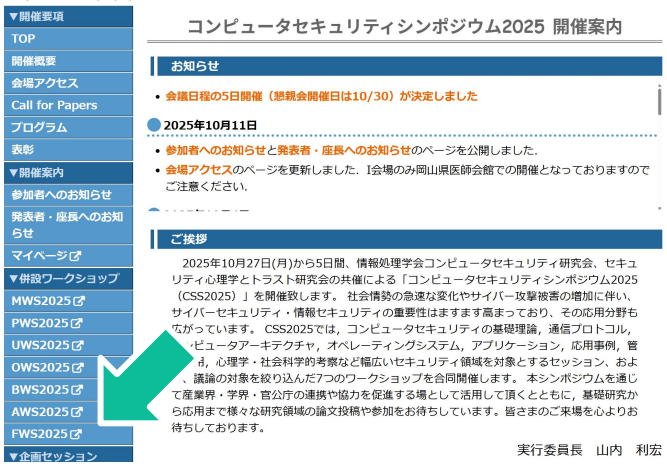
CryptoVerifハンズオン

CSS2025 形式検証とセキュリティワークショップ（FWS）

中林 美郷（NTT社会情報研究所） 花谷 嘉一（東芝） 米山 一樹（茨城大学）

はじめに

- 本ハンズオンではCryptoVerifのオンラインデモを使います
- インターネットにつながるPCをご用意ください
- 資料はこちら（FWSのページ）からダウンロードできます



https://www.iwsec.org/fws/2025/hands-on.html

はじめに

ご不明な点があればお気軽にどうぞ！

- slackのFWSチャンネル
- 挙手による質問

CryptoVerifとは

自動化に特化した，計算論的モデルに基づく暗号システムの形式検証ツール

計算論的モデルに基づくツール

暗号部品を確率的多項式時間のアルゴリズムとして扱い，攻撃者の能力を厳密に評価する

CryptoVerif，EasyCryptなど

示せる安全性　高
検証の自動度　低

記号論的モデルに基づくツール

暗号部品を記号操作として単純化し，到達可能性の解析などを用いて安全性を検証する

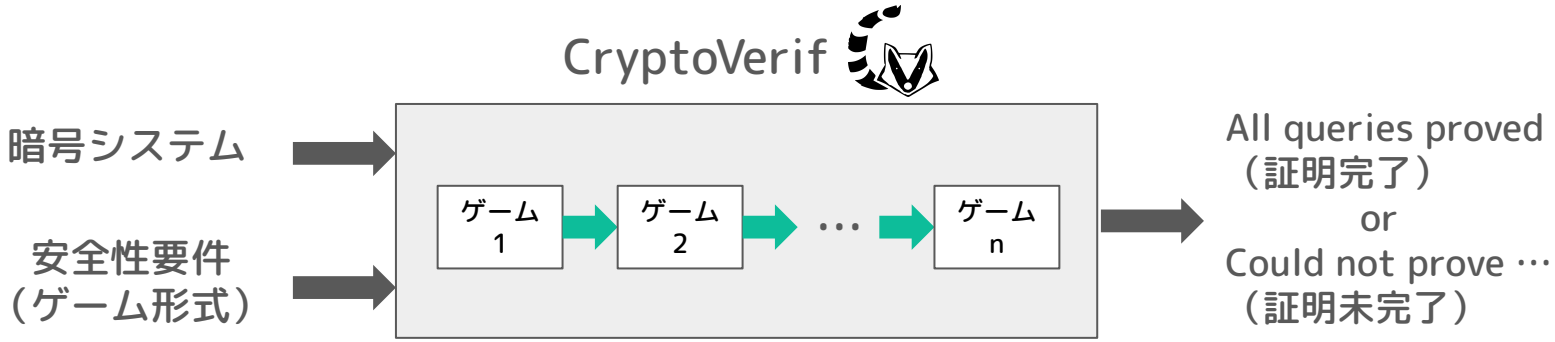
ProVerif，Tamarin Proverなど

示せる安全性　低
検証の自動度　高

計算論的モデルに基づくツールは一般に手動部分が大きいが，CryptoVerifは証明の大部分を自動的に進めてくれる

CryptoVerifとは

自動化に特化した，計算論的モデルに基づく暗号システムの形式検証ツール



ゲーム変換を自動で行い，安全性要件を満たすことを示せるゲームまで到達できれば証明完了と出力

※ 複雑なシステムの証明では，ゲーム変換の方針を手動で入力する必要がある
（インタラクティブモード）

本ハンズオンの概要

CryptoVerifにおける

- 暗号システムの記述方法
- 安全性モデルの記述方法
- 検証結果の読み方

の基礎を理解してもらうことをゴールとします

このハンズオンは，下記のチュートリアルを参考に作成しました：
CryptoVerif Tutorial, Marc Hafner,
https://rub-nds.github.io/AKE-Cryptoverif-Tutorial/Tutorial_mdbook/book/

画像は https://bblanche.gitlabpages.inria.fr/CryptoVerif/ より

もくじ

もくじ

オンラインデモの使い方

1. オンラインデモにアクセス
2. コードをフォームに入力
3. Verifyボタンを押す

http://proverif24.paris.inria.fr/cryptoverif.php

予備知識：ハンズオンで出てくる暗号部品

共通鍵暗号
：暗号化と復号に同じ鍵を用いる暗号

・enc (平文, 鍵) → 暗号文
・dec (暗号文, 鍵) → 平文 or ⊥（復号失敗）

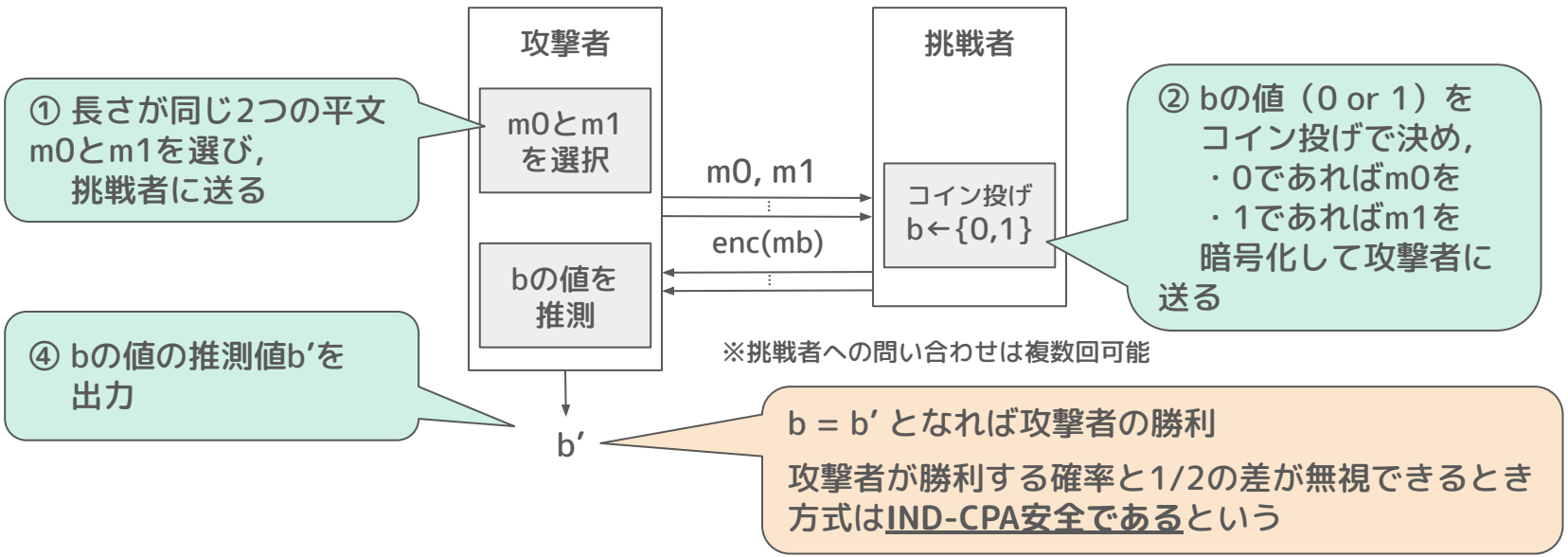
MAC（メッセージ認証コード）
：メッセージの改ざんの検出などに用いられる認証子

