暗号利用モードの基礎と最近の研究事例

日本銀行金融研究所
情報セキュリティ・セミナー
「共通鍵暗号の安全な使い方
～暗号利用モードの基本とリスク～」

國廣昇

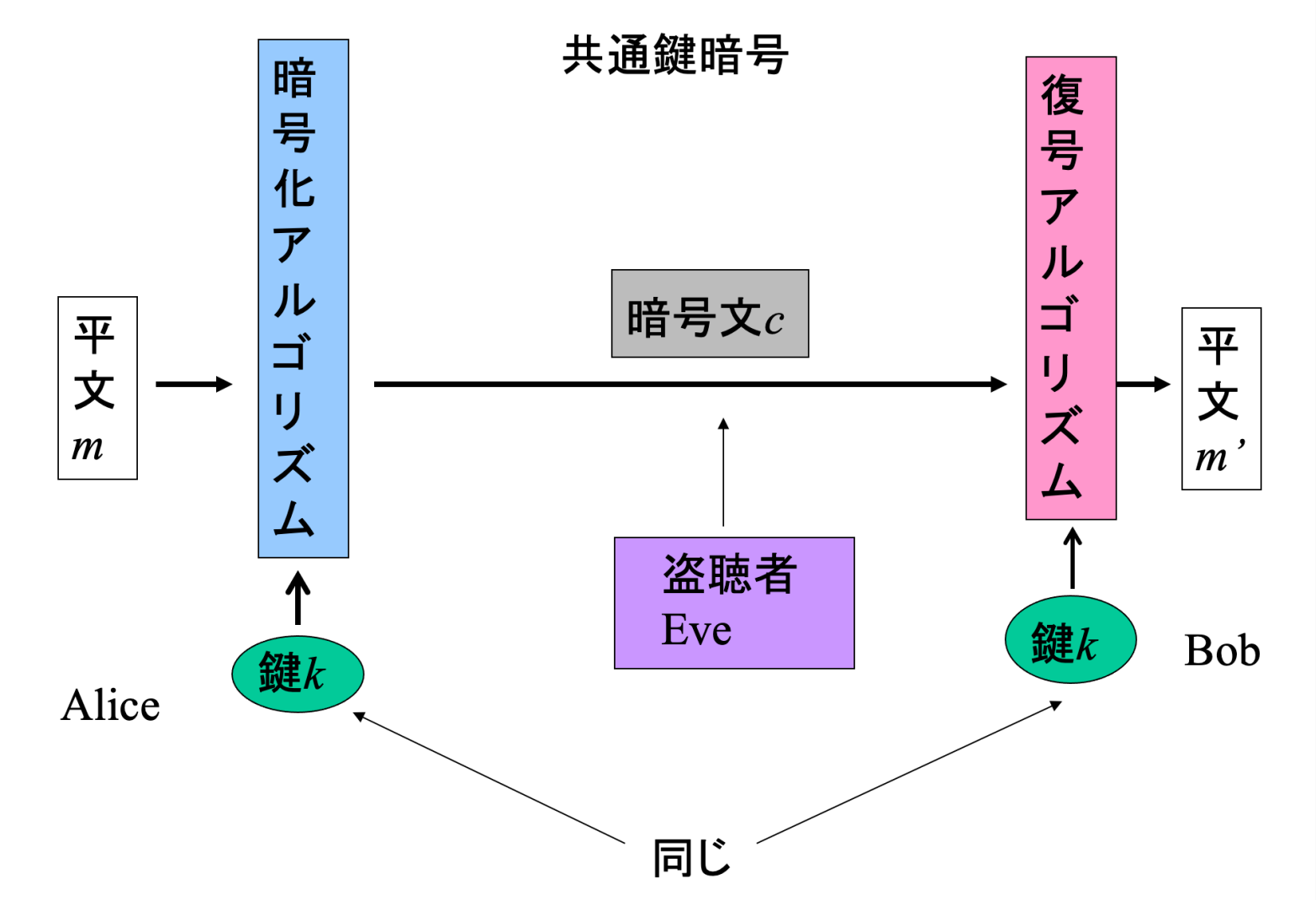
2024 年 12 月 16 日

筑波大学

今日の予定

1. 共通鍵暗号と公開鍵暗号（と鍵共有）
2. 共通鍵暗号
   - One-time pad
   - ブロック暗号
3. 暗号利用モード
4. ECB モードの脆弱性

