解　説

暗号系の安全性検証 — 入門から計算機による証明まで

川本 裕輔

本論文では，暗号系の安全性を検証する方法について，暗号理論の初歩からコンピュータ上で安全性証明を書く最先端の研究までを解説する．暗号系の安全性は攻撃の確率や計算量を用いて定式化されるため，その数学的証明は煩雑であり，しばしば間違いも生じる．そこで，安全性証明を形式手法で記述し，証明の正しさをコンピュータ上で機械的に確かめる研究が盛んになっており，様々な検証ツールが開発されている．その中でも，検証ツール EasyCrypt は，従来のツールよりも簡単に，厳密な安全性証明が得られる．本論文では，まず，確率的多項式時間チューリング機械の間のゲームとして暗号系の安全性を定式化し，ゲームの書き換えによって安全性を証明する手法を説明する．次に，確率関係ホーア論理を用いた安全性証明の形式化について述べ，検証ツール EasyCrypt を用いて安全性証明を書く方法を紹介する．なお，本論文は暗号理論や定理証明支援系についての知識を仮定していない．

This paper gives an overview of verification methods for the security of cryptosystems from basic concepts of cryptography to advanced topics on computer-aided security proofs. The security of cryptosystems is formalized using probabilities and computational complexity of attacks while the mathematical proofs for such security tend to be complicated and error prone. To obtain rigorous security proofs there have been many studies on formalizing and machine-checking proofs using formal methods. Among various verification tools EasyCrypt is the most successful tool that can rigorously construct security proofs more easily than previous tools. In this paper we introduce a method for defining the security of cryptosystems as games among probabilistic polynomial-time (PPT) Turing machines and proving it by game transformation techniques. Then we explain how to formalize such security proofs in the framework of probabilistic relational Hoare logic (pRHL) and to write and machine-check proofs using EasyCrypt. Note that readers do not need to be familiar with cryptography or interactive theorem provers.

1　はじめに

現代の暗号は数学に基づいており，様々な暗号系に対して数学的な安全性の証明が考えられてきた．しかし，安全性証明は，確率，計算量，通信（並行性）を扱って煩雑であり，間違いが見つかることも多い．

仮に暗号系が安全でなければ，問題が生じるのは情報セキュリティだけではない．自動車や医療機器などあらゆるモノがインターネットにつながる時代が近づいており，もし暗号系の安全性に不備があれば，物理的な危険も生じてしまう．

そこで，コンピュータ上で，暗号系の安全性証明を記述し，検証する研究が盛んに行われている．これらの研究では，プログラミング言語理論の分野で蓄積された知見が暗号系の安全性検証に活用されている．

本稿では，暗号系の安全性を証明する方法について，暗号理論の初歩からコンピュータ上で安全性証明を書く最先端の研究までを解説する．暗号理論や定理証明支援系を知らなくても，概略を掴めるよう説明する．まず，2 節では数学的に安全性を証明する方法を紹介する．3 節で形式手法（確率関係ホーア論理）を用いて証明を厳密に記述する方法を紹介し，4 節で検証ツール EasyCrypt を用いてコンピュータ上で証明を行う方法を紹介する．5 節では関連研究を紹介する．

本稿を読んでも，直ちにコンピュータ上で証明を書

Verification of Cryptosystems – from Introduction to Computer-Aided Security Proofs.

Yusuke Kawamoto, 国立研究開発法人産業技術総合研究所, AIST, Japan.

コンピュータソフトウェア, Vol.33, No.4 (2016), pp.1–17.
[解説論文] 2016 年 3 月 31 日受付.

けるようにはならないが，コンピュータ上での証明の書き方を知ることは，人手で証明を記述する際にも良い影響を及ぼすものと筆者は考える．また，前提知識を持たずに，EasyCrypt についての論文やマニュアルを理解することは非常に難しく，本稿がこれらを読むための足がかりとなることを期待している．

2 数学的に安全性を証明

暗号の歴史は，破られては新しい方式が考案されるということの繰り返しだったが，数学的に安全性が保証された暗号も提案されるようになった．このような暗号を**証明可能安全性**（provable security）を持つ暗号という．本節では**公開鍵暗号**（public-key encryption）の証明可能安全性を紹介する †1．

2.1 公開鍵暗号

公開鍵暗号には，**公開鍵**と**秘密鍵**の 2 種類の鍵があり，それぞれ暗号化と復号に用いる．A さんが B さんに平文（メッセージ）m を暗号化して伝えたい場合，図 1 のとおり，① 受信者 B さんの公開鍵 pk_B を使って平文 m を暗号化し，② 暗号文（暗号化されたメッセージ）を送る．③ 受信者 B さんがこの暗号文を復号する際には，B さん自身の秘密鍵 sk_B を使う．

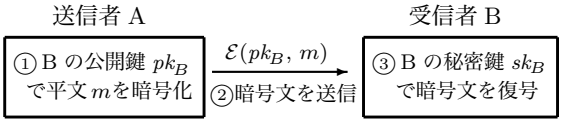


図 1: 公開鍵暗号による通信

ここで，秘密鍵 sk_B は B さんだけが持っているものだから，他の人はこの暗号文を復号できない．一方，公開鍵 pk_B は公開されているため，誰でも「B さんしか復号できない暗号文」を生成できる．

2.1.1 公開鍵暗号の定義と性質

公開鍵暗号は以下の 3 つのアルゴリズムの組 $(\mathcal{KG}, \mathcal{E}, \mathcal{D})$ として定義される．

- **鍵生成アルゴリズム** $\mathcal{KG}$ は，鍵の長さ（セキュリティ・パラメータ）η を受け取り，公開鍵と秘密鍵のペア (pk, sk) をランダムに生成する．
- **暗号化アルゴリズム** $\mathcal{E}$ は，公開鍵 pk と平文（メッセージ）m を受け取り，その暗号文 c を返す．
- **復号アルゴリズム** $\mathcal{D}$ は，秘密鍵 sk と暗号文 c を受け取り，復号の結果を返す．