・mac (文, MAC鍵) → MAC
・verify (文, MAC鍵, MAC)
→ ⊤（認証成功）or ⊥（認証失敗）

予備知識：暗号の安全性

暗号システムの安全性は挑戦者と攻撃者間のゲームで表現できます

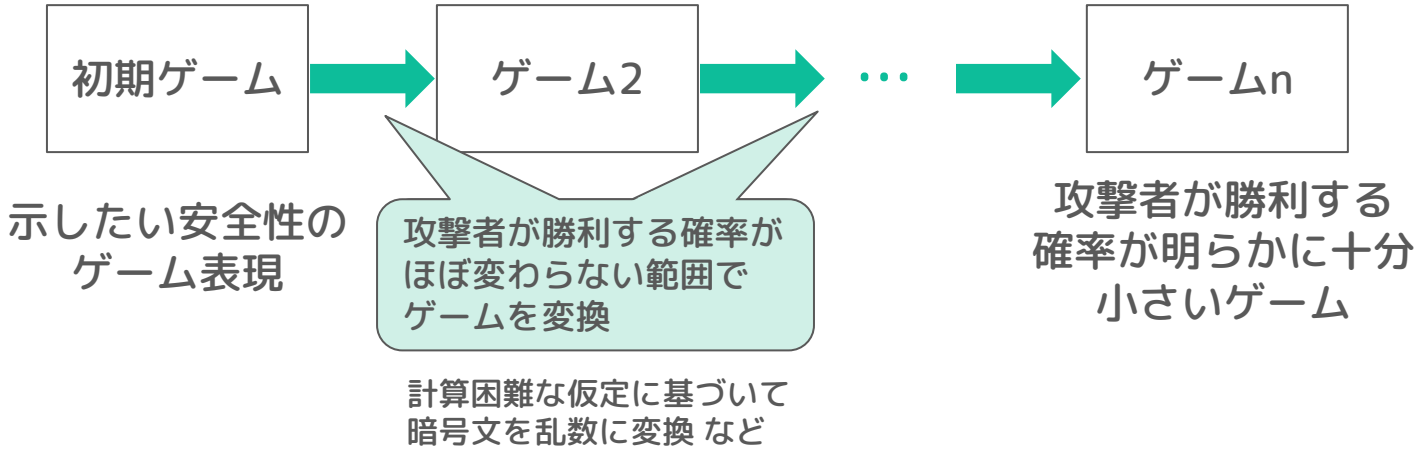
IND-CPAゲーム



※ この講演では，わかりやすさと参考元チュートリアルへの準拠を優先してこのゲームをIND-CPAゲームと呼びます

予備知識：ゲーム列による安全性証明

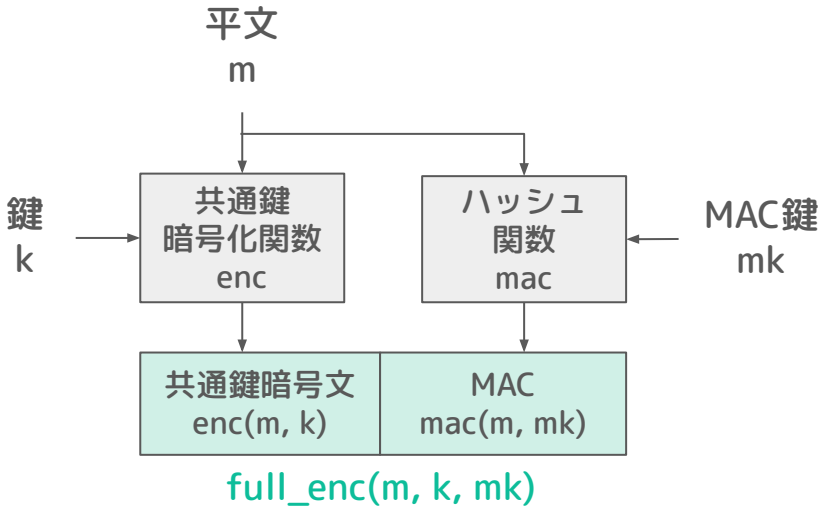
暗号システムが安全性を満たしていることは，ゲームを変換することで証明することができます



もくじ

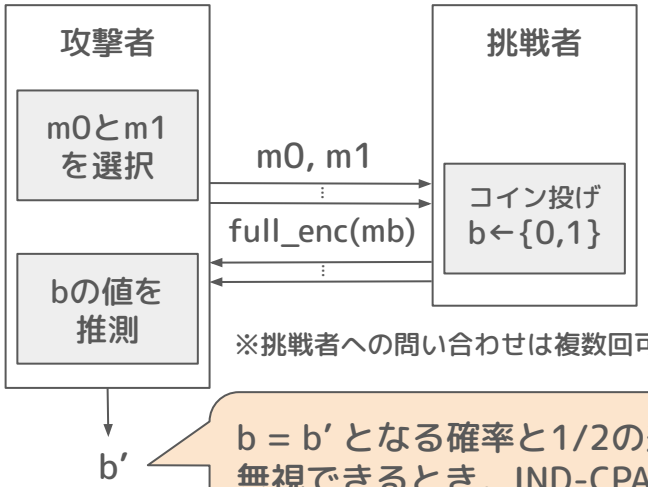
CryptoVerifを用いた暗号システムの安全性証明

例として，Enc-and-MAC方式のIND-CPA安全性を考えていきます



平文の暗号文とMACを連結し，通信相手に送る

※ 省略のため full_enc(m, k, mk) を full_enc(m) と書いたりします



b = b' となる確率と1/2の差が無視できるとき，IND-CPA安全であるという

CryptoVerifを用いた暗号システムの安全性証明

CryptoVerifのコードの全体像を見てみましょう
下記はEnc-and-MACのIND-CPA安全性に関するコードの一部です

```
(* Encrypt-and-MAC is IND-CPA *)
param qEnc.

type mkey [fixed].
type key [fixed].
type macs [fixed].

(* Shared-key encryption (CPA Stream cipher) *)
proba Penc.
expand IND_CPA_sym_enc(key, bitstring, bitstring, enc,
dec, injbot, Z, Penc).

(* Mac *)
proba Pmac.
expand SUF_CMA_det_mac(mkey, bitstring, macs, mac,
verify, Pmac).

fun concat(bitstring, macs): bitstring [data].

letfun full_enc(m: bitstring, k: key, mk: mkey) =
  enc(m, k).
```

```

(* Queries *)
query secret b.

let QencLR(b0: bool, k: key, mk: mkey) =
       foreach i <= qEnc do
       Oenc (m0: bitstring, m1: bitstring) :=
       if Z(m0) = Z(m1) then (* m0 and m1 have the same
length *)
       mb <- if b0 then m0 else m1;
       return(full_enc(mb, k, mk)).

process
       Ostart() :=
       b <-R bool;
       k <-R key;
       mk <-R mkey;
       return;
       run QencLR(b, k, mk)
```

コード1: 1_Enc-and-MAC_IND-CPA_part.cv

CryptoVerifを用いた暗号システムの安全性証明

CryptoVerifのコードの全体像を見てみましょう
下記はEnc-and-MACのIND-CPA安全性に関するコードの一部です

```
(* Encrypt-and-MAC is IND-CPA *)
param qEnc.

type mkey [fixed].
type key [fixed].
type macs [fixed].
```

宣言部

```
(* Shared-key encryption (CPA Stream cipher) *)
proba Penc.
expand IND_CPA_sym_enc(key, bitstring, bitstring, enc,
dec, injbot, Z, Penc).

(* Mac *)
proba Pmac.
expand SUF_CM
verify, Pmac).

fun concat(bitstring, macs): bitstring [data].

letfun full_enc(m: bitstring, k: key, mk: mkey) =
  enc(m, k).
```

暗号システム部

```
(* Queries *)
query secret b.

let QencLR(b0: bool, k: key, mk: mkey) =
       foreach i <= qEnc do
       Oenc (m0: bitstring, m1: bitstring) :=
       if Z(m0) = Z(m1) then (* m0 and m1 have the same
length *)
       mb <- if
       return(f

process
       Ostart() :=
       b <-R bool;
       k <-R key;
       mk <-R mkey;
       return;
       run QencLR(b, k, mk)
```

安全性証明部

コード1: 1_Enc-and-MAC_IND-CPA_part.cv

暗号システムの記述 ー 暗号部品を書いてみよう

CryptoVerifでは，多くの暗号部品が用意されています

ここでは，

（超ざっくり）攻撃者が暗号文から平文の情報を得られない共通鍵暗号方式

- IND-CPAを満たす共通鍵暗号
- SUF-CMAを満たすMAC

を呼び出します

（超ざっくり）攻撃者によって偽造できないMAC

暗号システムの記述 ー 暗号部品を書いてみよう

- IND-CPAを満たす共通鍵暗号

・enc (平文, 鍵) → 暗号文
・dec (暗号文, 鍵) → 平文 or ⊥（復号失敗）

```
8  (* Shared-key encryption (CPA Stream cipher) *)
9  proba Penc.
10 expand IND_CPA_sym_enc(key, bitstring, bitstring, enc, dec, injbot, Z, Penc).
```

鍵の型　平文の型　暗号文の型　暗号化/復号関数　平文の長さ判定関数
復号エラー時の戻り値　攻撃者がIND-CPAを破る確率

- SUF-CMAを満たすMAC

・mac (文, MAC鍵) → MAC
・verify (文, MAC鍵, MAC) → T（認証成功）or ⊥（認証失敗）

```
12 (* Mac *)
13 proba Pmac.
14 expand SUF_CMA_det_mac(mkey, bitstring, macs, mac, verify, Pmac).
```

MAC鍵の型　文の型　MACの型　MAC生成/認証関数
攻撃者がSUF-CMAを破る確率

Penc, Pmacは安全性の評価時に使用

演習1　オンラインデモを使ってみよう

1_Enc-and-MAC_IND-CPA_part.cv をオンラインデモに入力し，結果を見てみましょう

```
(* Encrypt-and-MAC is IND-CPA *)
param qEnc.

type mkey [fixed].
type key [fixed].
type macs [fixed].

(* Shared-key encryption (CPA Stream cipher) *)
proba Penc.
expand IND_CPA_sym_enc(key, bitstring, bitstring, enc,
dec, injbot, Z, Penc).

(* Mac *)
proba Pmac.
expand SUF_CMA_det_mac(mkey, bitstring, macs, mac,
verify, Pmac).

fun concat(bitstring, macs): bitstring [data].

letfun full_enc(m: bitstring, k: key, mk: mkey) =
  enc(m, k).
```

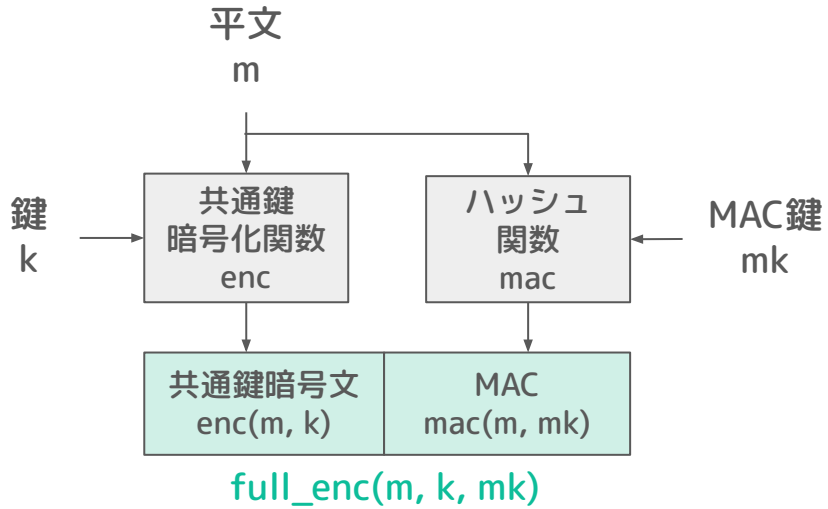
```

(* Queries *)
query secret b.

let QencLR(b0: bool, k: key, mk: mkey) =
       foreach i <= qEnc do
       Oenc (m0: bitstring, m1: bitstring) :=
       if Z(m0) = Z(m1) then (* m0 and m1 have the same
length *)
       mb <- if b0 then m0 else m1;
       return(full_enc(mb, k, mk)).

process
       Ostart() :=
       b <-R bool;
       k <-R key;
       mk <-R mkey;
       return;
       run QencLR(b, k, mk)
```

