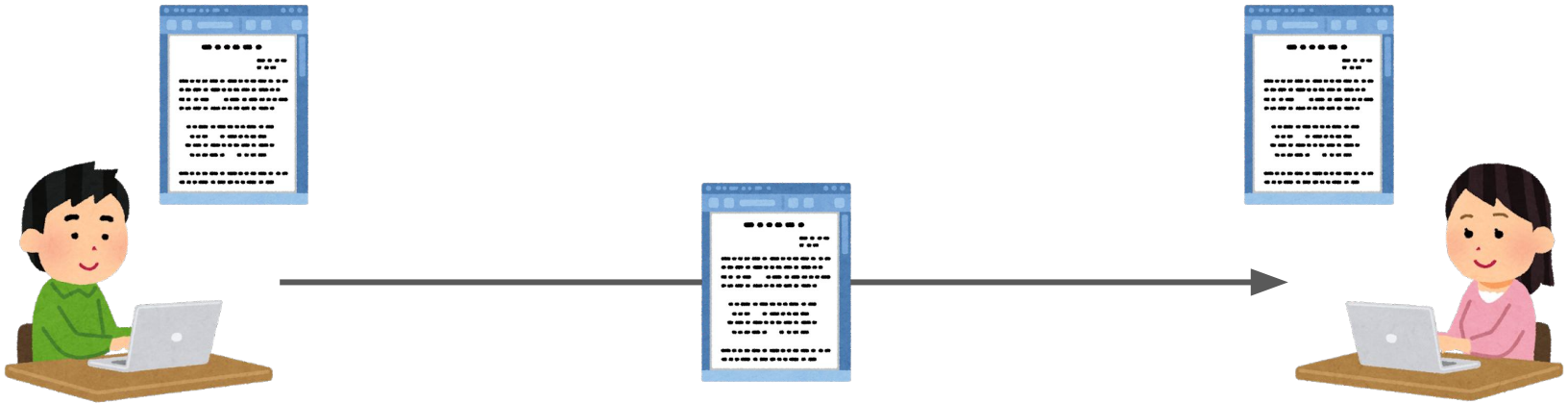
復号
アルゴリズム

復号

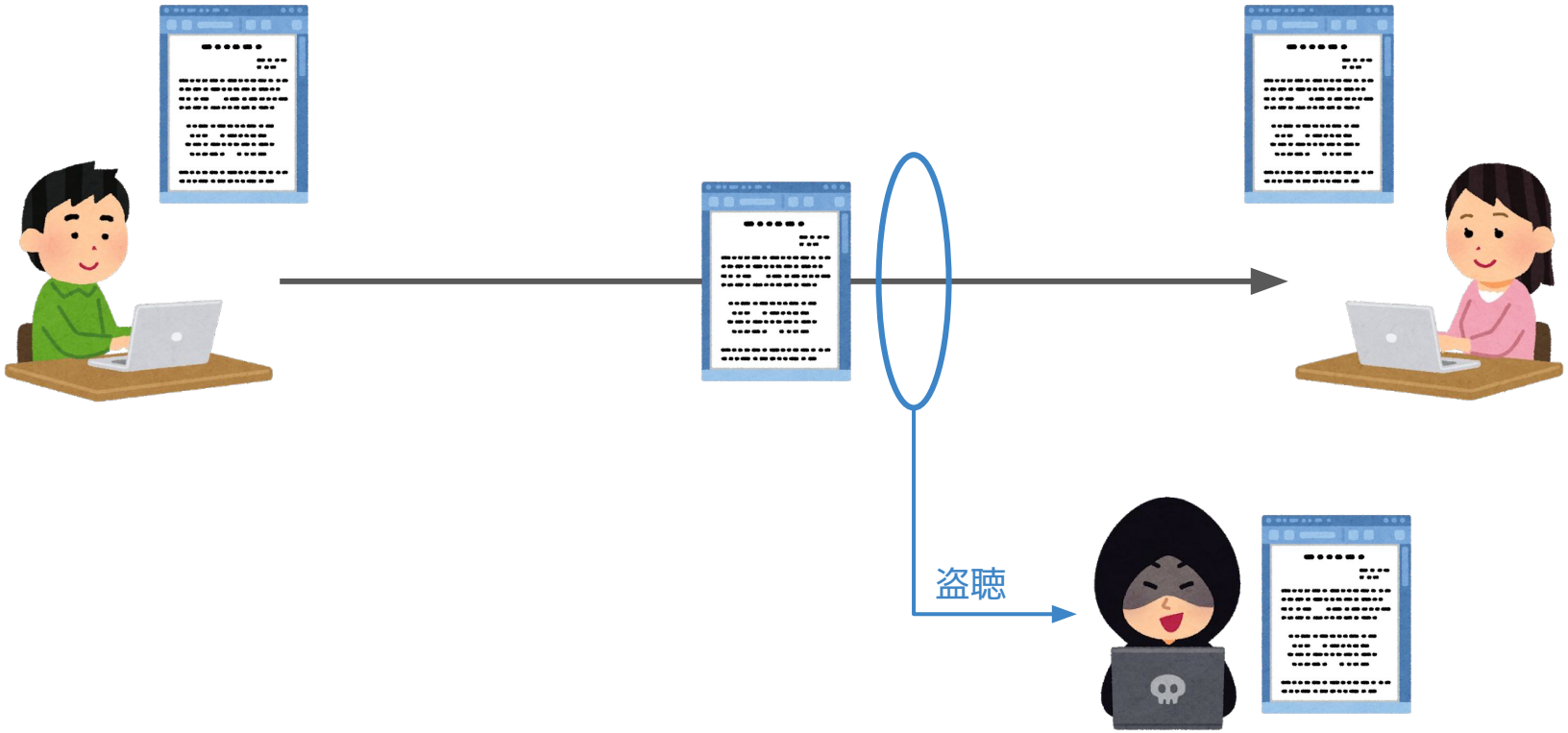
平文

正当な手段で平文化することを復号という

平文通信



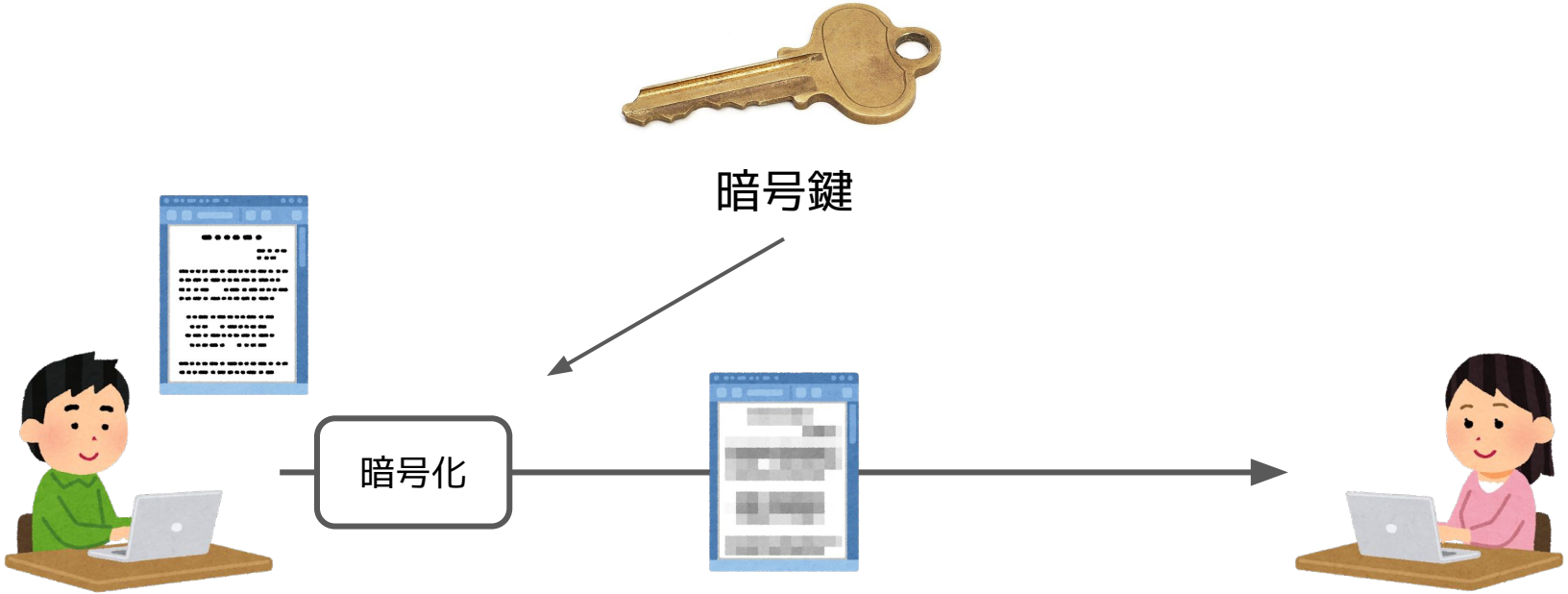
平文通信



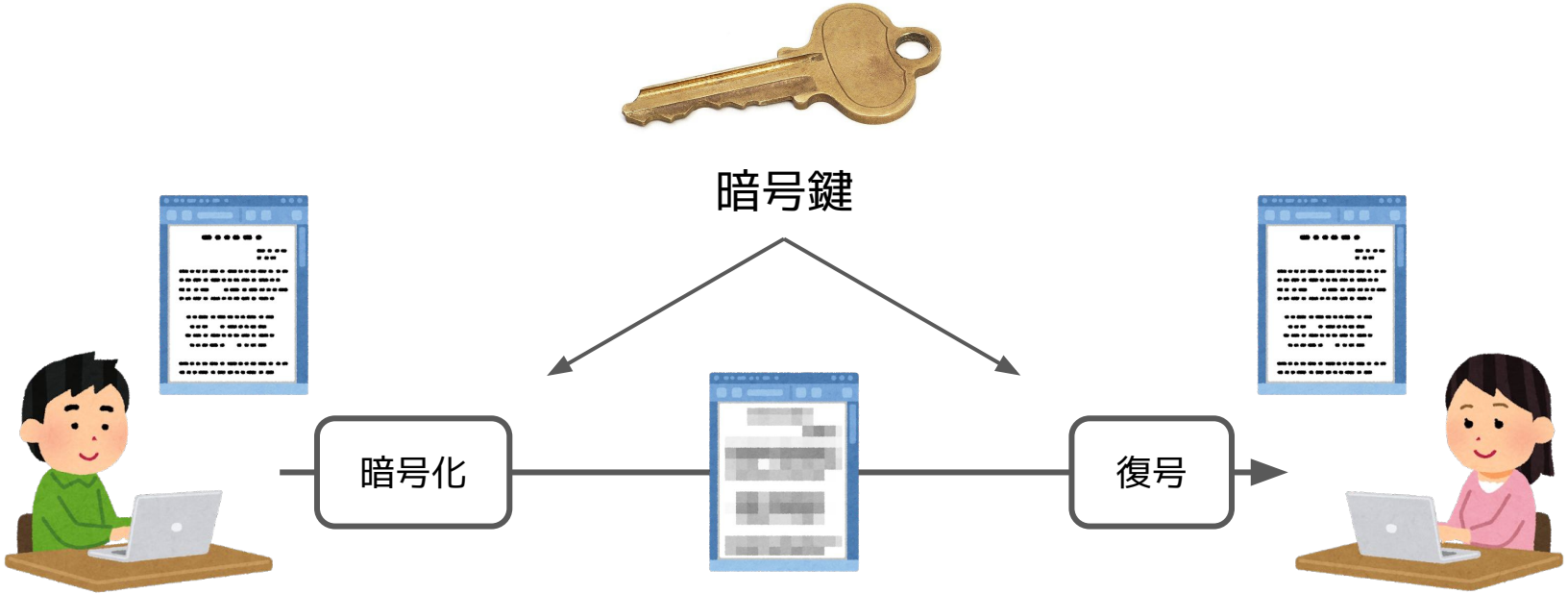
暗号通信

暗号鍵

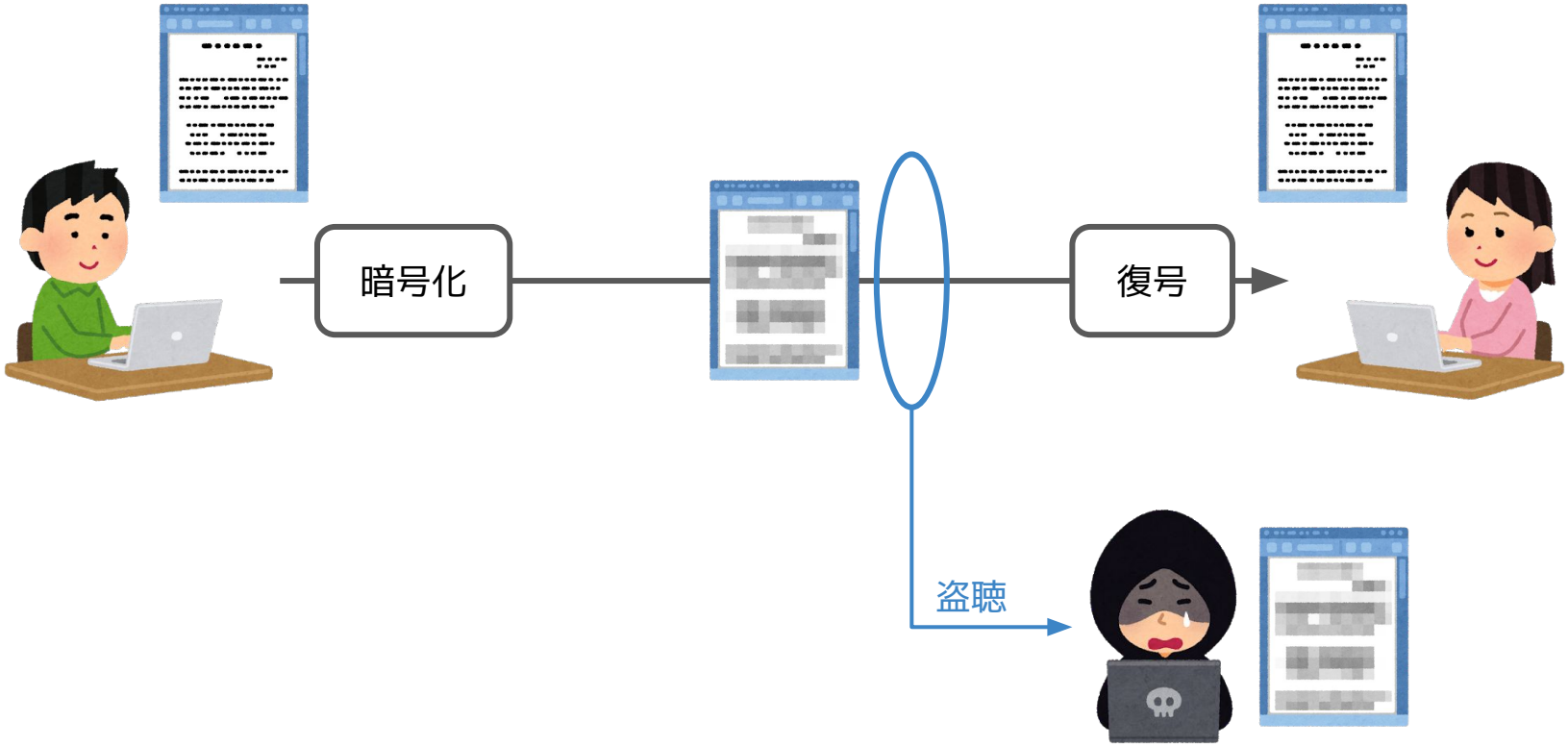
暗号通信



暗号通信



暗号通信



鍵交換問題

それぞれのマシンで暗号化／復号するなら，鍵を共有しておく必要がある