共通鍵暗号


共通鍵暗号
平文 m
暗号化アルゴリズム
暗号文c
復号アルゴリズム
平文 m'
盗聴者 Eve
鍵k
鍵k
Alice
Bob
同じ

共通鍵暗号

前提

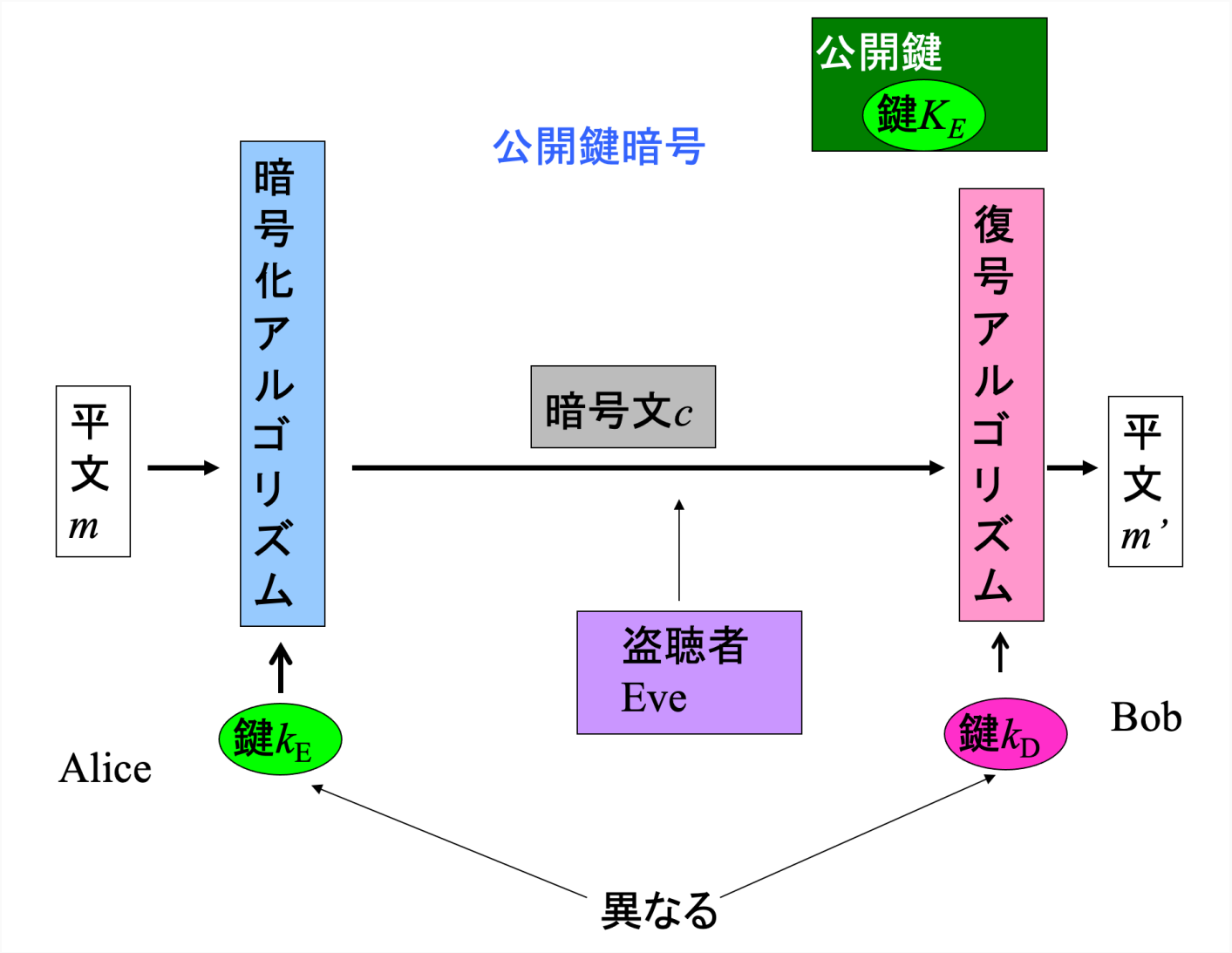
- 送信者と受信者の間で共通の鍵をあらかじめ共有する
  - （どうやって？ $\Rightarrow$ 公開鍵暗号，鍵共有を用いる）
- 同じ鍵で暗号化，復号を行う