※コードはハンズオンのページにあります

演習２　Enc-and-MACを構成しよう

2_Enc-and-MAC_IND-CPA_fill.cv の???部分を埋めて，Enc-and-MACを完成させましょう

```
1  (* Encrypt-and-MAC is IND-CPA *)
2  param qEnc.
3
4  type mkey [fixed].
5  type key [fixed].
6  type macs [fixed].
7
8  (* Shared-key encryption (CPA Stream cipher) *)
9  proba Penc.
10 expand IND_CPA_sym_enc(key, bitstring, bitstring, enc,
   dec, injbot, Z, Penc).
11
12 (* Mac *)
13 proba Pmac.
14 expand SUF_CMA_det_mac(mkey, bitstring, macs, mac,
   verify, Pmac).
15
16 fun concat(bitstring, macs): bitstring [data].
17
18 letfun full_enc(m: bitstring, k: key, mk: mkey) =
19   ???.
```



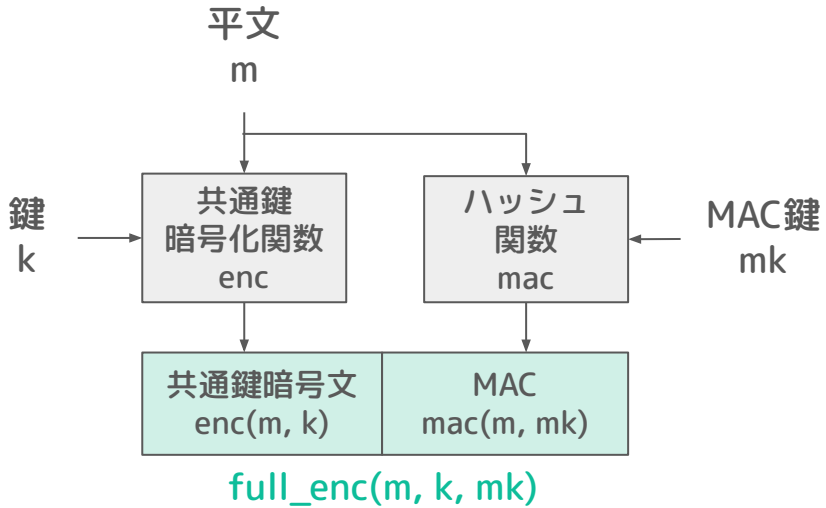
平文m，鍵k，MAC鍵mkを受け取り，暗号文とMACの連結を返す関数full_encを記述します

要素の連結はconcat関数（16行目）を使ってください

暗号システムの記述 ー 暗号システムを書いてみよう

Enc-and-MACは次のように表現できます

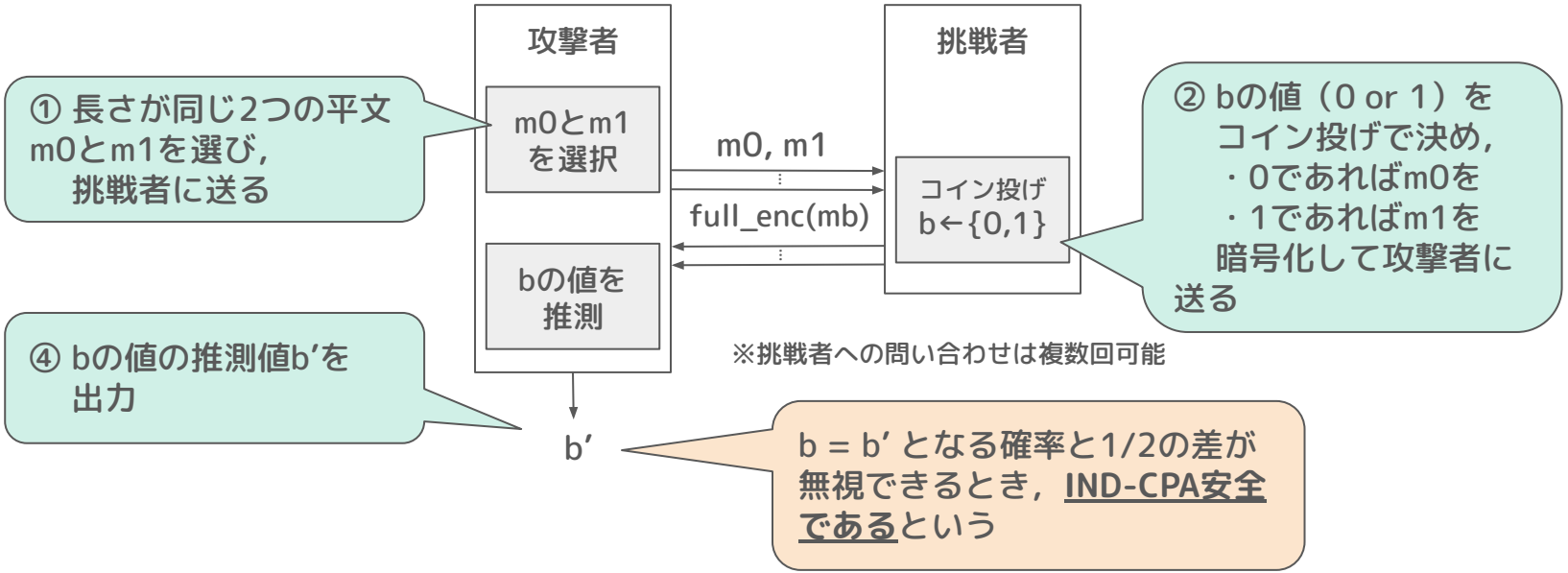
```
16 fun concat(bitstring, macs):
   bitstring [data].
17
18 letfun full_enc(m: bitstring, k:
   key, mk: mkey) =
19   concat(enc(m, k), mac(m, mk)).
```



安全性証明の記述

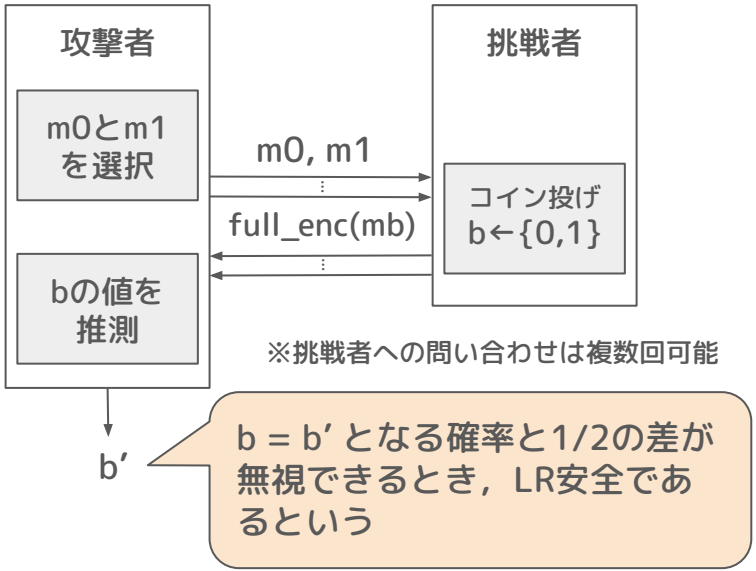
続いて，安全性ゲームを書いていきます

次のIND-CPAゲームを考えます



安全性証明の記述

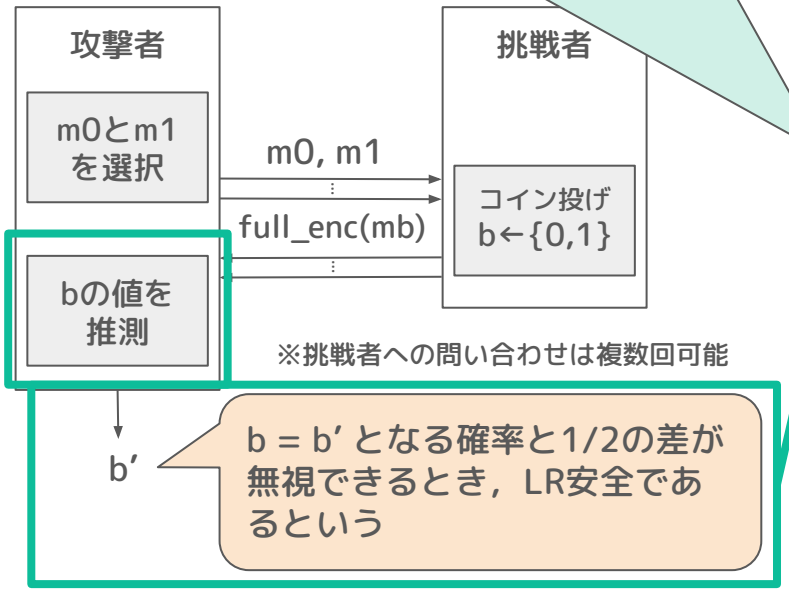
IND-CPAゲームは次のように記述できます




```
21 (* Queries *)
22 query secret b.
23
24 let QencLR(b0: bool, k: key, mk: mkey) =
25       foreach i <= qEnc do
26       Oenc (m0: bitstring, m1: bitstring)
   :=
27       if Z(m1) = Z(m2) then (* m1 and m2
   have the same length *)
28       mb <- if b0 then m0 else m1;
29       return(full_enc(mb, k, mk)).
30
31 process
32       Ostart() :=
33       b <-R bool;
34       k <-R key;
35       mk <-R mkey;
36       return;
37       run QencLR(b, k, mk)
```