でも鍵を共有するタイミングで鍵ごと盗聴されたら終わり

暗号化と復号に同一の鍵を使う以上，鍵交換問題からは避けられない

暗号化と復号に同一の鍵を使わなければいいのでは？

暗号鍵と復号鍵を分離する

１ユーザごとに鍵を２つもつ

自分に送ってもらう際に，自分の暗号鍵を使って暗号化してもらう

それを自分が持ってる復号鍵で復号する

暗号鍵

公開する

Data

復号鍵

公開しない

暗号鍵と復号鍵を分離する

１ユーザごとに鍵を２つもつ

自分に送ってもらう際に，自分の暗号鍵を使って暗号化してもらう

それを自分が持ってる復号鍵で復号する

暗号鍵

公開する

復号鍵

公開しない

暗号鍵と復号鍵を分離する

１ユーザごとに鍵を２つもつ

自分に送ってもらう際に，自分の暗号鍵を使って暗号化してもらう

それを自分が持ってる復号鍵で復号する

暗号鍵

公開する

復号鍵

公開しない

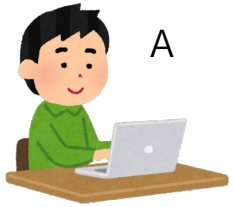
公開鍵

暗号鍵A

秘密鍵

復号鍵A

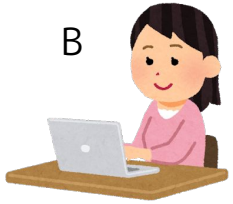
Data



A

AからBへの送信を想定

B



暗号鍵B

公開鍵

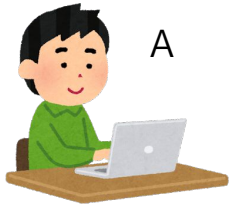
復号鍵B

秘密鍵

公開鍵 暗号鍵A

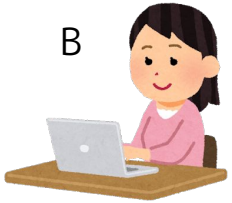
秘密鍵 復号鍵A

Data



A

B



暗号鍵B 公開鍵

復号鍵B 秘密鍵

0.事前に公開鍵を交換

