暗号化技術

名城大学 理工学部 情報工学科
渡邊研究室4年　土井敏樹

- 本資料は下記書籍を基にして作成されたものです。文書の内容の正確さは保証できない為、正確な知識を求める方は原文を参照してください。

- 題目
  - 情報セキュリティの基本と仕組み
- 著者
  - 相戸浩志
- 発行
  - 2010年
- 発行所
  - 秀和システム

はじめに

- インターネット社会において暗号技術は不可欠
  - ex.SSL(Secure Socket Layer)
  - 情報を暗号化して送受信するプロトコル

- 暗号技術
  - 機密性 – 第三者から読めなくする
  - 完全性 – データが送信されたままの状態で
    改竄されていないことの証明
  - 本人が作成した文書であることの証明

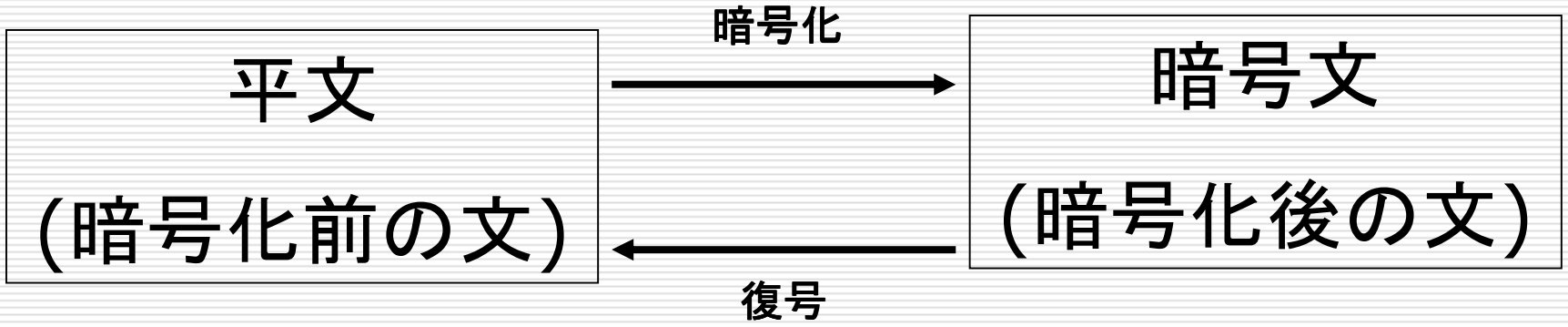
暗号技術

- 暗号技術を支えるもの
  - 共通鍵暗号 - 暗号化と復号に共通鍵を使用
  - 公開鍵暗号 – 暗号化に公開鍵と秘密鍵を使用
  - ハッシュ関数 – 入力データから固定長のビット列を出力する関数

共通鍵暗号

- 暗号化する鍵と復号化する鍵に同じ鍵を用いる暗号方式
- 公開鍵暗号が登場するまでの暗号は全て共通鍵暗号
- シーザー暗号
  - 文字の置き換え(換字)
  - 文字をアルファベット順にずらす