なお，暗号化アルゴリズムが内部で乱数を生成して使っている場合は，同じ公開鍵と平文に対して，毎回異なるビット列の暗号文を生成できる．

公開鍵暗号は，**正当性**（correctness）と**秘匿性**（security）を満たす必要がある．

- 正当性とは，秘密鍵 sk を使って暗号文 c を復号すると元の平文 m を取り出せるという性質で，式で表すと，$\mathcal{D}(sk, \mathcal{E}(pk, m)) = m$ である．
- 秘匿性とは，暗号を解読できる確率（秘密鍵 sk を持たない人が元の平文 m を取り出せる確率）が非常に小さいという性質である．鍵の長さ η が大きいほど，秘匿性の度合いが増す．

2.1.2 公開鍵暗号の例： ElGamal 暗号

単純な公開鍵暗号の例として **ElGamal 暗号** [20] を取り上げよう．アルゴリズムを図 2 に示す．

η ビットの素数 q に対して，$\mathbb{G}$ を位数 q の巡回群とし，g を $\mathbb{G}$ の生成元とし，平文を $m \in \mathbb{G}$ とする．

鍵生成アルゴリズム $\mathcal{KG}$

$\mathcal{KG}(\eta) \stackrel{\text{def}}{=} sk \xleftarrow{\$} \mathbb{Z}_q^*;\ pk \leftarrow g^{sk};\ \mathsf{return}\ (pk, sk);$

暗号化アルゴリズム $\mathcal{E}$

$\mathcal{E}(pk, m) \stackrel{\text{def}}{=} y \xleftarrow{\$} \mathbb{Z}_q^*;\ c \leftarrow (g^y, pk^y \cdot m);\ \mathsf{return}\ c;$

復号アルゴリズム $\mathcal{D}$

$\mathcal{D}(sk, c) \stackrel{\text{def}}{=} (a, b) \leftarrow c;\ m \leftarrow b/a^{sk};\ \mathsf{return}\ m;$

図 2: ElGamal 暗号

確率的代入 $sk \xleftarrow{\$} \mathbb{Z}_q^*$ は，集合 $\mathbb{Z}_q^* \stackrel{\text{def}}{=} \{1, \ldots, q-1\}$ からランダムに取ってきた要素を sk とおくことを表す．より厳密に言うと，他の事象から**独立に**，$\mathbb{Z}_q^*$ の要素を**一様に**（等確率 $\frac{1}{q-1}$ で）選ぶという意味である．

一方，$pk \leftarrow g^{sk}$ は通常の決定的代入を表す．g^{sk} や $pk^y \cdot m$ は巡回群 $\mathbb{G}$ の元である．

ElGamal 暗号の正当性は，簡単な式変形

$$\mathcal{D}(sk, \mathcal{E}(pk, m)) = (pk^y \cdot m)/(g^y)^{sk} = m$$

で確認できる．ElGamal 暗号の秘匿性は**離散対数問**

†1 暗号理論の基礎を説明するので，証明可能安全性に詳しい読者は 2.1 節〜2.3 節を読み飛ばしてほしい．

題（の変種）の難しさに基づいている．例えば，鍵の長さ η が十分に大きいとき，巡回群 $\mathbb{G}$ の位数 q が大き過ぎて，公開鍵 g^{sk} から秘密鍵 sk を計算することが困難である．y が乱数であるため，乗算の結果 $pk^y \cdot m$ から平文 m を取り出すことも困難である．

2.2 計算量的安全性

ElGamal 暗号の秘匿性を数学的に証明するために，まず，秘匿性とは何であるかを数学的に定義しよう．証明可能安全性の理論では，暗号の秘匿性を，**攻撃者と挑戦者の間のゲーム**を用いて定義する．大雑把に言うと，このゲームでは，「挑戦者が用意した暗号文」を攻撃者が解読しようとする．攻撃者が平文についての情報を全く手に入れられないとき，攻撃者の負けであり，このとき，秘匿性が成り立つと定義する．

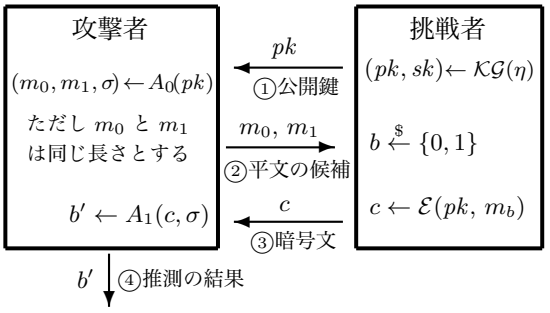


図 3: IND-CPA ゲーム

厳密には，図 3 の **IND-CPA ゲーム**で定義される．① 攻撃者は最初に公開鍵 pk を受け取る．② 攻撃者が，同じ長さの平文 m_0 と m_1 を用意し，挑戦者に渡す．③ 挑戦者は，ビット $b \in \{0,1\}$ をランダムに選び，m_b を暗号化し，暗号文 c を攻撃者に渡す．（このとき，m_0 を暗号化する確率も，m_1 を暗号化する確率も 50%.）④ 攻撃者は，暗号文 c が m_0 と m_1 のどちらの暗号文なのか（すなわちビット b の値）を推測し，推測の結果を b' とおく．

攻撃者が**どちらの平文なのかを正しく推測**できる確率（つまり $b = b'$ が成り立つ確率）がほとんど 1/2 のとき，暗号の秘匿性が成り立つものと定義する．

ここで，秘匿性が成り立つためには，攻撃者の計算能力に制限が必要である．というのも，可能な平文や鍵の数が有限であるため，仮に攻撃者が無限の計算能力を持っていたら，すべての可能性を調べ上げて平文を特定できてしまう．そのため，秘匿性の定義では，攻撃者を**確率的多項式時間（PPT）チューリング機械** †2 としてモデル化することが多い．