暗号鍵A

暗号鍵B

公開鍵

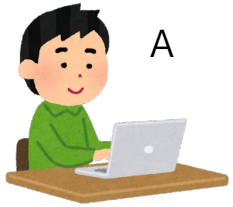
秘密鍵

暗号鍵A

復号鍵A

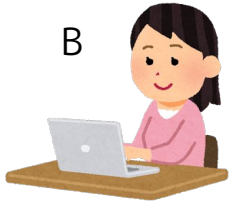
Data

暗号鍵B



A

B



暗号鍵B

復号鍵B

暗号鍵A

公開鍵

秘密鍵

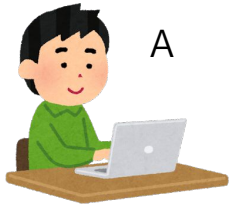
0.事前に公開鍵を交換

公開鍵 暗号鍵A

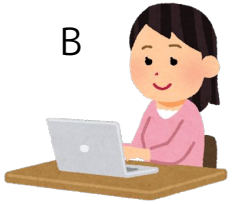
秘密鍵 復号鍵A

Data

暗号鍵B



A



B

暗号鍵B 公開鍵

復号鍵B 秘密鍵

暗号鍵A

0.事前に公開鍵を交換

1.送るデータをBの公開鍵で暗号化

公開鍵

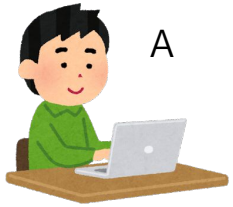
秘密鍵

暗号鍵A

復号鍵A

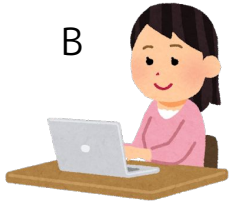
Data

暗号鍵B



A

B



暗号鍵B

復号鍵B

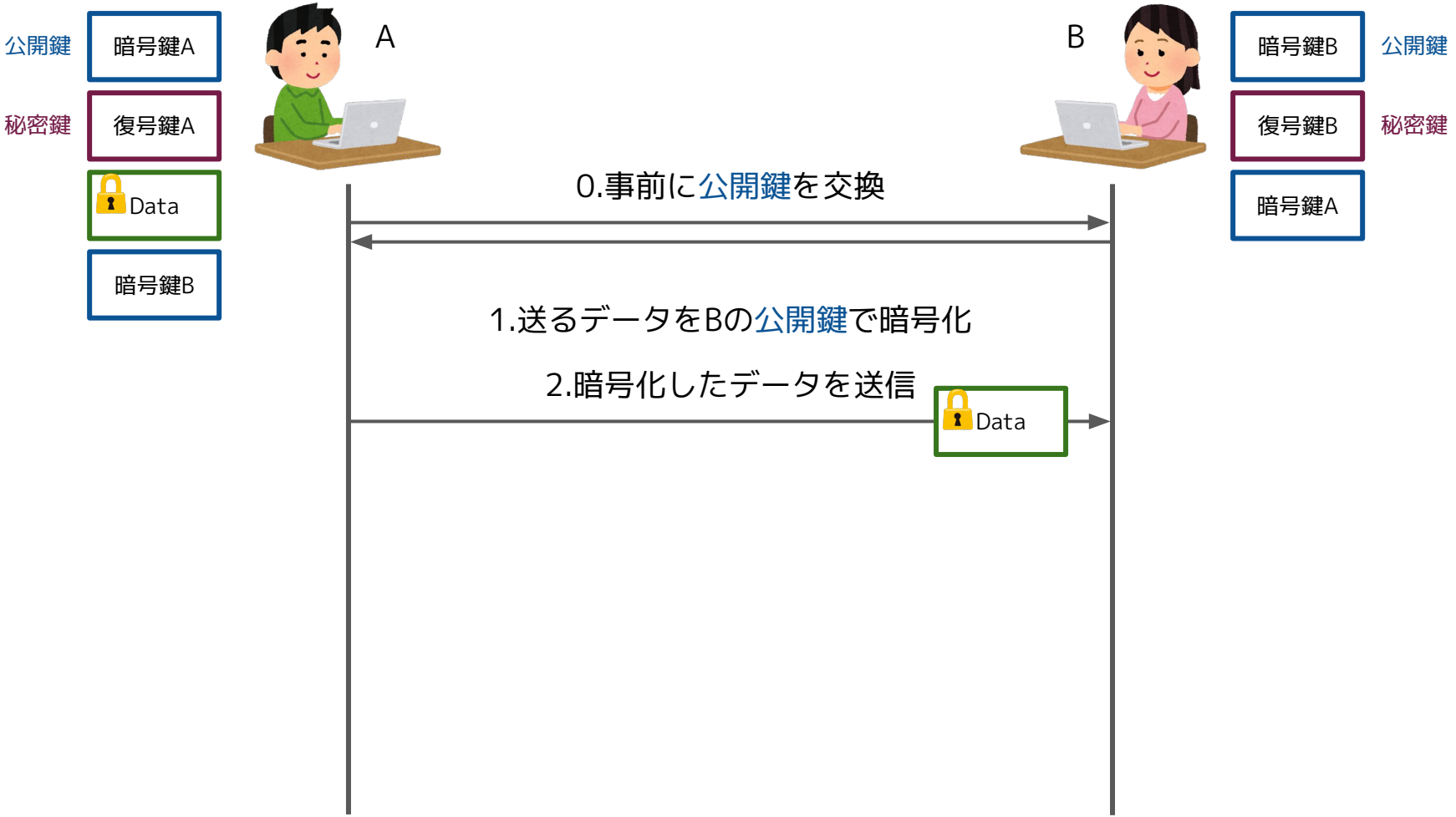
暗号鍵A

公開鍵

秘密鍵

0.事前に公開鍵を交換

1.送るデータをBの公開鍵で暗号化


公開鍵
暗号鍵A
秘密鍵
復号鍵A
Data
暗号鍵B
A
B
暗号鍵B
公開鍵
復号鍵B
秘密鍵
暗号鍵A
0.事前に公開鍵を交換
1.送るデータをBの公開鍵で暗号化
2.暗号化したデータを送信
Data