安全性証明の記述

ゲームの勝利条件（攻撃者がbの値を推測できるか）を定義

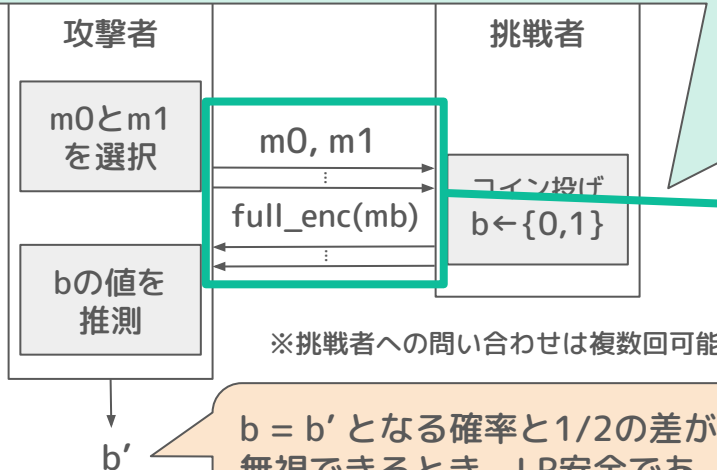


きます

```
(* Queries *)
query secret b.

let QencLR(b0: bool, k: key, mk: mkey) =
      foreach i <= qEnc do
      Oenc (m0: bitstring, m1: bitstring)
:=
      if Z(m1) = Z(m2) then (* m1 and m2
have the same length *)
      mb <- if b0 then m0 else m1;
      return(full_enc(mb, k, mk)).

process
      Ostart() :=
      b <-R bool;
      k <-R key;
      mk <-R mkey;
      return;
      run QencLR(b, k, mk)
```

安全性証明の記述

暗号化オラクルを定義
- 25行目：呼び出し可能回数（qEnc）内であれば，
- 26行目：m0とm1の値を受け取り，
- 27行目：m0とm1の長さが同じであれば，
- 28-29行目：b0の値に応じて full_enc(m0) または full_enc(m1) を返す

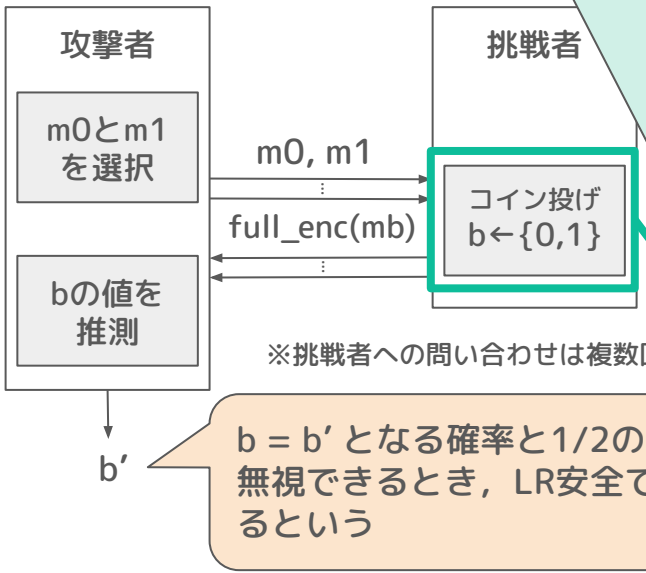


b = b' となる確率と1/2の差が無視できるとき，LR安全であるという

```
25  let QencLR(b0: bool, k: key, mk: mkey) =
26        foreach i <= qEnc do
27        Oenc (m0: bitstring, m1: bitstring)
    :=
28        if Z(m1) = Z(m2) then (* m1 and m2
    have the same length *)
29        mb <- if b0 then m0 else m1;
30        return(full_enc(mb, k, mk)).
31
32  process
33        Ostart() :=
34        b <-R bool;
35        k <-R key;
36        mk <-R mkey;
37        return;
          run QencLR(b, k, mk)
```

安全性証明の記述

（暗号化鍵，MAC鍵を設定し）コイン投げでbの値を決定し，暗号化オラクルを呼び出す



b = b’ となる確率と1/2の差が無視できるとき，LR安全であるという

```
22  query secret b.
23
24  let QencLR(b0: bool, k: key, mk: mkey) =
25        foreach i <= qEnc do
26        Oenc (m0: bitstring, m1: bitstring)
    :=
27        if Z(m1) = Z(m2) then (* m1 and m2
    have the same length *)
28        mb <- if b0 then m0 else m1;
29        return(full_enc(mb, k, mk)).
30
31  process
32        Ostart() :=
33        b <-R bool;
34        k <-R key;
35        mk <-R mkey;
36        return;
37        run QencLR(b, k, mk)
```

演習3　安全性ゲームを入力してみよう

先ほどの2_Enc-and-MAC_IND-CPA_fill.cv を実行し，
検証結果を見てみましょう

※ 参考コードは3_Enc-and-MAC_IND-CPA.cv にあります

実行と実行結果

出力結果を見てみましょう

CryptoVerif output:

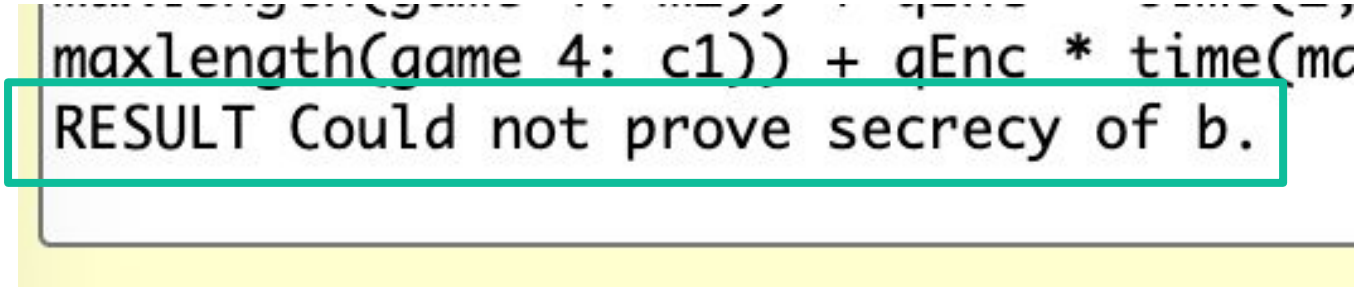
```
File "./tmpfiles/35247923/inpProt.ocv", line 23, characters 7-15:
Warning: For booleans defined under no replication, the option cv_bit may be more appropriate than real_or_random (the default)
Doing expand get, insert and prove unique annotations... No change.
Doing simplify (non-expanded game)... No change.
Doing expand... Done.
Doing remove assignments of findcond... Done.
Doing simplify... Run simplify 1 time(s). Fixpoint reached.
No change.
Doing move all binders... No change.
Doing SA rename new without array accesses and remove assignments of findcond... Done.
Doing merge branches... No change.
Proof of (one-session) secrecy of b failed:
  at 17, bad usage(s) of b.
Trying equivalence suf_cma(mac)... Failed:
Random variables: mk_2 -> k_2
The transformation did not use the useful_change oracles, or oracles deemed useful by default.
Trying equivalence ind_cpa(enc)... Transf. OK Proba. computed Transf. done Succeeded.
Doing simplify (non-expanded game)... No change.
Doing expand... Done.
Doing remove assignments of findcond... Done.
Doing simplify... Run simplify 1 time(s). Fixpoint reached.
No change.
Doing move all binders... No change.
Doing SA rename new without array accesses and remove assignments of findcond... No change.
Doing merge branches... No change.
Proof of (one-session) secrecy of b failed:
  at 17, bad usage(s) of b.
Trying equivalence suf_cma(mac)... Failed:
Random variables: mk_2 -> k_2
The transformation did not use the useful_change oracles, or oracles deemed useful by default.
Trying equivalence ind_cpa(enc)... Failed
```

結構長いようです

何も考えず一番下までスクロールしていただいて・・・

実行と実行結果

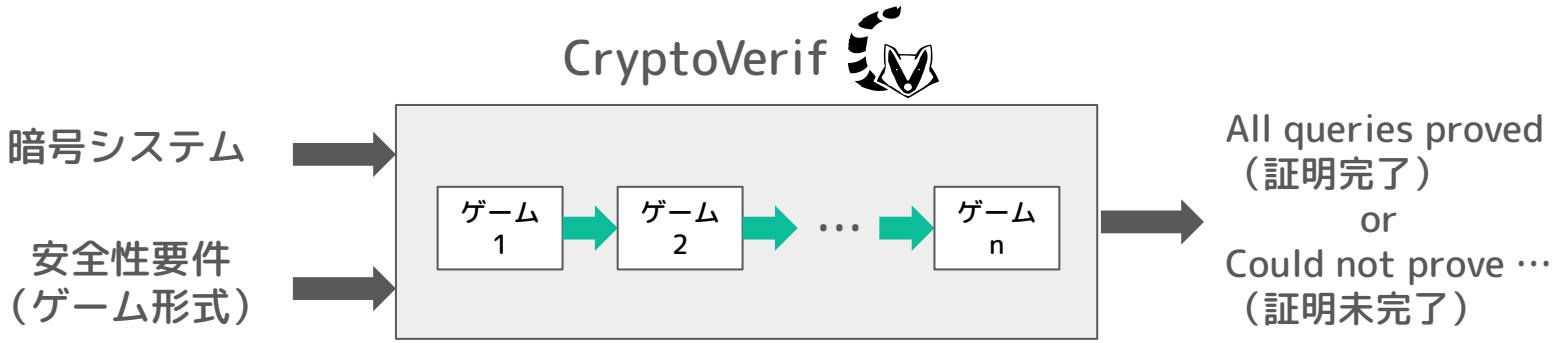
一番最後の行を見てみましょう



どうやら証明できなかったようです

実行と実行結果

CryptoVerifでは，初期ゲームを入力すると自動で変換を行ってくれます



CryptoVerifはProVerifやTamarinのように攻撃を導出することはできませんが，
証明に失敗した際の出力が攻撃の発見に役立つ場合があります

※「証明未完了＝安全性を満たさない」ではありません
　あくまでも，安全であることを示せなかっただけです

実行と実行結果 ー うまくいかない例

出力されたゲーム変換を見ていきましょう

```
==================== Proof starts ====================
Initial state
Game 1 is
      Ostart() :=
      b <-R bool;
      k_3 <-R key;
      mk_2 <-R mkey;
      return();
      foreach i <= qEnc do
      Oenc(m1: bitstring, m2: bitstring) :=
      if Z(m1) = Z(m2) then
      m0: bitstring <- (if b then m1 else m2);
      return((m: bitstring <- m0; c1: bitstring <- (m_1:
bitstring <- m; k_2: key <- k_3; r <-R enc_seed; enc_r(m_1,
k_2, r)); concat(c1, mac(m, mk_2))))

Applying expand
  - Expand if/find/let
yields
Game 2 is
      Ostart() :=
      b <-R bool;
      k_3 <-R key;
      mk_2 <-R mkey;
      return();
```