換字式暗号



換字式暗号

暗号化したい文字が

「NOTEの場合」

M	N	S	D
N	O	T	E
O	P	U	F
P	Q	V	G
Q	R	W	H

3文字ずらして暗号化

3文字ずらして復号

暗号は

「QRWH」となる

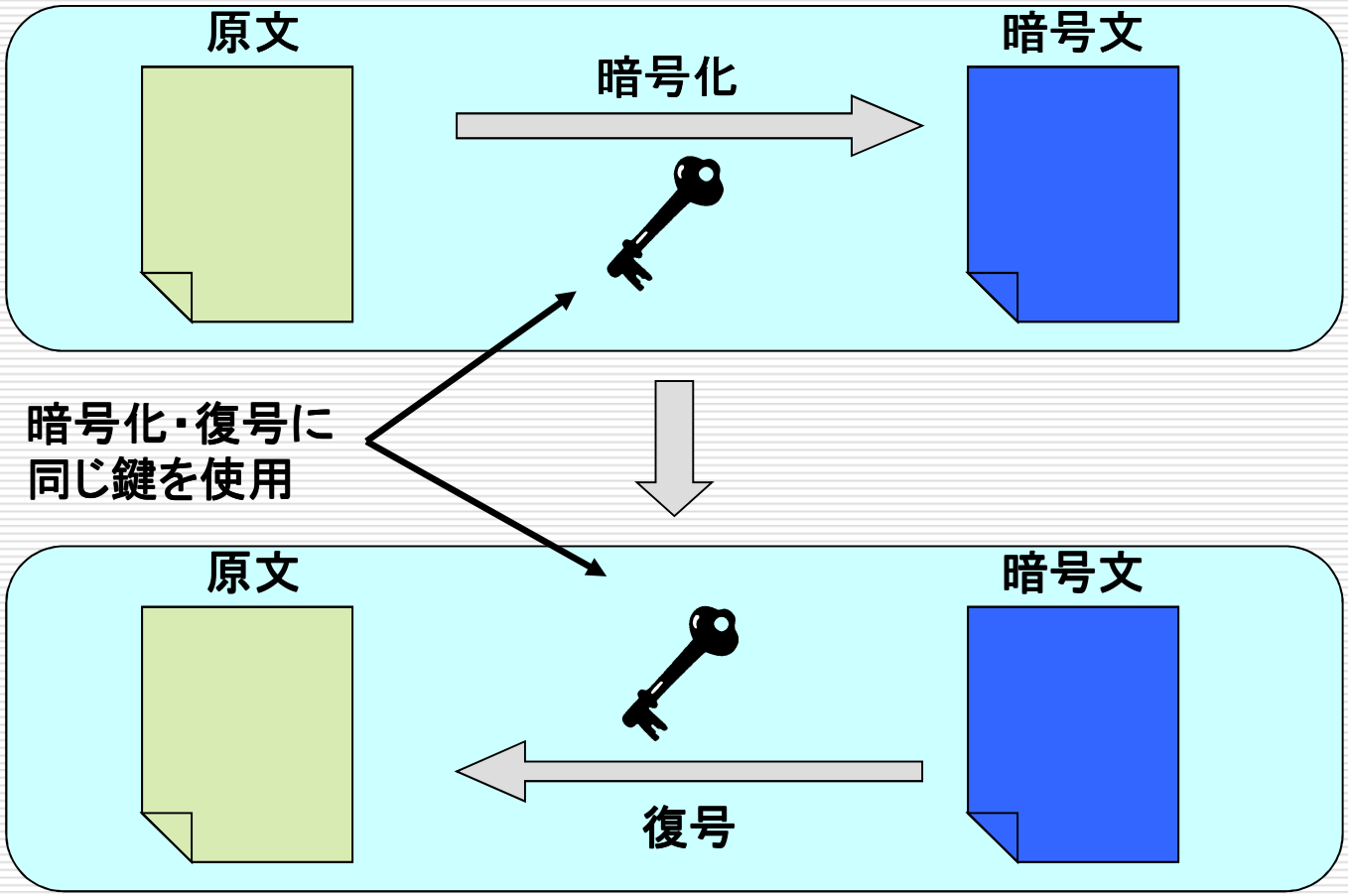
M	N	S	D
N	O	T	E
O	P	U	F
P	Q	V	G
Q	R	W	H

換字式暗号

Ｎ Ｏ Ｔ Ｅ ⟶ Ｑ Ｒ Ｗ Ｈ

- 換字式
  - 平文を意味のない暗号文に置き換える
- 復号に必要な情報
  - 何文字ずらして暗号化したか
  - ある値が分からなければ復号できない
    ⇒このような値の事を鍵という