公開鍵

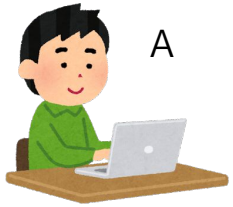
秘密鍵

暗号鍵A

復号鍵A

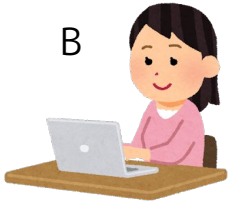
Data

暗号鍵B



A

B



暗号鍵B

復号鍵B

暗号鍵A

Data

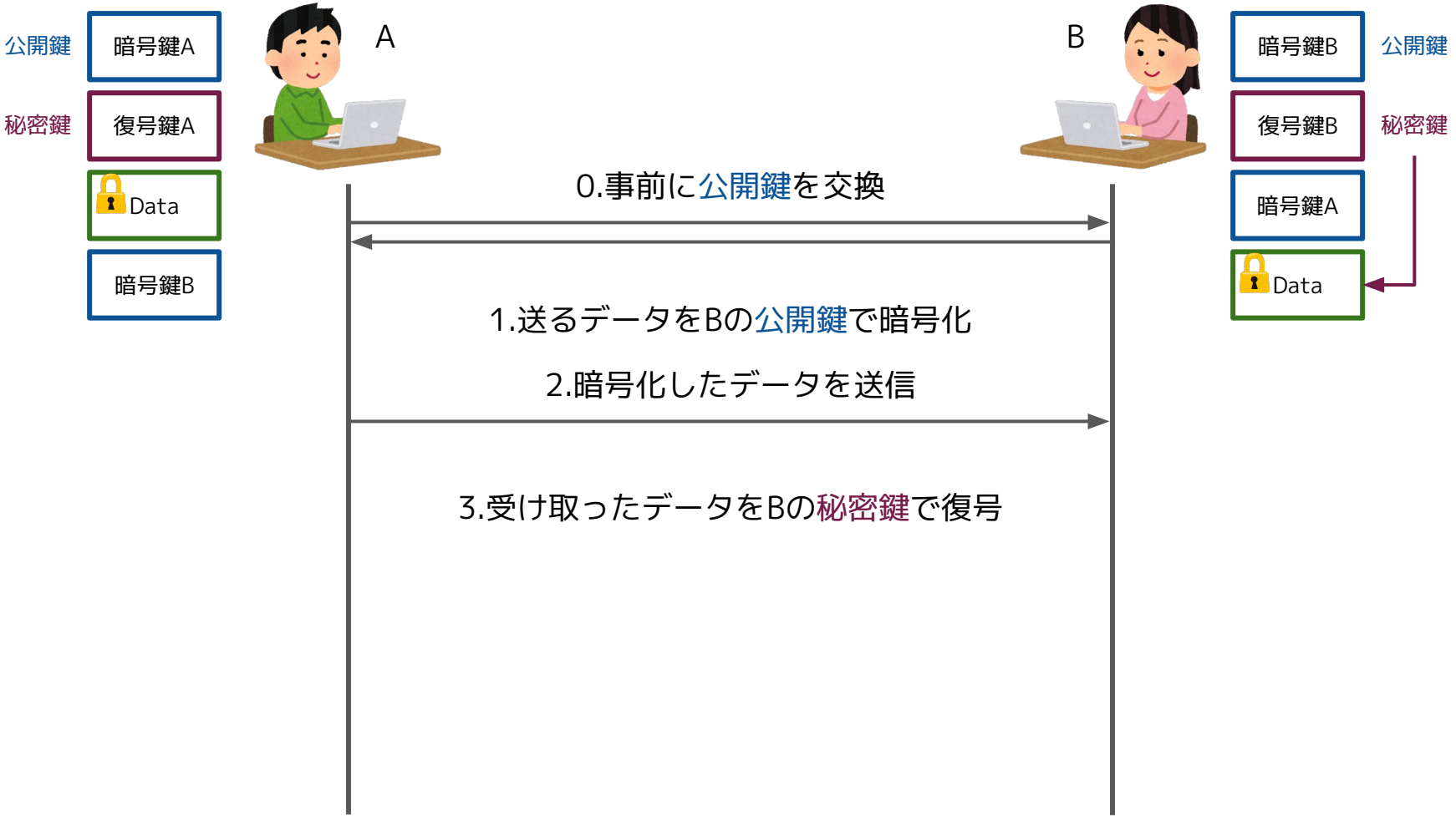
公開鍵

秘密鍵

0.事前に公開鍵を交換

1.送るデータをBの公開鍵で暗号化

2.暗号化したデータを送信

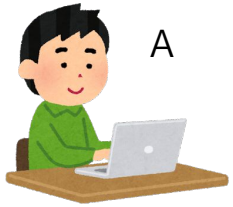

公開鍵
暗号鍵A
秘密鍵
復号鍵A
Data
暗号鍵B
A
B
暗号鍵B
公開鍵
復号鍵B
秘密鍵
暗号鍵A
Data
0.事前に公開鍵を交換
1.送るデータをBの公開鍵で暗号化
2.暗号化したデータを送信
3.受け取ったデータをBの秘密鍵で復号