Game 1

Game 1から2への変換方法

Game 2

実行と実行結果 ー うまくいかない例

ゲーム1からゲーム2へ

```
1  Game 1 is
2       Ostart() :=
3       b <-R bool;
4       k_3 <-R key;
5       mk_2 <-R mkey;
6       return();
7       foreach i <= qEnc do
8       Oenc(m0: bitstring, m1: bitstring) :=
9       if Z(m0) = Z(m1) then
10      mb: bitstring <- (if b then m0 else m1);
11      return((m: bitstring <- mb; concat((m_1:
bitstring <- m; k_2: key <- k_3; r <-R
enc_seed; enc_r(m_1, k_2, r)), mac(m, mk_2))))
```

bの値による分岐を展開

```
1  Game 2 is
2       Ostart() :=
3       b <-R bool;
4       k_3 <-R key;
5       mk_2 <-R mkey;
6       return();
7       foreach i <= qEnc do
8       Oenc(m0: bitstring, m1: bitstring) :=
9       if Z(m0) = Z(m1) then
10      if b then
11        mb: bitstring <- m0;
12        m: bitstring <- mb;
13        m_1: bitstring <- m;
14        k_2: key <- k_3;
15        r <-R enc_seed;
16        return(concat(enc_r(m_1, k_2, r),
mac(m, mk_2)))
17      else
18        mb: bitstring <- m1;
19        m: bitstring <- mb;
20        m_1: bitstring <- m;
21        k_2: key <- k_3;
22        r <-R enc_seed;
23        return(concat(enc_r(m_1, k_2, r),
mac(m, mk_2)))
```

実行と実行結果 ー うまくいかない例

ゲーム2からゲーム3へ

```
1  Game 2 is
2       Ostart() :=
3       b <-R bool;
4       k_3 <-R key;
5       mk_2 <-R mkey;
6       return();
7       foreach i <= qEnc do
8       Oenc(m0: bitstring, m1: bitstring) :=
9       if Z(m0) = Z(m1) then
10      if b then
11        mb: bitstring <- m0;
12        m: bitstring <- mb;
13        m_1: bitstring <- m;
14        k_2: key <- k_3;
15        r <-R enc_seed;
16        return(concat(enc_r(m_1, k_2, r),
    mac(m, mk_2)))
17      else
18        mb: bitstring <- m1;
19        m: bitstring <- mb;
20        m_1: bitstring <- m;
21        k_2: key <- k_3;
22        r <-R enc_seed;
23        return(concat(enc_r(m_1, k_2, r),
    mac(m, mk_2)))
```

不要な一時変数を削除

```
1  Game 3 is
2       Ostart() :=
3       b <-R bool;
4       k_3 <-R key;
5       mk_2 <-R mkey;
6       return();
7       foreach i <= qEnc do
8       Oenc(m0: bitstring, m1: bitstring) :=
9       if Z(m0) = Z(m1) then
10      if b then
111       r <-R enc_seed;
2         return(concat(enc_r(m0, k_3, r), mac(m0,
    mk_2)))
13      else
14        r <-R enc_seed;
15        return(concat(enc_r(m1, k_3, r), mac(m1,
    mk_2)))
```

実行と実行結果 ー うまくいかない例

ゲーム3からゲーム4へ

乱数の名前を変更

```
Game 3 is
     Ostart() :=
     b <-R bool;
     k_3 <-R key;
     mk_2 <-R mkey;
     return();
     foreach i <= qEnc do
     Oenc(m0: bitstring, m1: bitstring) :=
     if Z(m0) = Z(m1) then
     if b then
       r <-R enc_seed;
       return(concat(enc_r(m0, k_3, r), mac(m0,
mk_2)))
     else
       r <-R enc_seed;
       return(concat(enc_r(m1, k_3, r), mac(m1,
mk_2)))
```

```
Game 4 is
     Ostart() :=
     b <-R bool;
     k_3 <-R key;
     mk_2 <-R mkey;
     return();
     foreach i <= qEnc do
     Oenc(m0: bitstring, m1: bitstring) :=
     if Z(m0) = Z(m1) then
     if b then
       r_2 <-R enc_seed;
       return(concat(enc_r(m0, k_3, r_2),
mac(m0, mk_2)))
     else
       r_1 <-R enc_seed;
       return(concat(enc_r(m1, k_3, r_1),
mac(m1, mk_2)))
```

実行と実行結果 ー うまくいかない例

ゲーム4からゲーム5へ

暗号文部分をダミー暗号文に差し替え
（共通鍵暗号がIND-CPAであることから変換OK）

最終的な安全性評価に攻撃者が共通鍵暗号を破る確率Pencが計上される

```
1   Game 4 is
2       Ostart() :=
3       b <-R bool;
4       k_3 <-R key;
5       mk_2 <-R mkey;
6       return();
7       foreach i <= qEnc do
8       Oenc(m0: bitstring, m1: bitstring) :=
9       if Z(m0) = Z(m1) then
10      if b then
111       r_2 <-R enc_seed;
2         return(concat(enc_r(m0, k_3, r_2),
    mac(m0, mk_2)))
13      else
14        r_1 <-R enc_seed;
15        return(concat(enc_r(m1, k_3, r_1),
    mac(m1, mk_2)))
```

```
Applying equivalence ind_cpa(enc) [probability
Penc(time_1, qEnc, max(maxlength(game 4: m1),
maxlength(game 4: m2)))]
  - Equivalence ind_cpa(enc) with variables: r_2
-> r_1, k_3 -> k_2, r_1 -> r_1
yields
```

```
2   Game 5 is
3       Ostart() :=
4       b <-R bool;
5       k_4 <-R key;
6       mk_2 <-R mkey;
7       return();
8       foreach i <= qEnc do
9       Oenc(m0: bitstring, m1: bitstring) :=
10      if Z(m0) = Z(m1) then
111     if b then
2         r_4 <-R enc_seed;
            return(concat((x_1: bitstring <- m0;
    enc_r'(Z(x_1), k_4, r_4)), mac(m0, mk_2)))
13      else
14        r_3 <-R enc_seed;
15        return(concat((x: bitstring <- m1;
    enc_r'(Z(x), k_4, r_3)), mac(m1, mk_2)))
```

enc_r’(Z(x), … ：平文の長さが等しいダミー暗号文

実行と実行結果 ー うまくいかない例

ゲーム5からゲーム6へ

```
1   Game 5 is
2        Ostart() :=
3        b <-R bool;
4        k_4 <-R key;
5        mk_2 <-R mkey;
6        return();
7        foreach i <= qEnc do
8        Oenc(m0: bitstring, m1: bitstring) :=
9        if Z(m0) = Z(m1) then
10       if b then
111        r_4 <-R enc_seed;
2          return(concat((x_1: bitstring <- m0;
   enc_r'(Z(x_1), k_4, r_4)), mac(m0, mk_2)))
13       else
14         r_3 <-R enc_seed;
15         return(concat((x: bitstring <- m1;
   enc_r'(Z(x), k_4, r_3)), mac(m1, mk_2)))
```

代入の順序を変更

```
1   Game 6 is
2        Ostart() :=
3        b <-R bool;
4        k_4 <-R key;
5        mk_2 <-R mkey;
6        return();
7        foreach i <= qEnc do
8        Oenc(m0: bitstring, m1: bitstring) :=
9        if Z(m0) = Z(m1) then
10       if b then
111        r_4 <-R enc_seed;
2          x_1: bitstring <- m0;
13         return(concat(enc_r'(Z(x_1), k_4, r_4),
   mac(m0, mk_2)))
14       else
15         r_3 <-R enc_seed;
16         x: bitstring <- m1;
17         return(concat(enc_r'(Z(x), k_4, r_3),
   mac(m1, mk_2)))
```

実行と実行結果 ー うまくいかない例

ゲーム6からゲーム7へ

```
1   Game 6 is
2       Ostart() :=
3       b <-R bool;
4       k_4 <-R key;
5       mk_2 <-R mkey;
6       return();
7       foreach i <= qEnc do
8       Oenc(m0: bitstring, m1: bitstring) :=
9       if Z(m0) = Z(m1) then
10      if b then
111       r_4 <-R enc_seed;
2         x_1: bitstring <- m0;
13        return(concat(enc_r'(Z(x_1), k_4, r_4),
    mac(m0, mk_2)))
14      else
15        r_3 <-R enc_seed;
16        x: bitstring <- m1;
17        return(concat(enc_r'(Z(x), k_4, r_3),
    mac(m1, mk_2)))
```