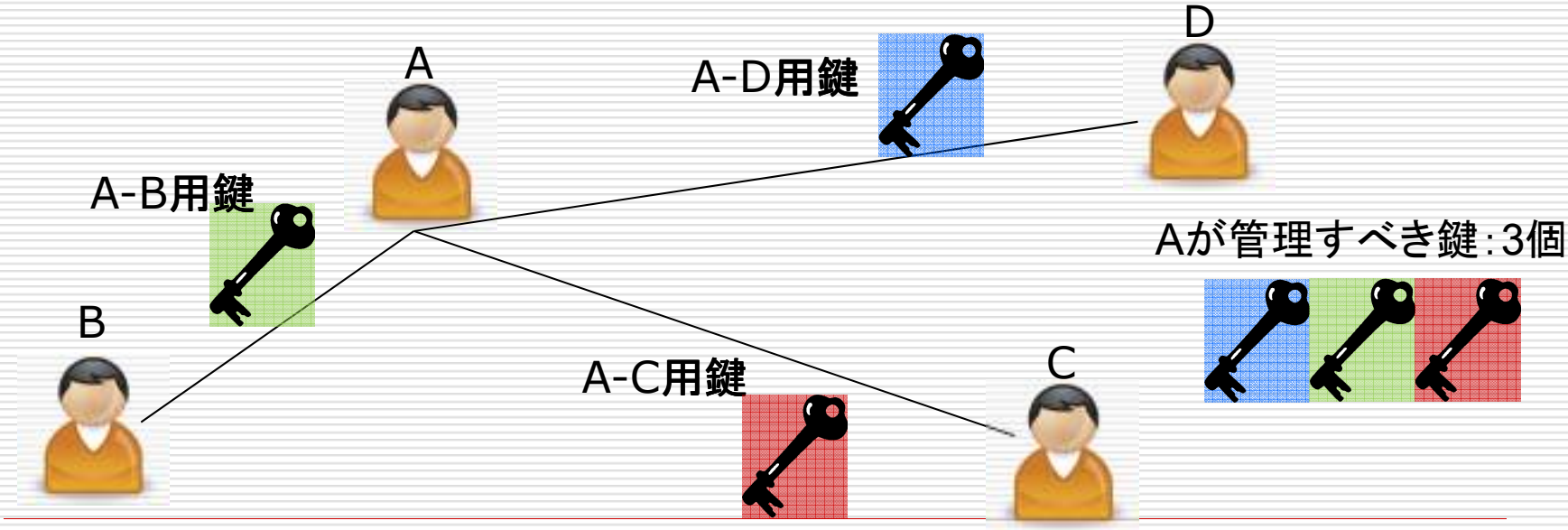
共通鍵暗号方式の暗号化と復号



共通鍵暗号のデメリット

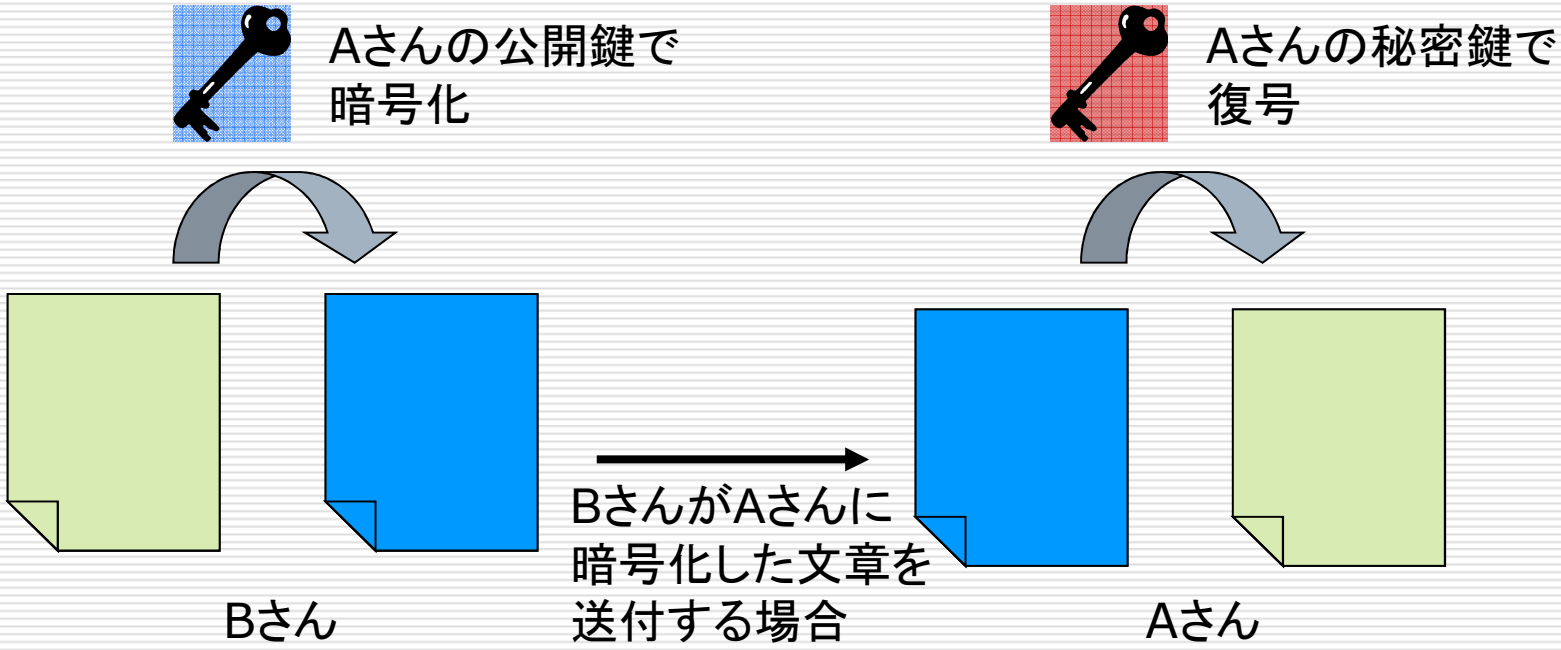
- 共通鍵の安全な送付が必須 ⇒ 鍵の盗難や漏洩の危険性

- 鍵の管理
  - 人数が増えて行くと鍵管理が大変



公開鍵暗号