定義 1 任意の PPT 攻撃者に対し，「IND-CPA ゲームで $b = b'$ となる確率」と 1/2 の差が**無視できる** †3 ほど小さいとき，**IND-CPA 安全性**が成り立つという．

2.3 安全性証明の仮定

暗号の証明可能安全性は，数学の問題の難しさに基づいている．例えば，ElGamal 暗号の IND-CPA 安全性の証明では，**DDH 問題**（離散対数問題の変種）を解くことの困難さを利用して，多項式時間帰着の議論を行う．DDH 問題の定義を図 4 に示す．

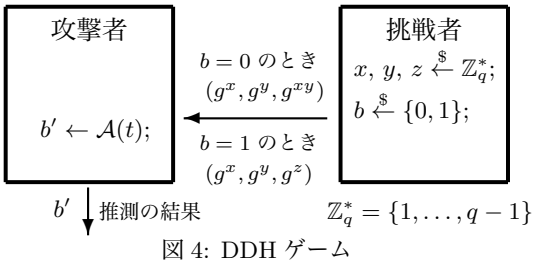


図 4: DDH ゲーム

DDH ゲームでは，挑戦者が集合 $\mathbb{Z}_q^*$ からランダムに x, y, z を選び，ビット $b \in \{0,1\}$ をランダムに選ぶ．$b = 0$ のとき (g^x, g^y, g^{xy}) を攻撃者に送り，$b = 1$ のときは (g^x, g^y, g^z) を送る．攻撃者は，g^{xy} と g^z のどちらを受け取ったのか（b の値）を推測する．

DDH 問題とは，DDH ゲームにおけるビット b を正しく求める問題である．DDH 問題の難しさは，数学的には証明されていないが，広く信じられており，さまざまな暗号系の安全性を証明する上で仮定されている．**DDH 仮定**を厳密に書くと，

いかなる PPT 攻撃者 $\mathcal{A}$ に対しても，「攻撃

†2 PPT チューリング機械（PPT 攻撃者）は，乱数テープを読み，入出力テープを通じて挑戦者と通信を行い，η の多項式で表される回数の動作の後に停止する．

†3 鍵の長さ η が大きくなるにつれて，「$b = b'$ となる確率」は 1/2 に近づく．「$b = b'$ となる確率」と 1/2 の差を $p(\eta)$ とおく．どんな多項式 $poly(\eta)$ に対しても，η を十分に大きく取ったときに $p(\eta) < \frac{1}{poly(\eta)}$ となるとき，関数 $p(\eta)$ が**無視できる**（negligible）という．

者 $\mathcal{A}$ が DDH ゲームのビット b を正しく推測できる確率」と 1/2 の差が無視できる，

となる．

2.4 ゲーム列による安全性証明

暗号系の安全性を証明する方法として**ゲーム列による安全性証明**[12][21][28] を紹介する．具体例として「ElGamal 暗号が，DDH 仮定のもとで，IND-CPA 安全性を満たすこと」の証明を説明する．

2.4.1 証明したいこと

証明したい事柄は，

「いかなる PPT 攻撃者も DDH 問題を効率的に解くことができない」ならば，「いかなる PPT 攻撃者も ElGamal 暗号の IND-CPA 安全性を破ることができない」

である．

公開鍵暗号 $\mathcal{S} = (\mathcal{KG}, \mathcal{E}, \mathcal{D})$ の IND-CPA ゲームを，図 5 の $\mathsf{CPA}(\mathcal{S}, \mathcal{A})$ のように記述する．ゲーム末尾のブール型変数 res の値は，$b = b'$ のとき true，$b \neq b'$ のとき false である．ゲーム $\mathsf{CPA}(\mathcal{S}, \mathcal{A})$ を実行した後で変数 res の値が true である確率を

$$\Pr[\mathsf{CPA}(\mathcal{S}, \mathcal{A}) \colon res]$$

と書くことにする．すると，ElGamal 暗号の IND-CPA 安全性は，任意の確率的多項式時間アルゴリズムの対 $\mathcal{A} = (A_0, A_1)$ に対して，確率

$$|\Pr[\mathsf{CPA}(\mathsf{ElGamal}, \mathcal{A}) \colon res] - 1/2|$$

が無視できることである．

2.4.2 証明の全体概要

IND-CPA 安全性を証明するために，図 5 に示すゲームの列を考える．このゲーム列では，左上のゲーム CPA を，$\mathsf{CPA}_{\mathsf{ElGamal}}$, DDH_0, DDH_1, G_1, G_2 の順に，少しずつ書き換えている．

一番最後のゲーム G_2 では，b と b' が独立に選ばれているため，$b = b'$ となる確率はちょうど 1/2 である．すなわち，任意の PPT 攻撃者の対 $\mathcal{A}$ に対し，

$$\Pr[\mathsf{G}_2(\mathcal{A}) \colon res] = 1/2$$

である．よって，

$$|\Pr[\mathsf{CPA}(\mathsf{ElGamal}, \mathcal{A}) \colon res] - 1/2|$$

が無視できること（IND-CPA 安全性）を示すには，

$$|\Pr[\mathsf{CPA}(\mathsf{ElGamal}, \mathcal{A}) \colon res] - \Pr[\mathsf{G}_2(\mathcal{A}) \colon res]|$$