不要な一時変数を削除

```
1   Game 7 is
2       Ostart() :=
3       b <-R bool;
4       k_4 <-R key;
5       mk_2 <-R mkey;
6       return();
7       foreach i <= qEnc do
8       Oenc(m0: bitstring, m1: bitstring) :=
9       if Z(m0) = Z(m1) then
10      if b then
111       r_4 <-R enc_seed;
2         return(concat(enc_r'(Z(m0), k_4, r_4),
    mac(m0, mk_2)))
13      else
14        r_3 <-R enc_seed;
15        return(concat(enc_r'(Z(m1), k_4, r_3),
    mac(m1, mk_2)))
```

実行と実行結果 ー うまくいかない例

ゲーム7が最終ゲームです

戻り値に平文（m0またはm1）の情報が含まれているため，攻撃者は暗号文オラクルを用いてbの値を容易に推測することができます

そのため，証明が失敗したようです

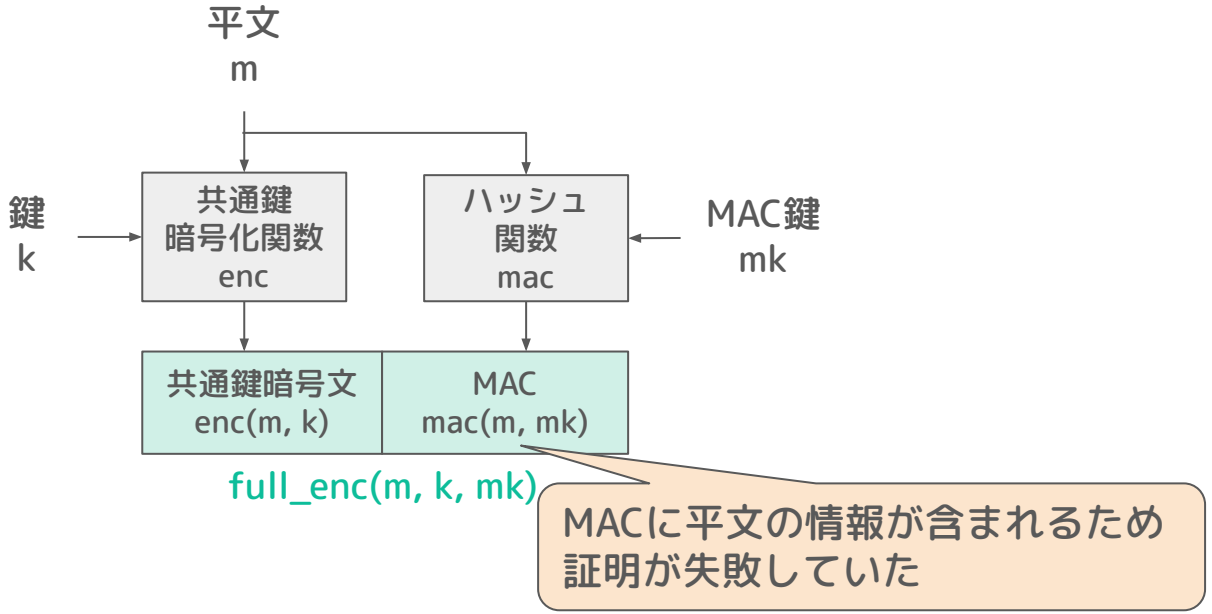
```
RESULT time_1 = qEnc * time(= bitstring, length(Z,
maxlength(game 4: m0)), length(Z, maxlength(game 4: m1))) +
qEnc * time(Z, maxlength(game 4: m1)) + qEnc * time(Z,
maxlength(game 4: m0)) + qEnc * time(concat, length(enc_r,
maxlength(game 4: m0))) + qEnc * time(mac, maxlength(game 4:
m0)) + qEnc * time(concat, length(enc_r, maxlength(game 4:
m1))) + qEnc * time(mac, maxlength(game 4: m1)) + time
RESULT Could not prove secrecy of b.
```

```
1   Game 7 is
2       Ostart() :=
3       b <-R bool;
4       k_4 <-R key;
5       mk_2 <-R mkey;
6       return();
7       foreach i <= qEnc do
8       Oenc(m0: bitstring, m1: bitstring) :=
9       if Z(m0) = Z(m1) then
10      if b then
111       r_4 <-R enc_seed;
2         return(concat(enc_r'(Z(m0), k_4, r_4),
    mac(m0, mk_2)))
13      else
14        r_3 <-R enc_seed;
15        return(concat(enc_r'(Z(m1), k_4, r_3),
    mac(m1, mk_2)))
```

m0またはm1のの情報を含む

実行と実行結果 ー うまくいく例

CryptoVerifの出力を基に，システムがIND-CPA安全になるように修正していきましょう



演習4　システムをIND-CPA安全にしよう

先ほどのコードのシステム部分を修正して，IND-CPA安全を満たすようにしましょう

```
(* Encrypt-and-MAC is IND-CPA *)
param qEnc.

type mkey [fixed].
type key [fixed].
type macs [fixed].

(* Shared-key encryption (CPA Stream cipher) *)
proba Penc.
expand IND_CPA_sym_enc(key, bitstring, bitstring, enc,
dec, injbot, Z, Penc).

(* Mac *)
proba Pmac.
expand SUF_CMA_det_mac(mkey, bitstring, macs, mac,
verify, Pmac).

fun concat(bitstring, macs): bitstring [data].

letfun full_enc(m: bitstring, k: key, mk: mkey) =
  ★.
```

システム部分

出力の最後が

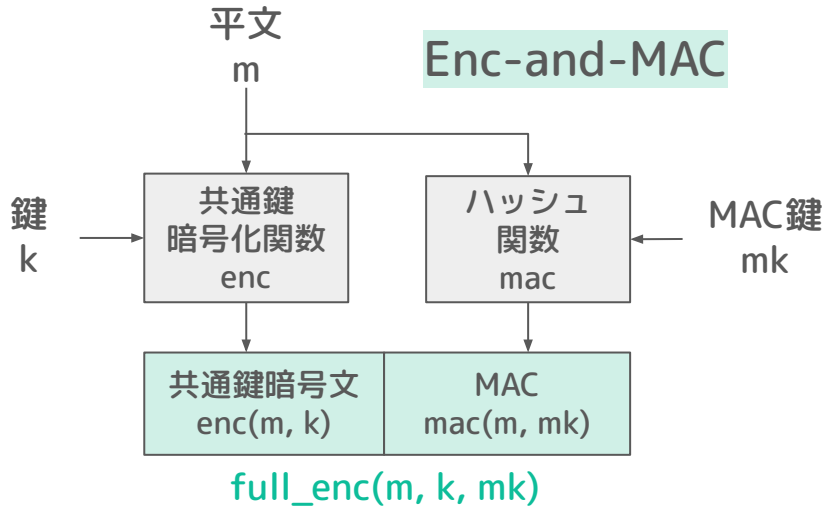
```
All queries proved.
```

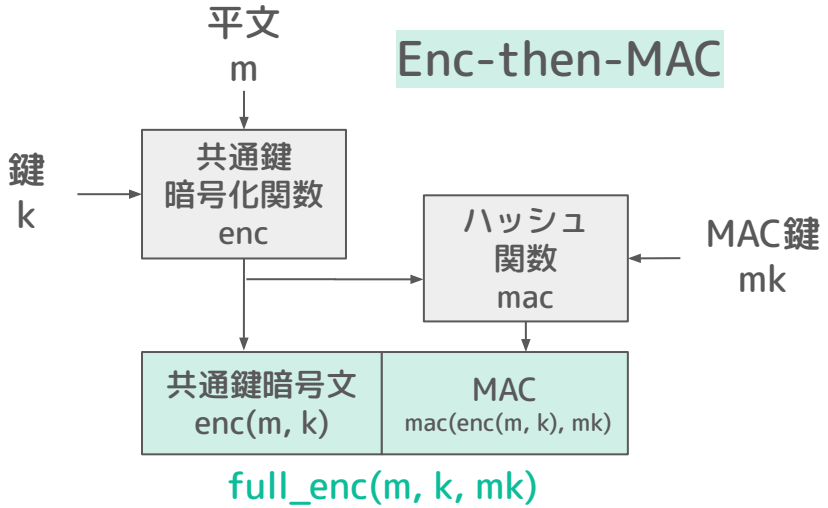
となれば成功です！

参考コード（穴埋め前）は
[4_Enc-and-MAC_IND-CPA_secure.cv]
にあります

実行と実行結果 ー うまくいく例

出力に平文のMACがそのまま含まれていることが攻撃につながっていたため，平文を暗号文に置き換えると良さそうです






```
19 letfun full_enc(m: bitstring, k:
key, mk: mkey) =
     concat(enc(m,k),mac(m,mk)).
```

```
19 letfun full_enc(m: bitstring, k:
key, mk: mkey) =
     concat(enc(m,k),mac(enc(m,k),mk)).
```

※ 修正の方法はもちろんこの限りではありません

実行と実行結果 ー うまくいく例

出力の最後の行を見ると，証明が完了している（＝Enc-then-MACがIND-CPA安全であることを示せた）ことがわかります

```
Proved secrecy of b in game 8
Adv[Game 1: secrecy of b] <= 2 * Penc(time_1, 2 * qEnc, max(maxlength(game 4: m1),
maxlength(game 4: m0))) + Adv[Game 8: secrecy of b]
Adv[Game 8: secrecy of b] <= 0
RESULT Proved secrecy of b up to probability 2 * Penc(time_1, 2 * qEnc, max(maxlength(game
4: m1), maxlength(game 4: m0)))
RESULT time_1 = qEnc * time(= bitstring, length(Z, maxlength(game 4: m0)), length(Z,
maxlength(game 4: m1))) + qEnc * time(Z, maxlength(game 4: m1)) + qEnc * time(Z,
maxlength(game 4: m0)) + qEnc * time(concat, length(enc_r, maxlength(game 4: m0))) + qEnc
* time(mac, length(enc_r, maxlength(game 4: m0))) + qEnc * time(concat, length(enc_r,
maxlength(game 4: m1))) + qEnc * time(mac, length(enc_r, maxlength(game 4: m1))) + time
All queries proved.
```