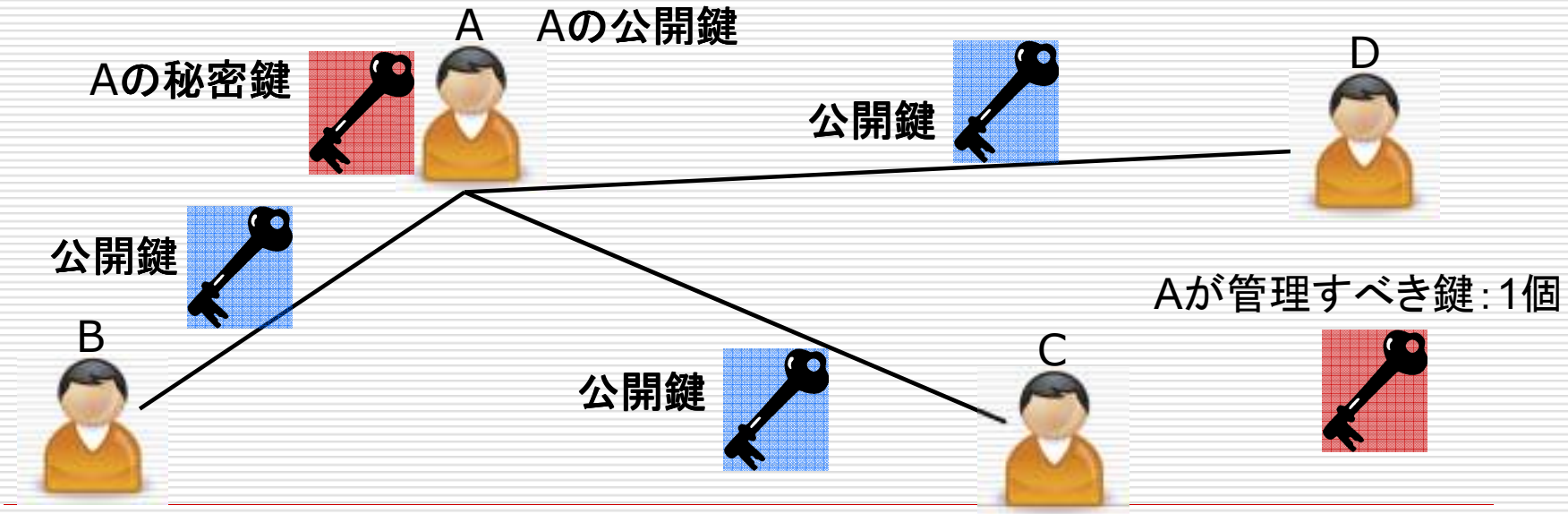
- BさんがAさんに文章を送付する場合



公開鍵暗号のメリット

- 共通鍵の送付が不要 ⇒ 受け渡しの必要なし

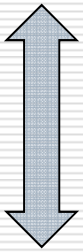
- 鍵の管理
  - 管理しなければいけない鍵が少ない



公開鍵暗号のデメリット

- 公開鍵の改竄を検出できない