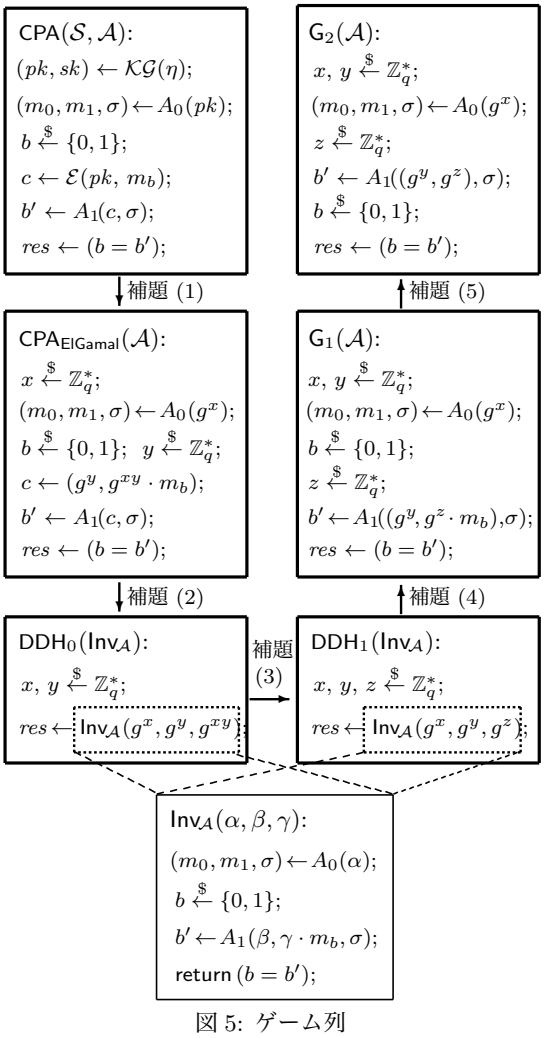


図 5: ゲーム列

が無視できることを証明すれば十分である．

これを示すために，図 5 の各ゲームの間の確率が等しいか，もしくは差が無視できることを証明する．具体的には以下の補題 (1)〜(5) を示す．

(1) $\Pr[\mathsf{CPA}(\mathsf{ElGamal}, \mathcal{A}) \colon res] = \Pr[\mathsf{CPA}_{\mathsf{ElGamal}}(\mathcal{A}) \colon res]$

(2) $\Pr[\mathsf{CPA}_{\mathsf{ElGamal}}(\mathcal{A}) \colon res] = \Pr[\mathsf{DDH}_0(\mathsf{Inv}_{\mathcal{A}}) \colon res]$

(3) $|\Pr[\mathsf{DDH}_0(\mathsf{Inv}_{\mathcal{A}}) \colon res] - \Pr[\mathsf{DDH}_1(\mathsf{Inv}_{\mathcal{A}}) \colon res]|$ が無視できる

(4) $\Pr[\mathsf{DDH}_1(\mathsf{Inv}_{\mathcal{A}}) \colon res] = \Pr[\mathsf{G}_1(\mathcal{A}) \colon res]$

(5) $\Pr[\mathsf{G}_1(\mathcal{A}) \colon res] = \Pr[\mathsf{G}_2(\mathcal{A}) \colon res]$

これらの補題により，任意の PPT 攻撃者 $\mathcal{A}$ に対し，

$$|\Pr[\mathsf{CPA}(\mathsf{ElGamal}, \mathcal{A}) \colon res] - \Pr[\mathsf{G}_2(\mathcal{A}) \colon res]|$$
$$= |\Pr[\mathsf{DDH}_0(\mathsf{Inv}_{\mathcal{A}}) \colon res] - \Pr[\mathsf{DDH}_1(\mathsf{Inv}_{\mathcal{A}}) \colon res]|$$

が無視できることが導かれる．

2.4.3 証明の各ステップ

各補題の証明を詳しく見てみよう．

まず補題 (1) を示そう．ゲーム $\mathsf{CPA}(\mathcal{S},\mathcal{A})$ の手続き $\mathcal{KG}$ と $\mathcal{E}$ に，ElGamal 暗号のアルゴリズムを代入する．すると，$pk = g^x$，$sk = x$，$c = (g^y, g^{xy}\cdot m_b)$ より，ゲーム $\mathsf{CPA}_{\mathsf{ElGamal}}(\mathcal{A})$ が得られる．これらの代入操作は，確率を変えないため，補題 (1) が成り立つ．

次に補題 (2) を示そう．x と y を生成する箇所以外を別のゲーム $\mathsf{Inv}_{\mathcal{A}}(g^x, g^y, g^{xy})$ としてまとめると，ゲーム $\mathsf{DDH}_0(\mathcal{A})$ が得られる．これらの操作は並べ替えだけで，確率を変えないため，補題 (2) が成り立つ．

次に補題 (3) を証明しよう．ゲーム $\mathsf{DDH}_0(\mathsf{Inv}_{\mathcal{A}})$ と $\mathsf{DDH}_1(\mathsf{Inv}_{\mathcal{A}})$ の違いは，サブルーチン $\mathsf{Inv}_{\mathcal{A}}$ に対して g^{xy} を渡すか，あるいは g^z を渡すかという点である．仮に補題 (3) が成り立たない，すなわち確率の差 $|\Pr[\mathsf{DDH}_0(\mathsf{Inv}_{\mathcal{A}}) : b = b'] - \Pr[\mathsf{DDH}_1(\mathsf{Inv}_{\mathcal{A}}) : b = b']|$ が無視できないほど大きいと仮定しよう．すると，PPT 攻撃者 $\mathsf{Inv}_{\mathcal{A}}$ が g^{xy} と g^z のどちらを受け取ったのかを識別できてしまい，DDH 仮定に反する．よって，背理法により補題 (3) が成り立つ．

次に補題 (4) を示そう．ゲーム $\mathsf{DDH}_1(\mathsf{Inv}_{\mathcal{A}})$ と $\mathsf{G}_1(\mathcal{A})$ の違いは，乱数 z を生成する箇所である．z は，$\mathsf{DDH}_1(\mathsf{Inv}_{\mathcal{A}})$ では冒頭に生成されているが，$\mathsf{G}_1(\mathcal{A})$ では b を選んだ後で生成されている．しかし，z は他の変数から独立に生成されているため，$b = b'$ が成り立つ確率は二つのゲームで等しく，補題 (4) が成り立つ．