公開鍵 暗号鍵A

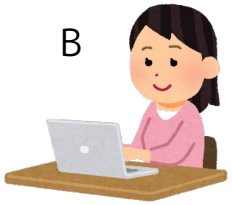
秘密鍵 復号鍵A

Data

暗号鍵B



A



B

暗号鍵B 公開鍵

復号鍵B 秘密鍵

暗号鍵A

Data

0.事前に公開鍵を交換

1.送るデータをBの公開鍵で暗号化

2.暗号化したデータを送信

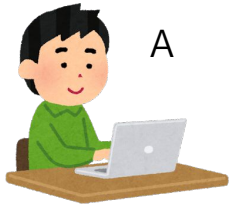
3.受け取ったデータをBの秘密鍵で復号

公開鍵 暗号鍵A

秘密鍵 復号鍵A

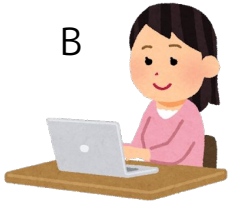
Data

暗号鍵B



A

B



暗号鍵B 公開鍵

復号鍵B 秘密鍵

暗号鍵A

Data

0.事前に公開鍵を交換

1.送るデータをBの公開鍵で暗号化

2.暗号化したデータを送信

3.受け取ったデータをBの秘密鍵で復号

無事，B側でデータをみることができた
どこで盗聴されても元の平文が漏れる心配はない

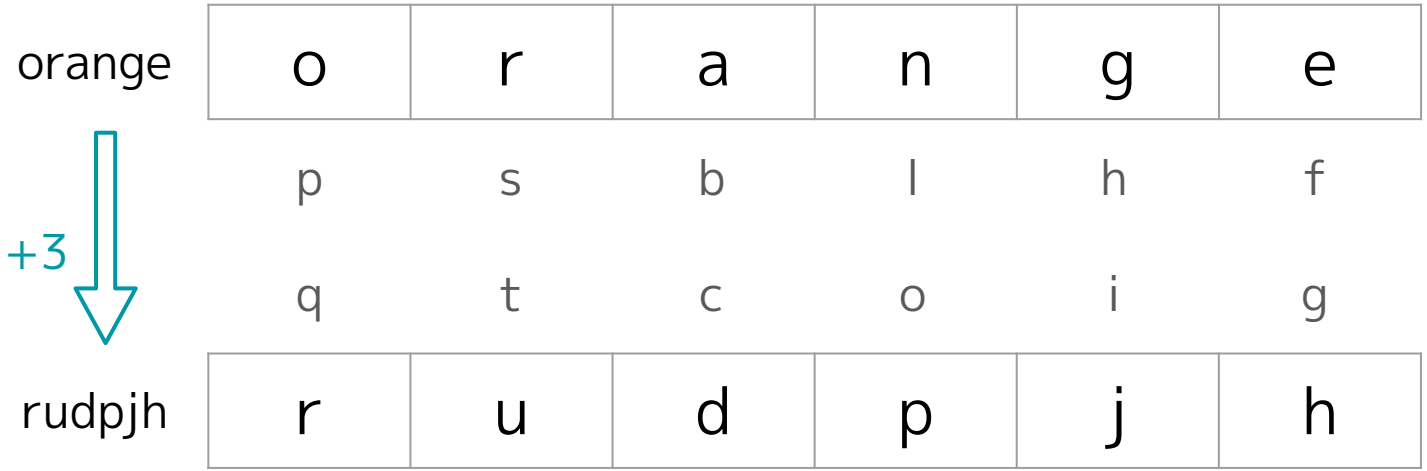
公開鍵暗号方式と呼ばれる方式

暗号アルゴリズムについて

- Caesar暗号（シーザー暗号／カエサル暗号）（換字式暗号）
  - アルファベットを任意の数ずらして暗号化する
  - Caesar暗号は単換字式暗号
- AES暗号
  - 128/192/256bitの鍵をもつブロック暗号アルゴリズム
- RSA暗号
  - 大きな２つの素数の素因数分解が困難なことを利用した暗号アルゴリズム
- ECC（楕円曲線暗号）
  - 離散対数問題と素因数分解などの数学において難解な問題を利用した暗号アルゴリズム
- など

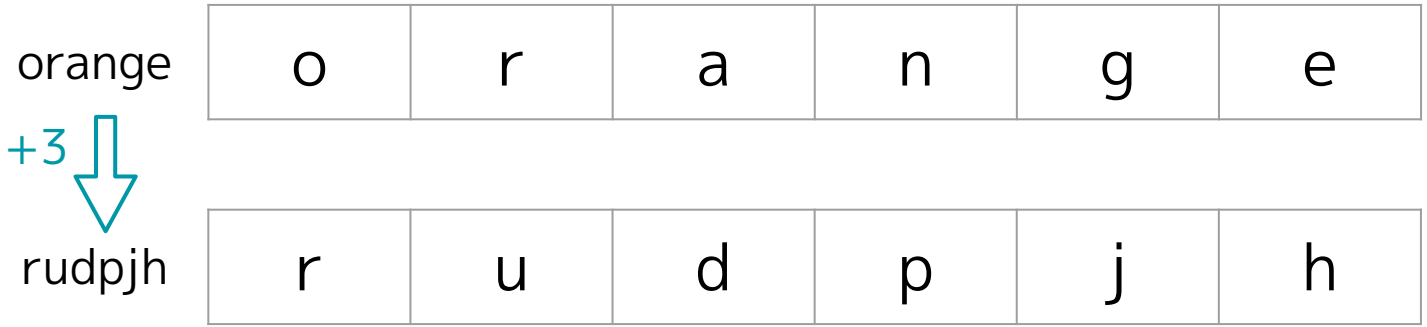
Caesar暗号

平文文字列をアルファベット順に任意の数ずらして暗号化する



Caesar暗号

平文文字列をアルファベット順に任意の数ずらして暗号化する



復号するにはCaesar暗号を使っているという事実と，
いくつずらすかを特定する必要がある

この暗号で13文字ずらすものをROT13という（**Rot**ate by **13**）

ハッシュ値（Hash value）

ハッシュ値とは、元になるデータから一定の計算手順により求められた固定長の値。その性質から**暗号**や認証、**データ構造**などに応用されている。
ハッシュ値を求めるための計算手順のことを**ハッシュ関数**、要約関数、メッセージダイジェスト関数などという。

ハッシュ値（ダイジェスト値）とは - IT用語辞典 e-Words
https://e-words.jp/w/%E3%83%8F%E3%83%83%E3%82%B7%E3%83%A5%E5%80%A4.html

元になるデータから計算される固定長の値

同じデータからは全く同じハッシュ値が求められる

ハッシュ化も広義では暗号化（基本は違う気がする）

ハッシュ関数（SHA256）にかけてみる

```
▶ echo -n "hello" | openssl dgst -sha256
2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824
```

```
▶ echo -n "hallo" | openssl dgst -sha256
d3751d33f9cd5049c4af2b462735457e4d3baf130bcbb87f389e349fbaeb20b9
```

```
▶ echo -n "hellooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo" | openssl dgst -sha256
67d1cfcfff29277a843ad5b3927236712f18169d5acbe23016ea342954d68516
```

```
▶ echo -n "aiueo" | openssl dgst -sha256
fa06926df12aec4356890d4847d43f79101c93548a6b65e4b57bcb651294beef
```

```
▶ echo "aiueo" | openssl dgst -sha256
1dcb283c9d6347574f0a3fb5c7b578303ee6a8b68beb27bf6f50a141f43f0573
```

ハッシュ関数（SHA256）にかけてみる

平文が少しでも違うと，ハッシュ値は全然違うものになる

```
▶ echo -n "hello" | openssl dgst -sha256
2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824
```

```
▶ echo -n "hallo" | openssl dgst -sha256
d3751d33f9cd5049c4af2b462735457e4d3baf130bcbb87f389e349fbaeb20b9
```