  ⇒公開鍵に対して証明が必要である
  ⇒第三者が署名する

- 共通鍵暗号 ⇒ 公開鍵暗号に比べれば暗号化は短時間
  - 共通鍵暗号のメリット



共通鍵暗号と公開鍵暗号の
演算速度の差は3桁以上

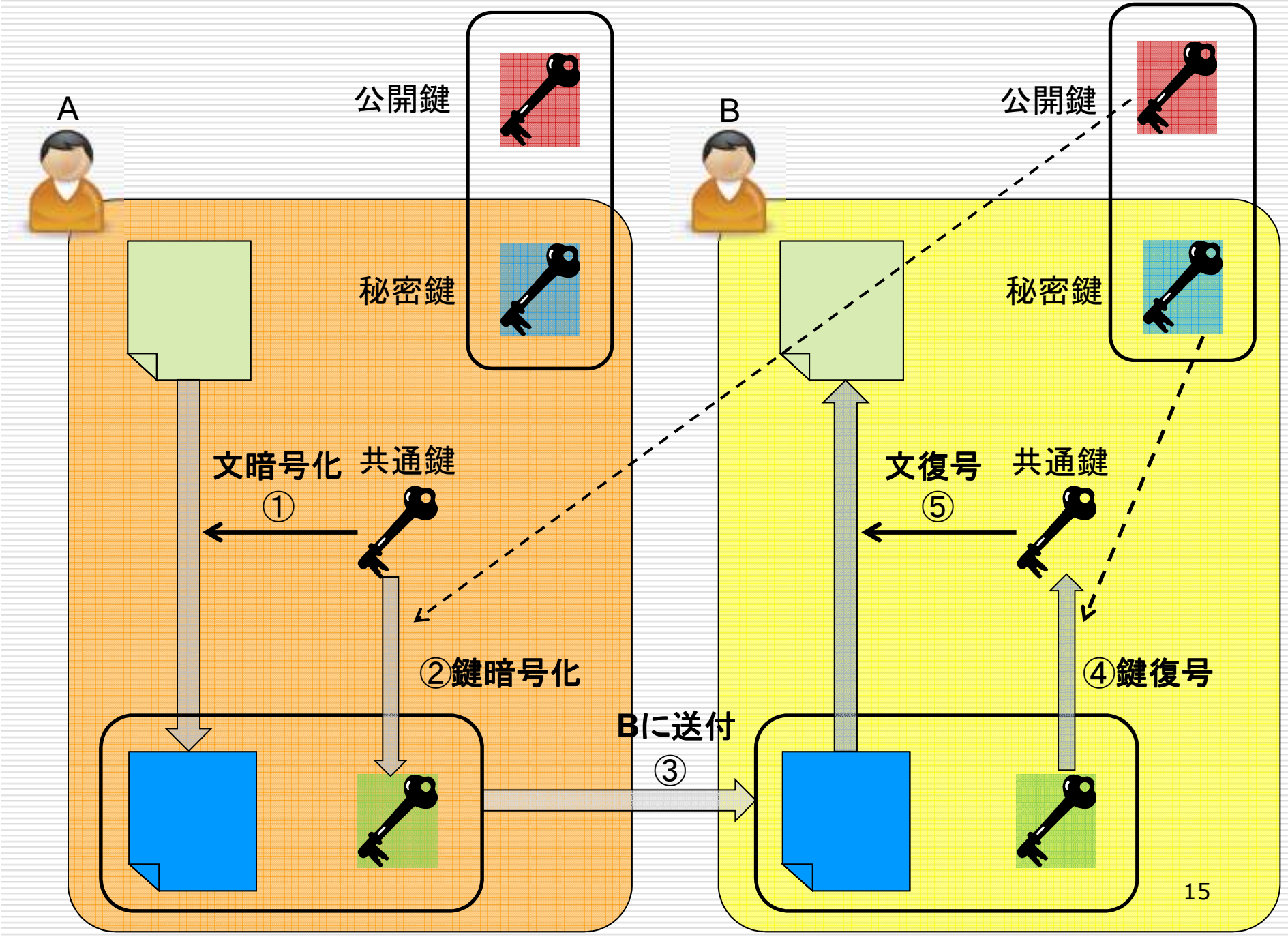
- 公開鍵暗号 ⇒ 共通鍵暗号に比べて時間がかかる
  - 長い情報を暗号化するには不適切