最後に補題 (5) を示そう．ゲーム $\mathsf{G}_1(\mathcal{A})$ と $\mathsf{G}_2(\mathcal{A})$ の違いは，A_1 の引数として $g^z\cdot m_b$ を渡すか，g^z を渡すかである．z は一様かつ独立な乱数であるため，g^z も一様かつ独立である．そのため，$g^z\cdot m_b$ も同様であり，二つのゲームで $b = b'$ となる確率は等しく，補題 (5) が成り立つ．

3 形式的に安全性を証明

前節で紹介した ElGamal 暗号は最も単純な暗号系のひとつであり，IND-CPA 安全性ではメッセージを盗聴するだけの比較的弱い攻撃者（**受動的攻撃者**）を扱っているため，安全性の証明は比較的単純である．

しかし，実用上は，メッセージを送信したり改竄したりできる強い攻撃者（**能動的攻撃者**）を考慮する必要がある．このような強い攻撃者のもとで安全な暗号系は，ElGamal 暗号よりもはるかに複雑である．そのため，安全性の証明はじつに煩雑であり，後から間違いが見つかることも多い †4．

間違った証明を書いてしまう原因は色々考えられるが，証明の導出の途中を省略したときに多い．省略した箇所で，なんとなく正しいと信じている事柄を使って，推論を飛躍させたり間違えたりするのである．

安全性証明に間違いが見つかっては新しい暗号系と安全性証明が考案されるということの繰り返しを防ぐためには，証明の導出を省略せず，証明を**形式的に記述する**ことが有効である．形式的に記述すれば，仮定を正しく使っているか，式変形に飛躍がないかなどをもっと厳密に確認できる．本節では，確率関係ホーア論理を用いて証明を記述する方法を紹介したい．

3.1 ゲームを記述するための言語 pWhile

まず，暗号系の安全性定義に用いるゲームを厳密に記述するための言語として，確率的手続き型プログラミング言語 pWhile を考えよう．図 6 に pWhile の構文を示す．

文 ::=	skip	（何もしない）
	変数 $\leftarrow$ 数式	（決定的代入）
|	変数 $\overset{\$}{\leftarrow}$ 確率分布表現	（確率的代入）
|	if 条件式 then 文 else 文	（条件文）
|	while 条件式 do 文	（ループ）
|	変数 $\leftarrow$ 手続 (数式, . . . , 数式)	（手続き呼出し）
|	文; 文	（逐次実行）

図 6: プログラミング言語 pWhile の構文

この言語では，通常の決定的代入，条件文，ループ，手続き呼び出し，逐次実行だけでなく，（有限集合上の）**確率的代入**を記述できる．確率的代入 $x \overset{\$}{\leftarrow} d$ は，確率分布 d から値を取って，変数 x に代入する操作

†4 誤りのない証明を書くことは難しい．暗号系の安全性証明に限らず，著名な数学者でさえ，飛躍や誤りのある証明を書くことがある．例えば，Wikipedia の英語版には不完全な証明の一覧がある．https://en.wikipedia.org/wiki/List_of_incomplete_proofs

を表す．例えば，$x \xleftarrow{\$} [1 \mathinner{..} q]$ は，集合 $\{1, \ldots, q\}$ から値をランダムに選ぶことを表す．数式では，ブール値，整数値，固定長ビット列，リストなどを扱える．

言語 pWhile を用いると，暗号系の安全性証明のゲームを，2.4 節の図 5 のように記述できる．

3.2 言語 pWhile の意味論

言語 pWhile で記述されたゲーム G の振る舞いは，G をプログラムとして実行したときの**メモリ状態**の変化として説明できる．ここで，メモリ状態とは，変数から値への写像のことである．例えば，メモリ状態 m で変数 x の値が 5 のとき，$m(x) = 5$ と書く．

何らかのメモリ状態においてゲーム G を実行し，実行後のメモリ状態がどうなるか考えてみよう．ゲーム G が乱数を使ったり，確率的な手続き呼び出しを含んでいると，実行後のメモリ状態は一つに定まらない．例えば，一つの命令文だけからなるゲーム $x \xleftarrow{\$} [0 \mathinner{..} 3]$ を実行すると，あるメモリ状態 m に確率 0.25 で遷移して $m(x) = 0$ となり，別の状態 m' に確率 0.25 で遷移して $m'(x) = 1$ となる．そういうわけで，ゲームを実行した後の状況は,「メモリ状態の確率分布」として捉えることができる †5．

ゲーム G の意味を,「実行前のメモリ状態」から「実行後のメモリ状態の確率分布」への関数 $[\![\mathsf{G}]\!]$ として定義しよう †6．このとき，メモリ状態 m でゲーム G を実行した後のメモリ状態の確率分布は $[\![\mathsf{G}]\!](m)$ と書ける．メモリ状態 m でゲーム G を実行した後に事象 E が成り立つ確率を $\Pr[\,\mathsf{G}, m\colon E\,]$ と書くことにする．

3.3 確率関係ホーア論理

それでは，言語 pWhile で記述されたゲームを使って，暗号系の安全性について推論する方法を考えよう．推論を記述する枠組みとして，本稿では**確率関係ホーア論理**（pRHL）を紹介する．

†5 ゲームがループを含んでいて停止しない場合，いかなるメモリ状態にも到達しない．したがって，厳密には，確率の総和が 1 より小さいかもしれない**部分分布**（sub-distribution）を考える必要がある．本稿では単純化のため，ループを含まないゲームのみを扱う．

†6 意味論の詳細は [9] を参照．厳密には確率分布をモナドで定義している [3].