- メリット：処理が高速
- デメリット：鍵共有をどうやって行うか？

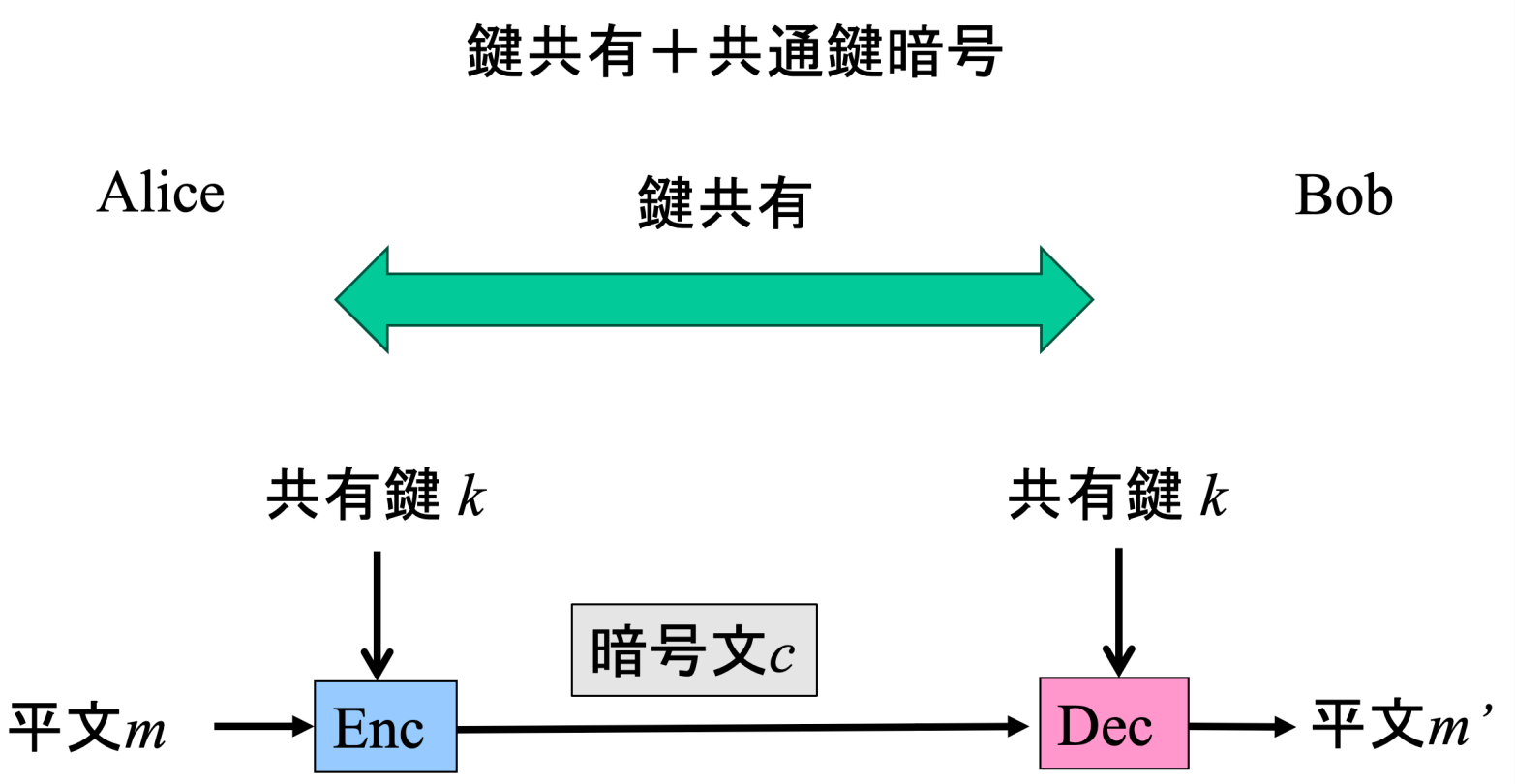
大きく分けて，

- ストリーム暗号
- ブロック暗号

公開鍵暗号


公開鍵
鍵K_E
公開鍵暗号
暗号化アルゴリズム
復号アルゴリズム
平文 m
暗号文c
平文 m'
盗聴者 Eve
鍵k_E
鍵k_D
Alice
Bob
異なる

鍵共有＋共通鍵暗号



TLS1.3 では，ECDH 鍵共有＋認証付き暗号を使う

電子政府における調達のために参照すべき暗号のリスト（CRYPTREC 暗号リスト）

共通鍵暗号

128 ビットブロック暗号

- AES, Camellia

暗号利用モード

秘匿モード

- CBC, CFB, CTR, OFB, XTS（注あり）

認証付き秘匿モード (*)

- CCM, GCM（注あり）

(*) CRYPTREC 暗号リスト掲載のブロック暗号を、認証付き秘匿モードと組み合わせて、「認証暗号」として使うことができる。

メッセージ認証コード

- CMAC
- HMAC

認証暗号

- ChaCha20-Poly1305

詳しくは，

`https://www.cryptrec.go.jp/list.html`

を御覧ください．

目次

1. 共通鍵暗号と公開鍵暗号（と鍵共有）
2. **共通鍵暗号**
   - One-time pad
   - **ブロック暗号**
3. 暗号利用モード
4. ECB モードの脆弱性

One-time pad

- 1 ビットのメッセージ $m \in \{0, 1\}$ を暗号化して送りたい
- 1 ビットの鍵 $k \in \{0, 1\}$ を送信者と受信者で共有しているとする
- 暗号化：$c \leftarrow m \oplus k \in \{0, 1\}$
- 復号：$m \leftarrow c \oplus k \in \{0, 1\}$

排他的論理和 $\oplus$

$0 \oplus 0 = 1 \oplus 1 = 0, 0 \oplus 1 = 1 \oplus 0 = 1$

$x \oplus 0 = x, x \oplus 1 = \bar{x}, x \oplus x = 0, x \oplus \bar{x} = 1$

$(x \oplus y) \oplus z = x \oplus (y \oplus z),\ x \oplus y = y \oplus x$

復号できるか？

$c \oplus k = (m \oplus k) \oplus k = m \oplus (k \oplus k) = m \oplus 0 = m$

安全か？

- 攻撃者は，c から m もしくは k を復元したい．
- $c = 0$ のとき，$(m, k) = (0, 0), (1, 1)$ のどちらか．
  - しかし，この 2 つを区別することができないので，$m = 0$ か $m = 1$ かを当てることはできない．
- $c = 1$ のとき，$(m, k) = (0, 1), (1, 0)$ のどちらか．
  - しかし，この 2 つを区別することができないので，$m = 0$ か $m = 1$ かを当てることはできない．
- ので，c を見ても，m の値を確率 $1/2$ でしか当てることができない．

安全性と問題点

安全性

適切に使用する状況において，無条件に安全

- 共有鍵をランダムに生成
- 共有鍵の使い回しをしない，など

問題点

- 共有鍵が長すぎる
- 送りたいメッセージと同じ長さの鍵が必要

使い回しをすると...

２つの平文 m_1, m_2 に対して，同じ鍵 k で暗号化したとする．
$c_1 = m_1 \oplus k, c_2 = m_2 \oplus k$ を攻撃者は得る．
$c_1 \oplus c_2 = (m_1 \oplus k) \oplus (m_2 \oplus k) = m_1 \oplus m_2$ が成り立つ．
攻撃者は，平文に関する非自明な関係を得る．

ブロック暗号

メッセージを n ビットのブロックに区切り，処理．

基本形

入力：n ビットのメッセージ M，k ビットの鍵 K
出力：n ビットの暗号文 C

暗号化

$$C = \mathsf{Enc}(K; M)$$

復号

$$M' = \mathsf{Dec}(K; C)$$

攻撃

- 攻撃者は，暗号文 C から M もしくは K を復元したい．

代表的な共通鍵暗号方式

DES

- 米国の元標準暗号．
- ブロック長 n は 64，鍵長 k は 56．
- 既に安全ではないとされており，使用は推奨されていない．

AES

- 米国標準技術局 (NIST) が，2000 年に標準化．CRYPTREC による「電子政府推奨暗号リスト」の一つ．
- ブロック長 n は 128，鍵長 k は $128, 192, 256$ の 3 つが利用可

Camellia

- 2000 年に，NTT と三菱電機により開発．CRYPTREC による「電子政府推奨暗号リスト」の一つ．
- ブロック長 n は 128，鍵長 k は $128, 192, 256$ の 3 つが利用可

目次

1. 共通鍵暗号と公開鍵暗号（と鍵共有）
2. 共通鍵暗号
   - One-time pad
   - ブロック暗号
3. **暗号利用モード**
4. ECB モードの脆弱性

暗号利用モード

暗号利用モード

- 多くの応用では，ブロック長 n より長いメッセージを暗号化する必要がある．
- そのため，メッセージを n ビット毎に区切って，処理することが多い（短い場合は，適切に padding する）
- どのように暗号化を組み合わせるかが重要

ECB モード

一番素朴なもの: ECB (Electronic Codebook) モード

メッセージ $M = M_1||M_2||\cdots||M_t$ と分割する.
各 M_i は n ビット.