共通鍵暗号と公開鍵暗号

	共通鍵暗号方式	公開鍵暗号方式
方式	暗号鍵＝復号鍵	暗号鍵≠復号鍵
管理すべき鍵	多数(相手毎に必要)	自分の秘密鍵のみ
処理速度	高速	低速
鍵管理	共通鍵を事前に共有する必要	事前に鍵を共有する必要なし 公開鍵への署名が必要

- それぞれメリットとデメリットが存在

  ⇒共通鍵暗号方式と公開鍵暗号方式を組み合わせた方法


A
公開鍵
秘密鍵
文暗号化
共通鍵
①
②鍵暗号化
Bに送付
③
B
公開鍵
秘密鍵
文復号
共通鍵
⑤
④鍵復号

各暗号の代表的方式

- 共通鍵暗号
  - DES(Data Encryption Standard)
  - 1970年代初めに発明
    ↓
  - Triple-DES
  - DESを3回繰り返して安全性を高める

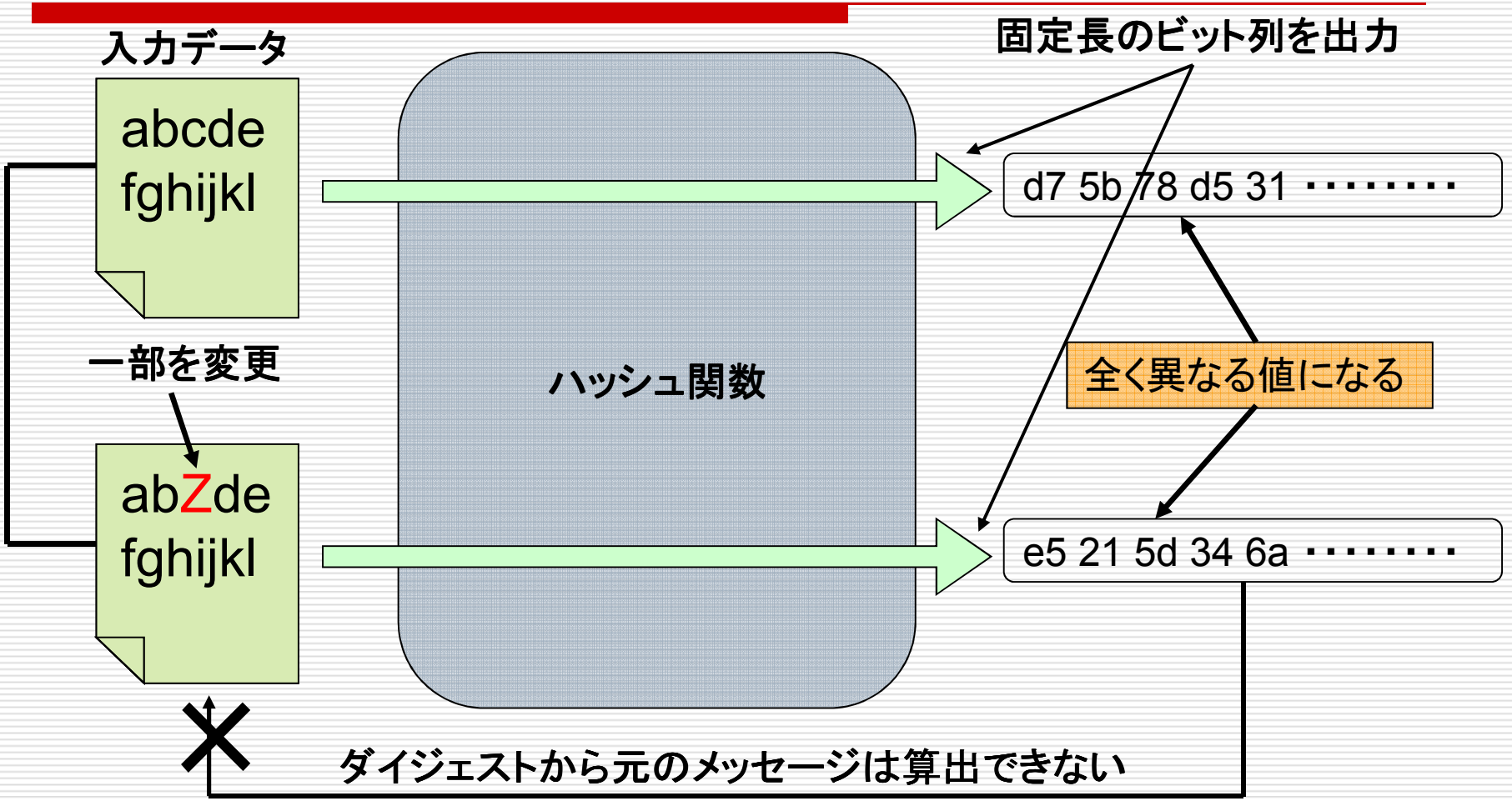
- 公開鍵暗号
  - RSA(Rivest、Shamir,Adleman - 3人の名前)
  - 1977年に発明
  - 大きな数の素因数分解が困難であることを利用