3.3.1 ホーア式

節 2.4 で述べたように，暗号系の安全性証明ではゲームの間の関係についての推論を行う．二つのゲーム G_1 と G_2 の間の関係は，例えば図 7 のように，ゲームを実行する前と後で変化する．

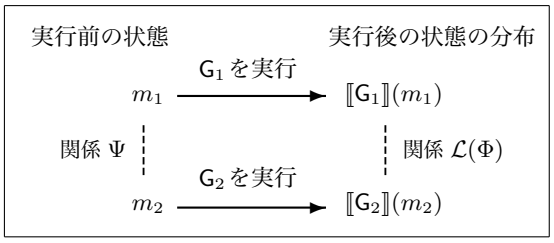


図 7: 二つのゲームの間の関係

ゲーム G_1 と G_2 の間の関係は，実行によって変化するメモリ状態の間の関係として表現できる．図 7 では，初期メモリ状態 m_1 と m_2 の間に関係 Ψ が成り立っており，これを $m_1 \mathrel{\Psi} m_2$ で表す．この状態 m_1 でゲーム G_1 を実行し，状態 m_2 でゲーム G_2 を実行すると，実行後のメモリ状態の確率分布がそれぞれ $[\![\mathsf{G}_1]\!](m_1)$ と $[\![\mathsf{G}_2]\!](m_2)$ となり，図 7 ではこの二つの確率分布の間に，関係 $\mathcal{L}(\Phi)$ が成り立っている．

この $\mathcal{L}(\Phi)$ は，メモリ状態の間の関係 Φ を「確率分布に持ち上げて」得られた「状態の確率分布の間の関係」である．持ち上げ $\mathcal{L}$ の特殊な場合については 3.3.2 節で簡単に説明するが，$\mathcal{L}$ の厳密な定義 [22] に興味のある読者は注釈 †7 を参照してほしい．

さて，図 7 のような二つのゲームの間の関係は，確率関係ホーア論理を用いて記述できる．$m_1 \mathrel{\Psi} m_2$ を満たす任意のメモリ状態 m_1 と m_2 に対して，$[\![\mathsf{G}_1]\!](m_1)\,\mathcal{L}(\Phi)\,[\![\mathsf{G}_2]\!](m_2)$ が成り立つことを，確率関

†7 二つの状態集合 M_1 と M_2 の間の関係 Φ の**確率分布への持ち上げ** $\mathcal{L}(\Phi)$ とは，M_1 上の分布と M_2 上の分布の間の関係で，次の条件を満たすものとする [10]：$d_1 \mathrel{\mathcal{L}(\Phi)} d_2$ を満たす任意の分布 d_1 と d_2 に対して，$M_1 \times M_2$ 上の確率分布 d が存在し，

1. 任意の $(m_1, m_2) \in M_1 \times M_2$ に対し，$d(m_1, m_2) > 0$ ならば $m_1 \mathrel{\Phi} m_2$,
2. 任意の $m_1 \in M_1$ に対し，$d_1(m_1) = \sum_{m_2 \in M_2} d(m_1, m_2)$,
3. 任意の $m_2 \in M_2$ に対し，$d_2(m_2) = \sum_{m_1 \in M_1} d(m_1, m_2)$.

係ホーア論理では，**ホーア式**（judgment）

$$\models \mathsf{G}_1 \sim \mathsf{G}_2 \colon \Psi \Rightarrow \Phi$$

で表す †8．Ψ を**事前条件**といい，Φ を**事後条件**という．

この記法は複雑でやや紛らわしいかもしれない．矢印 $\Rightarrow$ は古典論理の「ならば」ではない．記号 $\sim$ を使うことで二つのゲームの振る舞いが「等しい」ように見えるが，通常想像する意味での「等しさ」ではない．

3.3.2 事後条件を確率分布に持ち上げた関係

事後条件 Φ を確率分布に持ち上げた関係 $\mathcal{L}(\Phi)$ について少し説明しよう．分かりやすくするために，本稿では「事後条件 Φ が同値関係の場合」だけを説明する．実際のところ，多くの暗号系の安全性証明を記述する際には同値関係を考えれば十分である．

例えば，次の二つのゲーム G_1 と G_2 を考えよう．

- $\mathsf{G}_1 \stackrel{\text{def}}{=} x \xleftarrow{\$} \{0,1\}$
- $\mathsf{G}_2 \stackrel{\text{def}}{=} y \xleftarrow{\$} \{2,3\}$

G_1 と G_2 を実行する前のメモリ状態を，それぞれ m_1, m_2 で表す．G_1 と G_2 の実行の前後の状態に対し，

- 事前条件 $\Psi \stackrel{\text{def}}{=} \mathsf{true}$
- 事後条件 $\Phi \stackrel{\text{def}}{=} (x+2=y)$

を考える．事前条件 Ψ が true であるとは，初期状態 m_1 と m_2 が何でもよいことを意味する．事後条件 Φ を確率分布に持ち上げた関係 $\mathcal{L}(x+2=y)$ は「$x+2$ の確率分布と y の確率分布が等しいこと」を表す †9．

それでは，ホーア式 $\models \mathsf{G}_1 \sim \mathsf{G}_2 \colon \Psi \Rightarrow \Phi$，つまり

$$\models x \xleftarrow{\$} \{0,1\} \sim y \xleftarrow{\$} \{2,3\} \colon \mathsf{true} \Rightarrow (x+2=y)$$

が成り立つことを確かめてみよう．

任意の初期状態 m_1 と m_2 から G_1 と G_2 をそれぞれ実行すると，x の値は，確率 $1/2$ で 0，確率 $1/2$ で 1 となる．すなわち x は集合 $\{0,1\}$ 上の一様分布に従う．このとき $x+2$ の確率分布は，$\{2,3\}$ 上の一様分布であり，y の確率分布と同一である．つまり，G_1 実行後の状態の確率分布 $[\![\mathsf{G}_1]\!](m_1)$ と G_2 実行後の状態の確率分布 $[\![\mathsf{G}_2]\!](m_2)$ の間に，関係 $\mathcal{L}(x+2=y)$ が成り立つ．