暗号化：

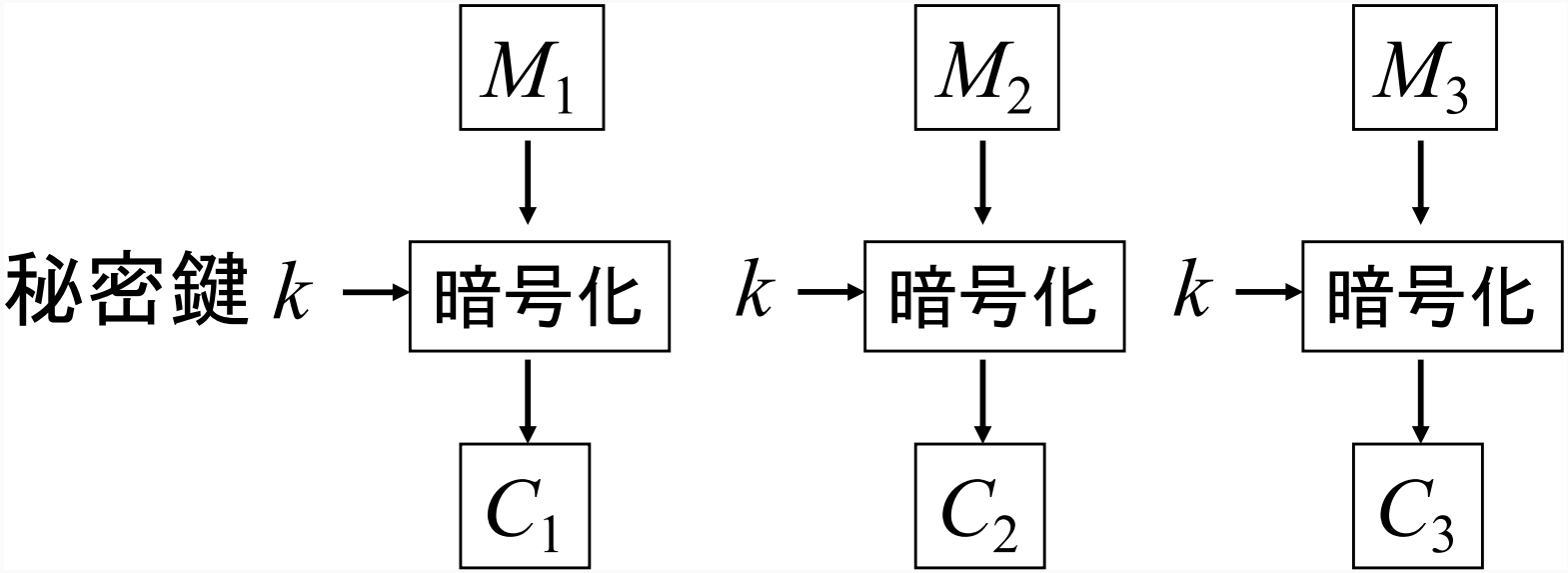
$$C_1 = \mathsf{Enc}(K; M_1),\ C_2 = \mathsf{Enc}(K; M_2),\ \ldots,\ C_t = \mathsf{Enc}(K; M_t)$$

復号：

$$M'_1 = \mathsf{Dec}(K; C_1),\ M'_2 = \mathsf{Dec}(K; C_2),\ \ldots,\ M'_t = \mathsf{Dec}(K; C_t)$$

ECB モードの概略図

暗号化



ECB モードの安全性

暗号方式自身が安全であっても，ECB モードは安全ではない．

- 暗号文を複製されても，
- 暗号文の順序を入れ替えられても，

不正を検知できない．

Alice	100
Bob	70
Charlie	90
David	50

$\overset{\text{暗号化}}{\Longrightarrow}$

Enc(Alice)	Enc(100)
Enc(Bob)	Enc(70)
Enc(Charlie)	Enc(90)
Enc(David)	Enc(50)

Alice$||100||$ Bob$||70||$ Charlie$||90||$ David$||50$ $\overset{\text{暗号化}}{\Longrightarrow}$

Enc(Alice) $||$Enc$(100)||$ Enc(Bob) $||$Enc$(70)||$ Enc(Charlie) $||$Enc$(90)||$ Enc(David) $||$Enc(50)

David の立場で考える

- データのフォーマットは既知（成績はアルファベット順に並んでいるなど），
- Alice が成績優秀であること，

がわかっているとする．

以下のように改ざんすることができる．

Enc(Alice)	Enc(100)
Enc(Bob)	Enc(70)
Enc(Charlie)	Enc(90)
Enc(David)	Enc(100)

Enc(Alice)	Enc(50)
Enc(Bob)	Enc(70)
Enc(Charlie)	Enc(90)
Enc(David)	Enc(100)

ECB モードの特徴

- 秘密鍵を知らないので，暗号文ブロックから，平文を復元することはできない．
- 秘密鍵を知らないので，新たな暗号ブロックを作ることはできない．
- ただし，コピーはできる．
- 同じ平文ブロックは，同じ暗号文ブロックになる．
- 同じ暗号文ブロックは，同じ平文ブロックに対応する．
- 暗号文ブロックから，平文ブロックに関する情報が漏洩する．
  $\Rightarrow$ 使用は推奨されていない．

対策

同じ平文ブロックでも，異なる暗号文ブロックが出力されるようにしたい．

CBC モード

CBC (Cipher Block Chaining) モード

メッセージ $M = M_1||M_2||\cdots||M_t$ と分割する.
各 M_i は n ビット.