ゲーム変換を見ていきましょう

演習5　各ゲーム変換を理解しよう

5_Enc-then-MAC_IND-CPA.cv を入力し，Enc-then-MACの安全性ゲーム列を手に入れましょう

各安全性ゲームの変換はそれぞれ何を意味しているでしょうか？

if文を展開	不要な一時変数の削除
if文をマージ	順序を変更
乱数の名前を変更	暗号文を ダミーに差し替え

実行と実行結果 ー うまくいく例

ゲーム1からゲーム2へ

bの値による分岐を展開

```
1  Game 1 is
2       Ostart() :=
3       b <-R bool;
4       k_3 <-R key;
5       mk_2 <-R mkey;
6       return();
7       foreach i <= qEnc do
8       Oenc(m0: bitstring, m1: bitstring) :=
9       if Z(m0) = Z(m1) then
10      mb: bitstring <- (if b then m0 else m1);
11      return((m: bitstring <- mb; concat((r_1
    <-R enc_seed; enc_r(m, k_3, r_1)), mac((r_2 <-R
    enc_seed; enc_r(m, k_3, r_2)), mk_2))))
```

if文を展開

```
1  Game 2 is
2       Ostart() :=
3       b <-R bool;
4       k_3 <-R key;
5       mk_2 <-R mkey;
6       return();
7       foreach i <= qEnc do
8       Oenc(m0: bitstring, m1: bitstring) :=
9       if Z(m0) = Z(m1) then
10      if b then
11        mb: bitstring <- m0;
12        m: bitstring <- mb;
13        r_1 <-R enc_seed;
14        r_2 <-R enc_seed;
15        return(concat(enc_r(m, k_3, r_1),
    mac(enc_r(m, k_3, r_2), mk_2)))
16      else
17        mb: bitstring <- m1;
18        m: bitstring <- mb;
19        r_1 <-R enc_seed;
20        r_2 <-R enc_seed;
21        return(concat(enc_r(m, k_3, r_1),
    mac(enc_r(m, k_3, r_2), mk_2)))
```

実行と実行結果 ー うまくいく例

不要な一時変数の削除

ゲーム2からゲーム3へ

不要な一時変数を削除

```
1  Game 2 is
2       Ostart() :=
3       b <-R bool;
4       k_3 <-R key;
5       mk_2 <-R mkey;
6       return();
7       foreach i <= qEnc do
8       Oenc(m0: bitstring, m1: bitstring) :=
9       if Z(m0) = Z(m1) then
10      if b then
11        mb: bitstring <- m0;
12        m: bitstring <- mb;
13        r_1 <-R enc_seed;
14        r_2 <-R enc_seed;
15        return(concat(enc_r(m, k_3, r_1),
    mac(enc_r(m, k_3, r_2), mk_2)))
16      else
17        mb: bitstring <- m1;
18        m: bitstring <- mb;
19        r_1 <-R enc_seed;
20        r_2 <-R enc_seed;
21        return(concat(enc_r(m, k_3, r_1),
    mac(enc_r(m, k_3, r_2), mk_2)))
```

```
1  Game 3 is
2       Ostart() :=
3       b <-R bool;
4       k_3 <-R key;
5       mk_2 <-R mkey;
6       return();
7       foreach i <= qEnc do
8       Oenc(m0: bitstring, m1: bitstring) :=
9       if Z(m0) = Z(m1) then
10      if b then
11        r_1 <-R enc_seed;
12        r_2 <-R enc_seed;
13        return(concat(enc_r(m0, k_3, r_1),
    mac(enc_r(m0, k_3, r_2), mk_2)))
14      else
15        r_1 <-R enc_seed;
16        r_2 <-R enc_seed;
17        return(concat(enc_r(m1, k_3, r_1),
    mac(enc_r(m1, k_3, r_2), mk_2)))
```

実行と実行結果 ー うまくいく例

乱数の名前を変更

ゲーム3からゲーム4へ

乱数の名前を変更

```
1  Game 3 is
2       Ostart() :=
3       b <-R bool;
4       k_3 <-R key;
5       mk_2 <-R mkey;
6       return();
7       foreach i <= qEnc do
8       Oenc(m0: bitstring, m1: bitstring) :=
9       if Z(m0) = Z(m1) then
10      if b then
11        r_1 <-R enc_seed;
12        r_2 <-R enc_seed;
13        return(concat(enc_r(m0, k_3, r_1),
    mac(enc_r(m0, k_3, r_2), mk_2)))
14      else
15        r_1 <-R enc_seed;
16        r_2 <-R enc_seed;
17        return(concat(enc_r(m1, k_3, r_1),
    mac(enc_r(m1, k_3, r_2), mk_2)))
```

```
1  Game 4 is
2       Ostart() :=
3       b <-R bool;
4       k_3 <-R key;
5       mk_2 <-R mkey;
6       return();
7       foreach i <= qEnc do
8       Oenc(m0: bitstring, m1: bitstring) :=
9       if Z(m0) = Z(m1) then
10      if b then
11        r_5 <-R enc_seed;
12        r_6 <-R enc_seed;
13        return(concat(enc_r(m0, k_3, r_5),
    mac(enc_r(m0, k_3, r_6), mk_2)))
14      else
15        r_3 <-R enc_seed;
16        r_4 <-R enc_seed;
17        return(concat(enc_r(m1, k_3, r_3),
    mac(enc_r(m1, k_3, r_4), mk_2)))
```

実行と実行結果 ー うまくいく例

暗号文を
ダミーに差し替え

ゲーム4からゲーム5へ

2か所の暗号文部分をダミー暗号文に差し替え
（共通鍵暗号がIND-CPAであることから変換OK）

```
1  Game 4 is
2       Ostart() :=
3       b <-R bool;
4       k_3 <-R key;
5       mk_2 <-R mkey;
6       return();
7       foreach i <= qEnc do
8       Oenc(m0: bitstring, m1: bitstring) :=
9       if Z(m0) = Z(m1) then
10      if b then
11        r_5 <-R enc_seed;
12        r_6 <-R enc_seed;
13        return(concat(enc_r(m0, k_3, r_5),
    mac(enc_r(m0, k_3, r_6), mk_2)))
14      else
15        r_3 <-R enc_seed;
16        r_4 <-R enc_seed;
17        return(concat(enc_r(m1, k_3, r_3),
    mac(enc_r(m1, k_3, r_4), mk_2)))
```

```
      mk_2 <-R mkey;
6     return();
7     foreach i <= qEnc do
8     Oenc(m0: bitstring, m1: bitstring) :=
9     if Z(m0) = Z(m1) then
10    if b then
11      r_9 <-R enc_seed;
12      r_8 <-R enc_seed;
13      return(concat((x_2: bitstring <- m0;
    enc_r'(Z(x_2), k_4, r_9)), mac((x_1:
    bitstring <- m0; enc_r'(Z(x_1), k_4, r_8)),
    mk_2)))
14    else
15      r_10 <-R enc_seed;
16      r_7 <-R enc_seed;
17      return(concat((x_3: bitstring <- m1;
    enc_r'(Z(x_3), k_4, r_10)), mac((x:
    bitstring <- m1; enc_r'(Z(x), k_4, r_7)),
    mk_2)))
```

実行と実行結果 ー うまくいく例

ゲーム5からゲーム6へ

```
1  Game 5 is
2       Ostart() :=
3       b <-R bool;
4       k_4 <-R key;
5       mk_2 <-R mkey;
6       return();
7       foreach i <= qEnc do
8       Oenc(m0: bitstring, m1: bitstring) :=
9       if Z(m0) = Z(m1) then
10      if b then
11        r_9 <-R enc_seed;
12        r_8 <-R enc_seed;
13        return(concat((x_2: bitstring <- m0;
          enc_r'(Z(x_2), k_4, r_9)), mac((x_1:
          bitstring <- m0; enc_r'(Z(x_1), k_4, r_8)),
          mk_2)))
14      else
15        r_10 <-R enc_seed;
16        r_7 <-R enc_seed;
17        return(concat((x_3: bitstring <- m1;
          enc_r'(Z(x_3), k_4, r_10)), mac((x:
          bitstring <- m1; enc_r'(Z(x), k_4, r_7)),
          mk_2)))
```

代入の順序を変更

順序を変更

```
1  Game 6 is
2       Ostart() :=
3       b <-R bool;
4       k_4 <-R key;
5       mk_2 <-R mkey;
6       return();
7       foreach i <= qEnc do
8       Oenc(m0: bitstring, m1: bitstring) :=
9       if Z(m0) = Z(m1) then
10      if b then
11        r_9 <-R enc_seed;
12        r_8 <-R enc_seed;
13        x_2: bitstring <- m0;
14        x_1: bitstring <- m0;
15        return(concat(enc_r'(Z(x_2), k_4,
          r_9), mac(enc_r'(Z(x_1), k_4, r_8), mk_2)))
16      else
17        r_10 <-R enc_seed;
18        r_7 <-R enc_seed;
19        x_3: bitstring <- m1;
20        x: bitstring <- m1;
21        return(concat(enc_r'(Z(x_3), k_4,
          r_10), mac(enc_r'(Z(x), k_4, r_7), mk_2)))
```

実行と実行結果 ー うまくいく例

不要な一時変数の削除

ゲーム6からゲーム7へ

```
Game 6 is
      Ostart() :=
      b <-R bool;
      k_4 <-R key;
      mk_2 <-R mkey;
      return();
      foreach i <= qEnc do
      Oenc(m0: bitstring, m1: bitstring) :=
      if Z(m0) = Z(m1) then
      if b then
        r_9 <-R enc_seed;
        r_8 <-R enc_seed;
        x_2: bitstring <- m0;
        x_1: bitstring <- m0;
        return(concat(enc_r'(Z(x_2), k_4,
r_9), mac(enc_r'(Z(x_1), k_4, r_8), mk_2)))
      else
        r_10 <-R enc_seed;
        r_7 <-R enc_seed;
        x_3: bitstring <- m1;
        x: bitstring <- m1;
        return(concat(enc_r'(Z(x_3), k_4,
r_10), mac(enc_r'(Z(x), k_4, r_7), mk_2)))
```