ハッシュ関数

- 仕組み
  - 入力データから固定長のビット列を出力
  - 入力データ(長さ不問) ⇒ 固定長のビット列を出力
  - 出力値を<u>ハッシュ値</u>または<u>メッセージダイジェスト</u>と呼ぶ

- 性質
  - ハッシュ値から元の値を推測できない
  - 元データの変更により異なる
  - 同ハッシュ値になる入力データを見つけるのは困難

ハッシュ関数の出力


入力データ
abcde
fghijkl
一部を変更
abZde
fghijkl
ハッシュ関数
固定長のビット列を出力
d7 5b 78 d5 31 ・・・・・・・・
全く異なる値になる
e5 21 5d 34 6a ・・・・・・・・
ダイジェストから元のメッセージは算出できない

ハッシュ関数の応用例

- パスワードの保存
  - Windows2000以降のパスワード保存に使用
- 改竄検知
  - 設定ファイルやプログラムのハッシュ値から検知
- ワンタイムパスワード(OneTimePassword)
  - 使い捨てパスワードとも呼ばれる
  - 認証の為に1回しか使えない

電子署名

- 元のメッセージに対しハッシュ関数と公開鍵暗号を用いることでメッセージの改竄を検出できる

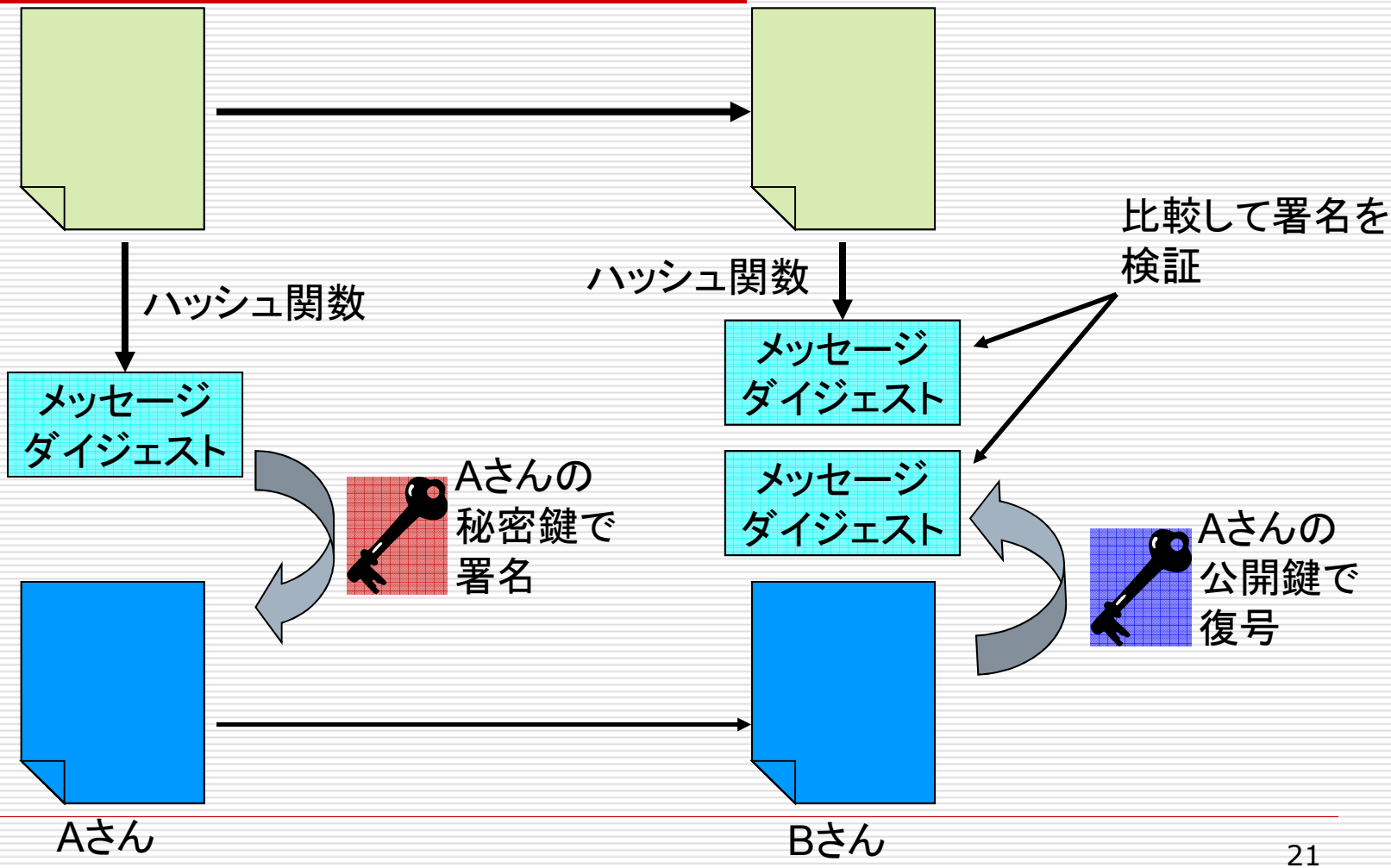
  ⇒デジタル署名(Digital Signature)

  - デジタル署名を生成する事 ⇒ 署名(Sign)