暗号化：初期化ベクトル IV を設定

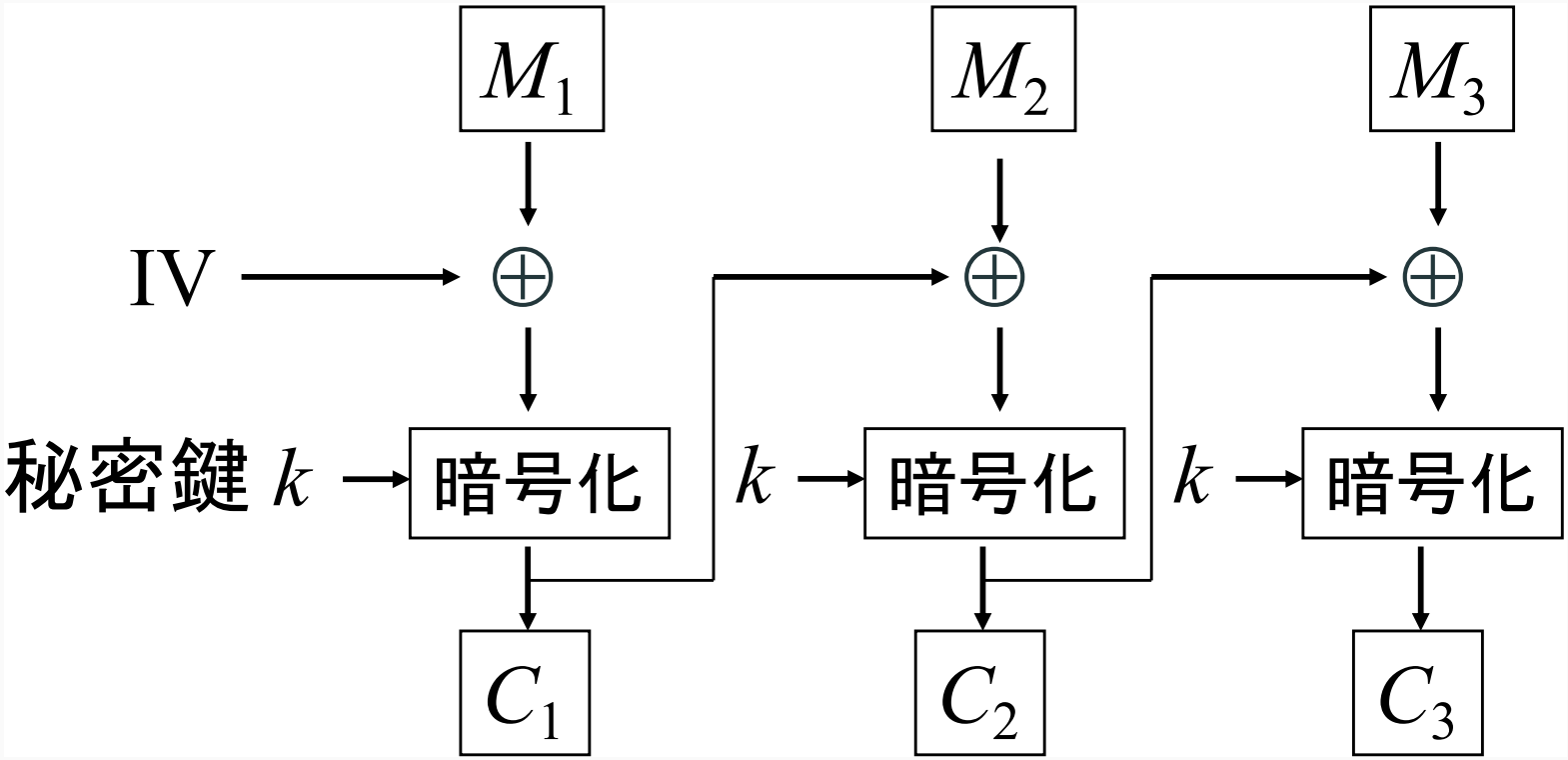
$$C_0 = \mathsf{IV},\ C_1 = \mathsf{Enc}(K; M_1 \oplus C_0),$$
$$C_2 = \mathsf{Enc}(K; M_2 \oplus C_1),\ \ldots,\ C_t = \mathsf{Enc}(K; M_t \oplus C_{t-1})$$

暗号文：$C_0,\ C_1,\ C_2,\ \ldots,\ C_t$

復号：

$$M'_1 = \mathsf{Dec}(K; C_1) \oplus C_0,$$
$$M'_2 = \mathsf{Dec}(K; C_2) \oplus C_1,\ \ldots,\ M'_t = \mathsf{Dec}(K; C_t) \oplus C_{t-1}$$

CBC モード（暗号化）の概略図



ストリーム暗号としての使用

共通鍵暗号をストリーム暗号として使うモード

CFB モード，OFB モード，CTR モード

CTR モードの暗号化

$$C_0 = \mathsf{ctr},\ C_1 = m_1 \oplus \mathsf{Enc}(K; \mathsf{ctr}+1),$$
$$C_2 = m_2 \oplus \mathsf{Enc}(K; \mathsf{ctr}+2),\ \ldots, C_t = m_t \oplus \mathsf{Enc}(K; \mathsf{ctr}+t)$$

暗号文： $(C_0, C_1, \ldots, C_t)$

CTR モードの復号

$$\mathsf{ctr} = C_0,\ m_1 = C_1 \oplus \mathsf{Enc}(K; \mathsf{ctr}+1),$$
$$m_2 = C_2 \oplus \mathsf{Enc}(K; \mathsf{ctr}+2),\ \ldots, m_t = C_t \oplus \mathsf{Enc}(K; \mathsf{ctr}+t)$$