不要な一時変数を削除

```
Game 7 is
      Ostart() :=
      b <-R bool;
      k_4 <-R key;
      mk_2 <-R mkey;
      return();
      foreach i <= qEnc do
      Oenc(m0: bitstring, m1: bitstring) :=
      if Z(m0) = Z(m1) then
 {16}if b then
        r_9 <-R enc_seed;
        r_8 <-R enc_seed;
        return(concat(enc_r'(Z(m0), k_4,
r_9), mac(enc_r'(Z(m0), k_4, r_8), mk_2)))
      else
        r_10 <-R enc_seed;
        r_7 <-R enc_seed;
        return(concat(enc_r'(Z(m1), k_4,
r_10), mac(enc_r'(Z(m1), k_4, r_7), mk_2)))
```

実行と実行結果 ー うまくいく例

if文をマージ

ゲーム7からゲーム8へ

if文をマージ
（分岐先の内容が同じなのでマージOK）

```
1  Game 7 is
2       Ostart() :=
3       b <-R bool;
4       k_4 <-R key;
5       mk_2 <-R mkey;
6       return();
7       foreach i <= qEnc do
8       Oenc(m0: bitstring, m1: bitstring) :=
9       if Z(m0) = Z(m1) then
10   {16}if b then
11        r_9 <-R enc_seed;
12        r_8 <-R enc_seed;
13        return(concat(enc_r'(Z(m0), k_4,
    r_9), mac(enc_r'(Z(m0), k_4, r_8), mk_2)))
14      else
15        r_10 <-R enc_seed;
16        r_7 <-R enc_seed;
17        return(concat(enc_r'(Z(m1), k_4,
    r_10), mac(enc_r'(Z(m1), k_4, r_7), mk_2)))
```

```
1  Game 8 is
2       Ostart() :=
3       b <-R bool;
4       k_4 <-R key;
5       mk_2 <-R mkey;
6       return();
7       foreach i <= qEnc do
8       Oenc(m0: bitstring, m1: bitstring) :=
9       if Z(m0) = Z(m1) then
10      r_10 <-R enc_seed;
11      r_7 <-R enc_seed;
12      return(concat(enc_r'(Z(m1), k_4,
    r_10), mac(enc_r'(Z(m1), k_4, r_7), mk_2)))
```

実行と実行結果 ー うまくいく例

ゲーム8が最終ゲームです

戻り値にはダミー暗号文しか含まれていないため，攻撃者は平文の情報を手に入れることができません

証明成功です！

```
Proved secrecy of b in game 8
Adv[Game 1: secrecy of b] <= 2 * Penc(time_1, 2 * qEnc,
max(maxlength(game 4: m1), maxlength(game 4: m0))) + Adv[Game
8: secrecy of b]
Adv[Game 8: secrecy of b] <= 0
RESULT Proved secrecy of b up to probability 2 * Penc(time_1,
2 * qEnc, max(maxlength(game 4: m1), maxlength(game 4: m0)))
RESULT time_1 = qEnc * time(= bitstring, length(Z,
maxlength(game 4: m0)), length(Z, maxlength(game 4: m1))) +
qEnc * time(Z, maxlength(game 4: m1)) + qEnc * time(Z,
maxlength(game 4: m0)) + qEnc * time(concat, length(enc_r,
maxlength(game 4: m0))) + qEnc * time(mac, length(enc_r,
maxlength(game 4: m0))) + qEnc * time(concat, length(enc_r,
maxlength(game 4: m1))) + qEnc * time(mac, length(enc_r,
maxlength(game 4: m1))) + time
All queries proved.
```

if文をマージ

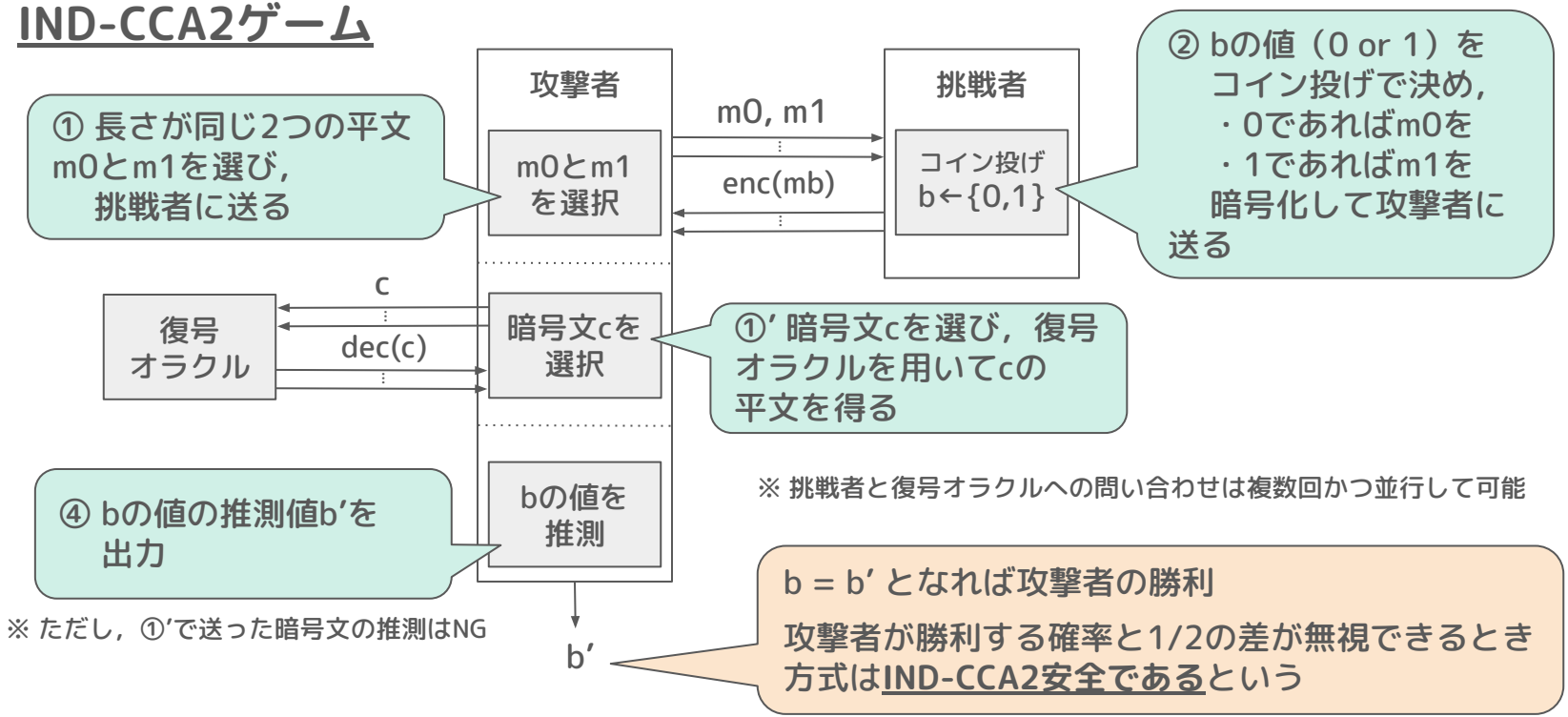
```
1  Game 8 is
2       Ostart() :=
3       b <-R bool;
4       k_4 <-R key;
5       mk_2 <-R mkey;
6       return();
7       foreach i <= qEnc do
8       Oenc(m0: bitstring, m1: bitstring) :=
9       if Z(m0) = Z(m1) then
10      r_10 <-R enc_seed;
11      r_7 <-R enc_seed;
12      return(concat(enc_r'(Z(m1), k_4,
13 r_10), mac(enc_r'(Z(m1), k_4, r_7), mk_2)))
```

m0とm1の情報を含まない

もくじ

応用編：Enc-then-MACがIND-CCA2を満たすこと

IND-CCA2ゲーム

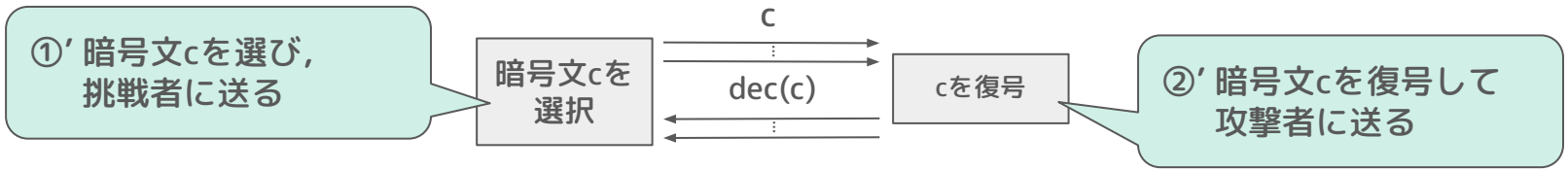


※（再）この講演では，わかりやすさと参考元チュートリアルへの準拠を優先してこのゲームをIND-CCA2ゲームと呼びます

応用編：Enc-then-MACがIND-CCA2を満たすこと

応用編として，Enc-then-MACがIND-CCA2を満たすことを証明できるかみてみましょう

IND-CCA2の証明では，IND-CPAのときに比べて**復号オラクル**を追加する必要があります



応用演習　IND-CCA2ゲームを記述しよう

暗号化オラクルを参考に，6_Enc-then-MAC_IND-CCA2_fill.cv
の???部分を埋めて復号オラクルを定義してみましょう

コードを実行し，どのような実行結果とゲーム変換が出力される
のか確認してみましょう

※ 応用編の詳しい解説はテキストをご覧ください

もくじ

おわりに

このハンズオンでは，暗号システムの形式検証ツールCryptoVerifの

- 暗号システムの記述方法
- 安全性モデルの記述方法
- 検証結果の読み方

を紹介しました

興味を持っていただけた方は，CryptoVerif公式マニュアルや関連研究のコードを見てもらえればと思います！

皆さんの研究に活用してもらえれば幸いです