この具体例の $[\![\mathsf{G}_1]\!](m_1)$ と $[\![\mathsf{G}_2]\!](m_2)$ は，同一の確率分布ではないけれども，事後条件 $x+2=y$ を介することにより，同一の確率分布となるものである．

なお，状態の確率分布の間の関係 $\mathcal{L}(x+2=y)$ が成り立っていても，状態の間の関係 $x+2=y$ 自体は必ずしも成り立たないことに注意したい．x と y は独立にランダムに選ばれており，例えば，$x+2=2$ かつ $y=3$ となる場合がある．

3.4 証明図とその書き方

さて，確率関係ホーア論理において，式（ホーア式や確率の等式・不等式）の証明をどのように記述するのかを大雑把に見てみよう †10．

3.4.1 証明図とは

式の証明には，次の形の**推論規則**を用いる．

$$\frac{J_1 \quad J_2 \quad \cdots \quad J_n}{J}\ [\text{規則の名称}]$$

横線の上にある式 J_1, J_2, …, J_n を**前提**と呼び，横線の下にある式 J を**結論**と呼ぶ．

この推論規則は，「前提 J_1, J_2, …, J_n が成り立つとき，結論 J が成り立つ」ことを表す．（逆は必ずしも成り立たない．すなわち，結論 J が成り立つときに，前提 J_1, J_2, …, J_n が成り立つとは限らない．）

前提の個数 n が 0 のとき，すなわち前提条件なしに必ず式 J が成り立つとき，J を**公理**と呼ぶ．

確率関係ホーア論理には，3.5 節に示すように，推論規則がたくさんある．式を証明するためには，このような推論規則を組み合わせて，**証明図**を作成する．証明図は，図 8 のような木構造をしている．

図 8 の証明図の一番下にある式 J（木構造の根の部分）が **証明したいこと**である．式 J の証明図を**上から下に読んでみよう**．まず，証明図の一番上にある式 J_{11}, J_{12}, J_2（木構造の葉の部分）が公理であり，前提

†8 通常のホーア論理のホーア式は，一つのプログラム G とその事前条件 Ψ および事後条件 Φ からなる三つ組 $\{\Psi\}\,\mathsf{G}\,\{\Phi\}$ であり，**ホーアの三つ組**（Hoare triple）と呼ばれる．一方，関係ホーア論理では二つのプログラムの間の関係を扱うため，**四つ組**である．

†9 より厳密に言うと，$([\![\mathsf{G}_1]\!](m_1))\ \mathcal{L}(\Phi)\ ([\![\mathsf{G}_2]\!](m_2))$ が表すのは，「$x+2$ の値の確率分布」と「y の値の確率分布」が等しくなるような関係が，「G_1 実行後の状態分布 $[\![\mathsf{G}_1]\!](m_1)$」と「$\mathsf{G}_2$ 実行後の状態分布 $[\![\mathsf{G}_2]\!](m_2)$」の間に成り立つということである．

†10 本稿は論理学の知識を仮定せずに説明しているつもりなので，証明論や定理証明器を熟知している読者は 3.4 節を読み飛ばしてほしい．そうでない読者は，当たり前に見えても，証明を機械的に読んで検査する計算機になったつもりで，辛抱強く読んでほしい．

$$\dfrac{\dfrac{\dfrac{}{J_{11}}\,[\text{規則 c}] \quad \dfrac{}{J_{12}}\,[\text{規則 d}]}{J_1}\,[\text{規則 b}] \quad \dfrac{}{J_2}\,[\text{規則 e}]}{J\ (\textbf{証明したいこと})}\,[\text{規則 a}]$$

図 8: 証明図

なしに成り立っている．公理 J_{11} と J_{12} に対して，規則 b を用いると，式 J_1 が得られる．式 J_1 と公理 J_2 に対して，規則 a を用いると，証明したい式 J が得られる．

3.4.2 証明図の書き方

では，どうやって図 8 の証明図を作り上げるのか，手順を説明しよう．最初に証明したい式 J を書き，そこから徐々に証明図を**下から上へ書いていく**．

証明したい式 J に適用できる規則を探してみて，規則 a を適用できると気付き，証明図の一部分

$$\dfrac{J_1 \qquad J_2}{J}\,[\textbf{規則 a}]$$

が得られたとしよう．これは「式 J_1 と J_2 が成り立つならば，式 J が成り立つ」ことを表す．したがって，式 J_1 と J_2 が証明できれば，式 J が証明できたことになる．

そういうわけで，**次に証明したいこと**は式 J_1 と J_2 である [†11]．証明したい式 J_2 に対して規則 e を適用できたとしよう．

$$\dfrac{J_1\ (\text{未証明}) \qquad \dfrac{}{J_2}\,[\textbf{規則 e}]}{J}\,[\text{規則 a}]$$

式 J_2 の上には式が現れない．つまり，J_2 は公理であり，前提なしに成り立っている．

J_2 に関しては，これ以上証明することは何も残っていないから，**次に証明したいこと**は式 J_1 である．J_1 に規則 b を適用できて証明図の一部分

$$\dfrac{\dfrac{J_{11}\ (\text{未証明}) \quad J_{12}\ (\text{未証明})}{J_1}\,[\textbf{規則 b}] \quad \dfrac{}{J_2}\,[\text{規則 e}]}{J\ (\textbf{証明したいこと})}\,[\text{規則 a}]$$

が得られたとしよう．さらに，これまでの要領で規則 c と規則 d を適用し，式 J_{11} と J_{12} が公理であることを示せば，図 8 の証明図が得られる．図 8 の証明図では，すべての葉が公理であり，これ以上証明することは何もないから，証明は完成している．

†11 しかし，もしかしたら J_1 か J_2 が証明できないかもしれない．その場合は，証明したい式 J に対し，規則 a 以外の別の規則を適用することを考える．