目次

1. 共通鍵暗号と公開鍵暗号（と鍵共有）
2. 共通鍵暗号
   - One-time pad
   - ブロック暗号
3. 暗号利用モード
4. ECB モードの脆弱性

ECB モードの脆弱性：事例 1

Zoom

- Zoom では，version 5.0 より前のものは，ECB モードを利用していた．
- 2020 年 5 月に，AES-GCM を使用するように修正された．

どのような暗号化が使われているかは，
設定 $\Rightarrow$ 統計情報 $\Rightarrow$ 暗号化から確認可能．

2024 年 12 月現在，AES-256-GCM が使われている

ECB モードの脆弱性：事例 2

`https://mega-awry.io/` の内容に基づいています．

MEGA

- Data Stroage サービス
- ユーザは，クラウドサーバにファイルを保管
- E2E 暗号化を実現（クラウドサーバも，ファイルの中身はわからない）
- ユーザは複数端末で使用することを想定
- 自身のパスワードですべてをコントロール（You control everything with your password.）マスター鍵として機能する．

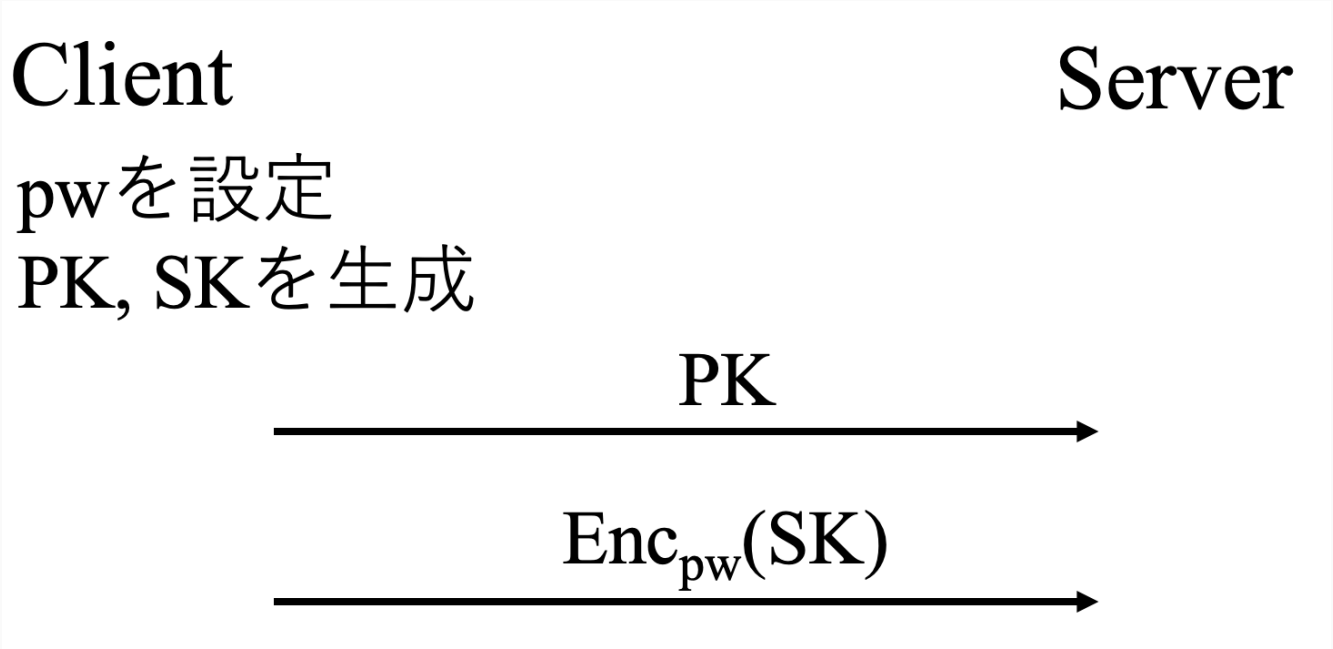
MEGA の脆弱性

- 2022 年 3 月 24 日に，ETH Zurich の研究グループは，MEGA に脆弱性を報告
- 3 つの対策法を提示
- MEGA は，2022 年 6 月 22 日に Update を実施（今回紹介する攻撃は対策済）.

準備

Client は，

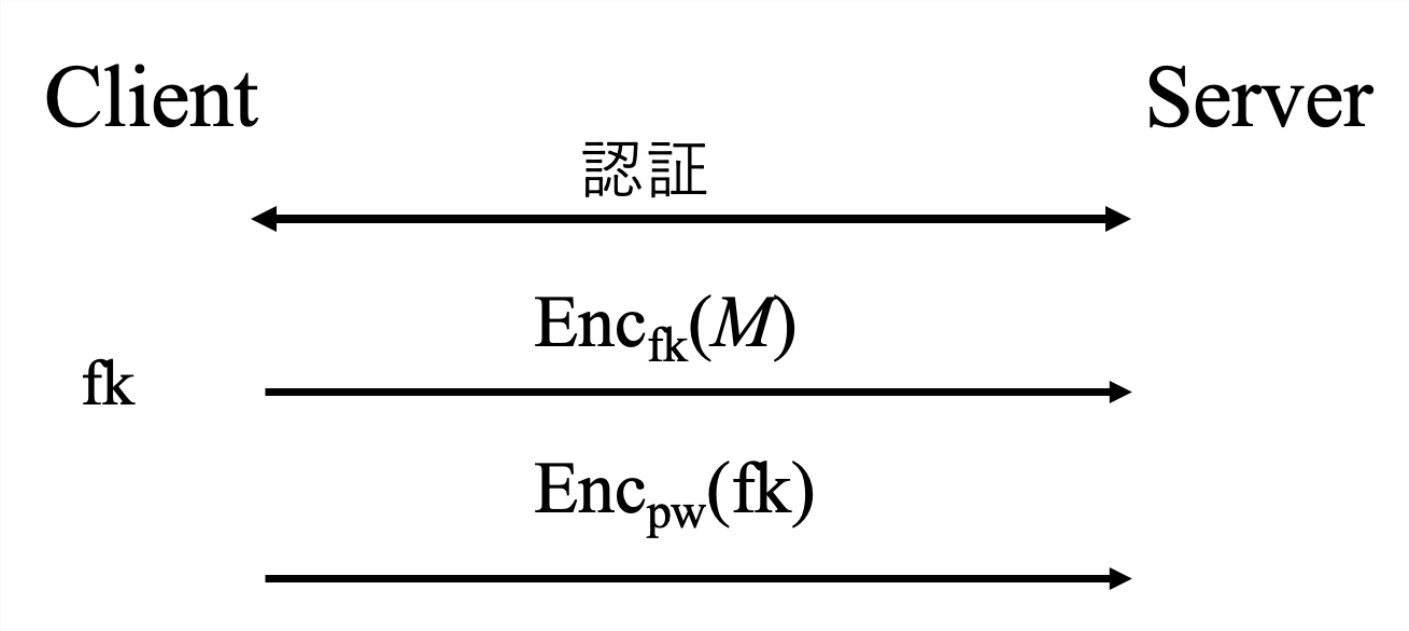
1. 自身で password `pw` を設定
2. RSA 暗号の暗号化鍵 `PK`，秘密鍵 `SK` を生成
3. 暗号化鍵 `PK` を Server に upload
4. `pw` を鍵として，RSA 秘密鍵 `SK` を共通鍵暗号により暗号化し，Server に送信（他の端末で使用できるようにするため）