ハッシュ関数（SHA256）にかけてみる

平文が少しでも違うと，ハッシュ値は全然違うものになる

```
▶ echo -n "aiueo" | openssl dgst -sha256
fa06926df12aec4356890d4847d43f79101c93548a6b65e4b57bcb651294beef
```

```
▶ echo "aiueo" | openssl dgst -sha256
1dcb283c9d6347574f0a3fb5c7b578303ee6a8b68beb27bf6f50a141f43f0573
```

-n オプションをつけると改行を末尾に入れない

ハッシュ関数（SHA256）にかけてみる

SHA256の場合，平文の文字数に関係なく65文字の[a-z0-9]に写っている

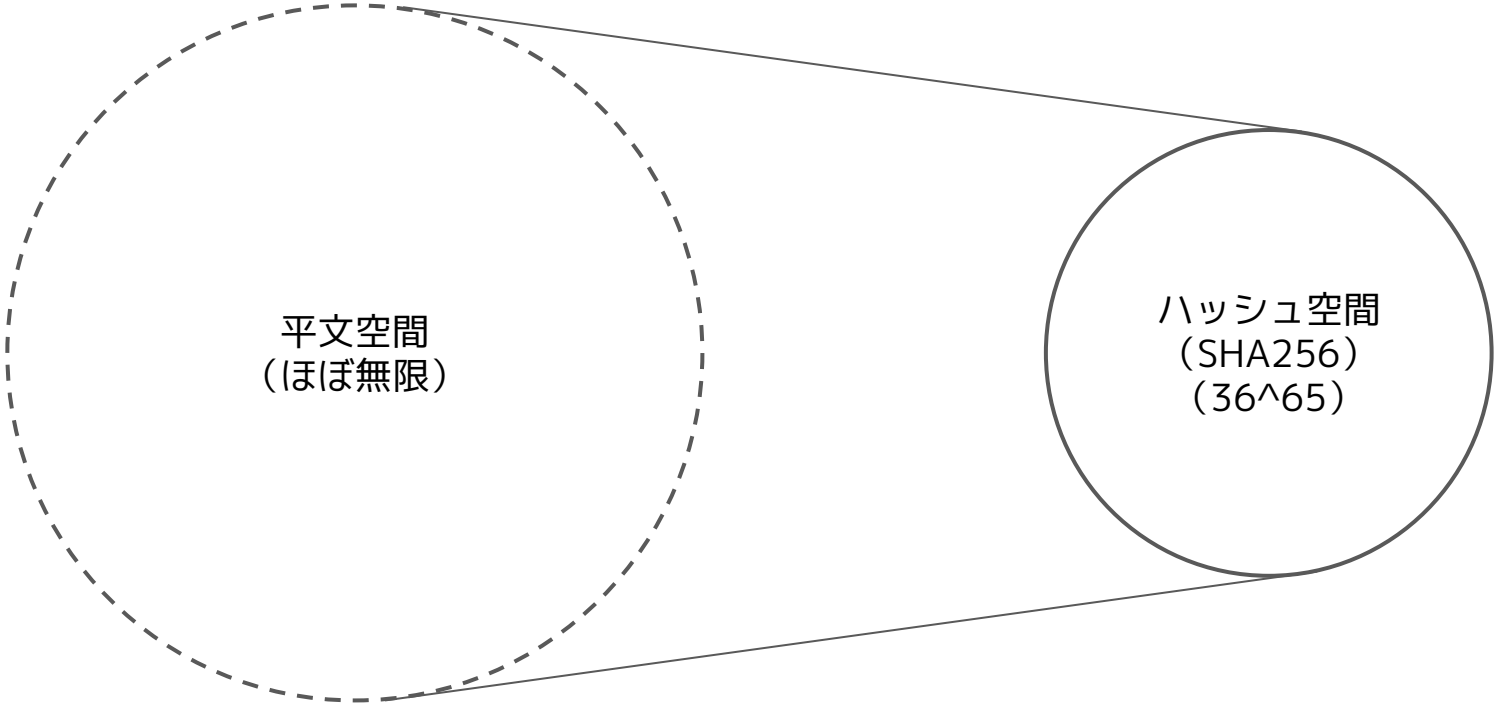
```
▶ echo -n "hello" | openssl dgst -sha256
2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824
```

```
▶ echo -n "hellooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo" | openssl dgst -sha256
67d1cfcfff29277a843ad5b3927236712f18169d5acbe23016ea342954d68516
```

```
~
▶ echo "hello" | openssl dgst -sha256 | wc -c
      65

~
▶ echo -n "hellooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo" | openssl dgst -sha256 | wc -c
      65
```

つまりハッシュ値の方が表現の幅が狭い



ハッシュの用途

ハッシュは重複しうるが実用上問題ないとされており，実際に使われている

（実際には衝突回避や衝突してもいいように実装されている）

- 改竄検出
  - 平文が少しでも違うとハッシュ値は大きく異なることを利用
- 暗号化
  - 適切なハッシュ関数を使えば平文を推測できない
  - 平文が必要ないデータに適用できる（パスワードなど）
- 重複／類似文字列検索
  - 文字列を細かく区切ってハッシュ化して，重複部分や類似部分を探す
- ハッシュテーブル

まとめ

暗号

- 実際にやり取りされるものを隠蔽する仕組み
- 基本的には可逆で，元の平文に戻すことができる

ハッシュ

- 平文の情報を捨てて保存することが可能
- 情報量を落として効率化したり整合性検査などに使われる