  - デジタル署名が有効である事を確認する

    ⇒ 署名の検証(Verify)

- デジタル署名によって以下の事が確認できる
  - 署名を作成したのが本人であること(本人認証)
  - メッセージが改竄されていない事(完全性)

電子署名の生成と検証


ハッシュ関数
メッセージ
ダイジェスト
Aさんの
秘密鍵で
署名
Aさん
ハッシュ関数
メッセージ
ダイジェスト
比較して署名を
検証
メッセージ
ダイジェスト
Aさんの
公開鍵で
復号
Bさん

電子署名の性質と応用

- メッセージダイジェストのサイズが小さい

  ⇒データを直接暗号化するより高速に通信可能
- 否認防止
  - 署名したことを後から否定するのは不可能
- 署名の応用
  - プログラムが改竄されていないことの証明
- コードサイニング
  - ActiveXやプラグインの配布に使用
  - 不正な変更が加えられていないことの証明

参考文献

- 「情報セキュリティの基本と仕組み」
  - 相戸浩志著
  - 秀和システム,2010年
- SSL
  - http://e-words.jp/w/SSL.html
- 公開鍵暗号と共通鍵暗号の概要
  - http://dev.sbins.co.jp/cryptography/cryptography02.html

まとめ

- 暗号技術を支える3つの技術を紹介
  - 共通鍵暗号
  - 公開鍵暗号
  - ハッシュ関数

END