ファイルの暗号化＆アップロード

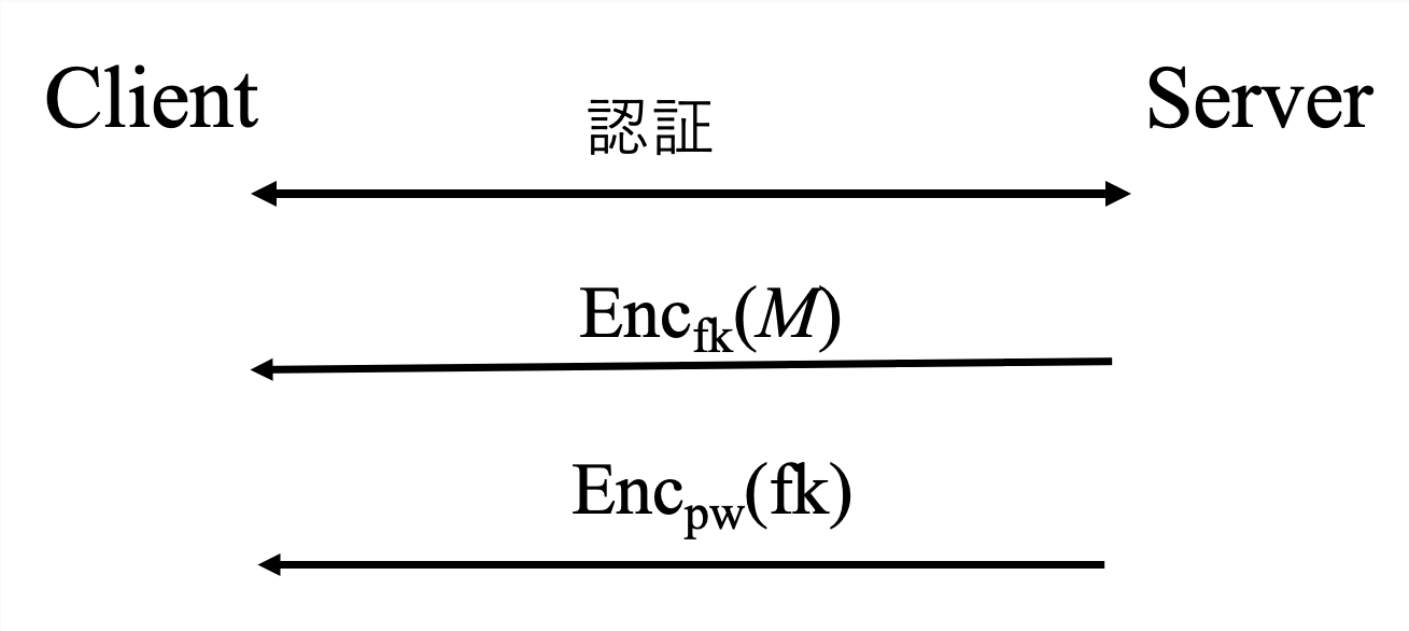
1. Client と Server は相互認証する
2. Client は file key `fk` を生成し，ファイル M を暗号化
3. Client は暗号化されたファイルを Server にアップロード
4. Client は password を使用して，file key `fk` を暗号化し，これを Server にアップロード

暗号化されたファイルの鍵もアップロードするのは，他の端末からアクセスできるようにするため．



ファイルのダウンロード＆復号（別の端末）

1. Cleint と Server は相互認証
2. 暗号化されたファイルと file key を取得する
3. password を使って file key を復号する
4. file key を使用してファイルを復号する



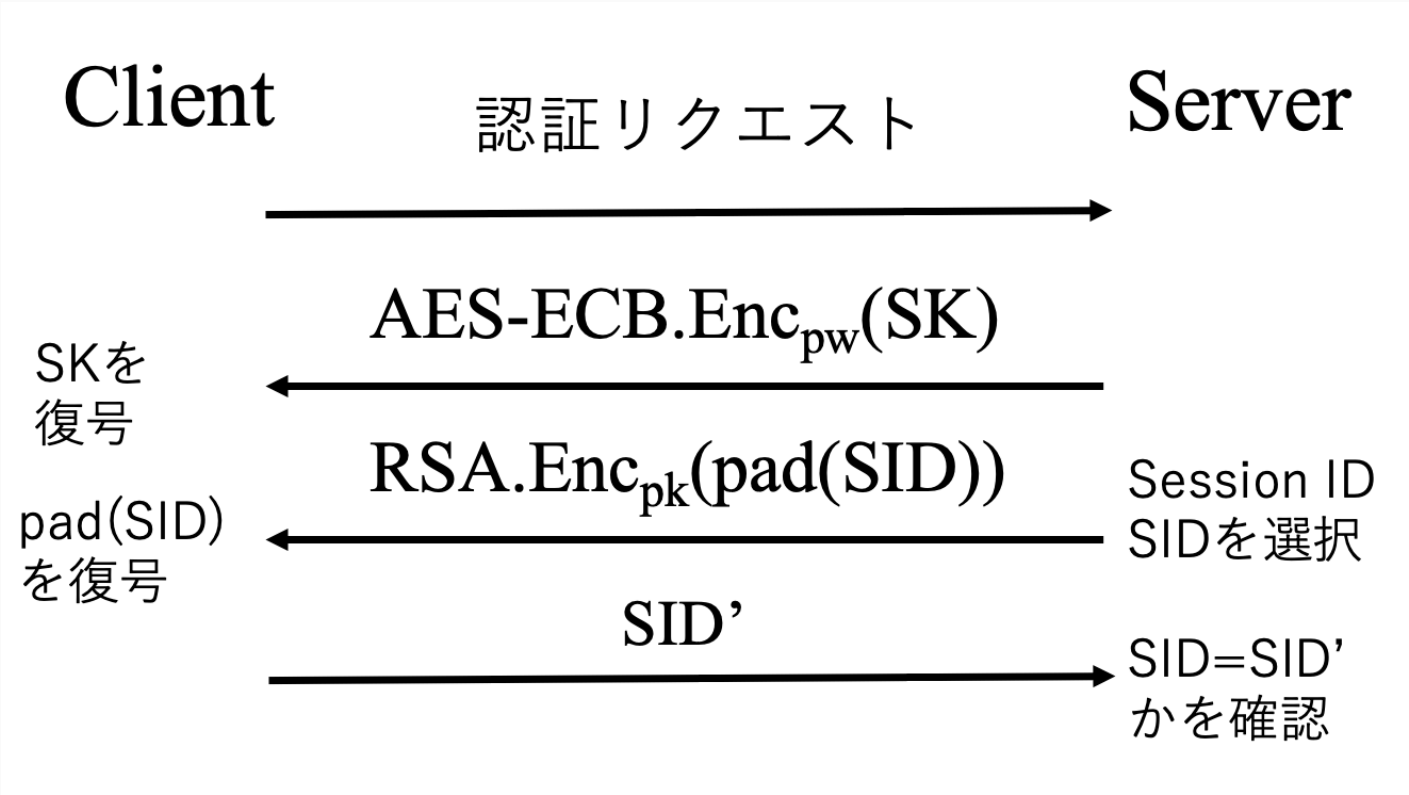
MEGA の攻撃の一つ

攻撃者は Server．Server が，ユーザの RSA 秘密鍵を復元する攻撃

- 認証プロトコルを悪用する

認証プロトコル

Server は，ユーザが password を知っているかを確認する



ECB 暗号化は書き換えが可能

- ブロックが独立して暗号化されていることを利用
- 一つの暗号文ブロックの改変により，（他の平文に影響を与えることなく），対応する平文ブロックを改変することが可能
- 攻撃を検知することができない

$$ct_1 \| ct_2 \| \cdots \| ct_{40} \| ct_{41}$$
$$= Enc(pt_1) \| Enc(pt_2) \| \cdots \| Enc(pt_{40}) \| Enc(pt_{41})$$

書き換え

$$ct_1 \| ct_2 \| \cdots \| ct'_{40} \| ct_{41}$$

Dec

$$pt_1 \| pt_2 \| \cdots \| v \| pt_{41}$$

鍵復元攻撃のアイデア

- RSA 秘密鍵の一部を改変する．Client の返答（SID）を観測して，秘密鍵の情報を得る
- 認証プロトコル中の Client は復号オラクルとして機能
- 悪意のある Server は暗号文を適切に選び，Client の復号結果を利用して，RSA 秘密鍵の１ビットを得る．
- この攻撃により，悪意のある Server は 512 回のログイン試行で RSA 秘密鍵を復元可能
- 後に 6 回にまで削減されている

攻撃２

- すべてのファイルは file key `fk` で暗号化されているので，Server はファイルは復号することはできない（ように見える）
- RSA 秘密鍵 `SK` 自身は，攻撃１により，攻撃者は知っている．
- file key `fk` と `SK` は，同じ鍵で AES-ECB により暗号化されていることに注意
- 認証の際に，`SK` の暗号文ではなく，`fk` の暗号文を Cut and Paste して送ることにより，`fk` を復元できることが可能

対処法のいくつか

- 秘密鍵の分離
  - `pw` の使いまわしが問題．異なるサービスには異なる鍵を使うべき．鍵導出関数を利用し，異なる秘密鍵を使用する．
- AES-ECB ではなく，認証暗号（例えば，AES-GCM）を使う．
  - 暗号文の改ざんを検知できる．攻撃の続行を止められる

詳しくは `https://mega-awry.io/` をご覧ください．

まとめ

今日のまとめ

- 共通鍵暗号，暗号化モードを説明した
- ECB モードを使用することによる脆弱性を説明した

さいごに

- 一見安全そうに見えて，実は安全ではないものが，一番危険
  - 安全な暗号方式で暗号化されているため，一見，RSA 秘密鍵は守られているように見える．
  - しかし，不適切に使用されている（ECB モード）ため，安全ではない．
- 推奨されている暗号方式，暗号利用モードを使うべき（認証暗号を使うべき）
- CRYPTREC が提供する報告書，ガイドライン，技術報告書をぜひご覧ください．