3.5 確率関係ホーア論理の推論規則

それでは，証明図を書く際に，具体的にどんな推論規則を使うのかを見てみよう．

3.5.1 推論規則 PrEq と PrLe

まず，確率の等式や不等式を証明するために，どのような確率関係ホーア論理のホーア式を証明すればよいのかを考えてみよう．

初期メモリ状態 m_1 と m_2 において $m_1\,\Psi\,m_2$ が成り立つときに，m_1 でゲーム G_1 を実行し，m_2 で G_2 を実行し，事後条件 Φ が成り立つ場合を考える．このとき，ホーア式 $\models \mathsf{G}_1 \sim \mathsf{G}_2 : \Psi \Rightarrow \Phi$ が成り立つ．

事後条件 Φ が成り立つときに，論理式 A と B が論理同値である，すなわち

$$\models \Phi \Rightarrow (A \Leftrightarrow B)$$

が成り立つものとする．すると，事後条件 Φ が成り立つとき「A が成り立つ確率」と「B が成り立つ確率」が等しい．よって，「G_1 実行後に A が成り立つ確率」と「G_2 実行後に B が成り立つ確率」も等しい．すなわち，

$$\Pr[\,\mathsf{G}_1, m_1 : A\,] = \Pr[\,\mathsf{G}_2, m_2 : B\,]$$

である．これをまとめたものが推論規則 PrEq である．

$$\dfrac{\models m_1 \Psi m_2 \ \text{ かつ } \ \models \mathsf{G}_1 \sim \mathsf{G}_2 : \Psi \Rightarrow \Phi \quad \text{かつ } \models \Phi \Rightarrow (A \Leftrightarrow B)}{\Pr[\,\mathsf{G}_1, m_1 : A\,] = \Pr[\,\mathsf{G}_2, m_2 : B\,]}\,[\text{PrEq}]$$

不等式に関しては，以下の推論規則 PrLe がある．

$$\dfrac{\models m_1 \Psi m_2 \ \text{ かつ } \ \models \mathsf{G}_1 \sim \mathsf{G}_2 : \Psi \Rightarrow \Phi \quad \text{かつ } \models \Phi \Rightarrow (A \Rightarrow B)}{\Pr[\,\mathsf{G}_1, m_1 : A\,] \le \Pr[\,\mathsf{G}_2, m_2 : B\,]}\,[\text{PrLe}]$$

3.5.2 推論規則 Rnd

次に，整数値の確率的代入の場合を見てみよう．具体例として，3.3.2 節で見たホーア式

$$\models x \xleftarrow{\$} \{0,1\} \sim y \xleftarrow{\$} \{2,3\} : \mathsf{true} \Rightarrow (x + 2 = y)$$

を導くために，どんな推論規則を使うのかを考えたい．

そもそも，このホーア式の意味は，

> 任意の初期メモリ状態 m_1 と m_2 において，それぞれ $x \xleftarrow{\$} \{0,1\}$ と $y \xleftarrow{\$} \{2,3\}$ を実行したときに，(前者の) $x+2$ の確率分布と（後

者の）y の確率分布が同一である，

というものだった．x の確率分布と y の確率分布がこのような関係を満たすことを導くためには，$f(v) \stackrel{\text{def}}{=} v+2$ で定義される全単射 $f : \{0,1\} \to \{2,3\}$ を考えて，

各 $v \in \{0,1\}$ に対し，「x の値が v である確率」と「y の値が $f(v)$ である確率」が等しいこと

を示せばよい．実際，いずれの確率も $\frac{1}{2}$ で等しい．

以上の議論を一般化しよう．確率分布 d_i に対し，確率が 0 でない要素の集合を $\mathsf{supp}(d_i)$ で表し，値 v を取る確率を $d_i(v)$ と書くことにする．整数値の確率分布 d_1 と d_2 として，ある全単射 $f : \mathsf{supp}(d_1) \to \mathsf{supp}(d_2)$ が存在して，すべての $v \in \mathsf{supp}(d_1)$ に対して，

$$d_1(v) = d_2(f(v))$$

を満たすようなものを考える．このとき，確率的代入のための推論規則 Rnd_f は以下のようになる [†12]．

$$\frac{\models \Psi \Leftrightarrow \bigwedge_{v \in \mathsf{supp}(d_1)} \Phi[v/x,\, f(v)/y] \wedge d_1(v) = d_2(f(v))}{\models x \stackrel{\$}{\leftarrow} d_1 \sim y \stackrel{\$}{\leftarrow} d_2 : \Psi \Rightarrow \Phi}\ [\mathrm{Rnd}_f]$$

この規則 Rnd_f を前述のホーア式に適用してみよう．$d_1 \stackrel{\text{def}}{=} \{0,1\}$，$d_2 \stackrel{\text{def}}{=} \{2,3\}$，$\Psi \stackrel{\text{def}}{=} \mathsf{true}$，$\Phi \stackrel{\text{def}}{=} (x+2=y)$ とおくと，前述のホーア式は

$$\models x \stackrel{\$}{\leftarrow} d_1 \sim y \stackrel{\$}{\leftarrow} d_2 : \mathsf{true} \Rightarrow \Phi$$

と書ける．d_1 から d_2 への全単射 $f(v) \stackrel{\text{def}}{=} v+2$ に対し，

$$\begin{aligned} &\bigwedge_{v \in \{0,1\}} \Phi[v/x,\, f(v)/y] \wedge d_1(v) = d_2(f(v)) \\ \Leftrightarrow\ &\bigwedge_{v \in \{0,1\}} (x+2=y)[v/x,\, f(v)/y] \wedge \tfrac{1}{2} = \tfrac{1}{2} \\ \Leftrightarrow\ &\bigwedge_{v \in \{0,1\}} (v+2 = f(v)) \\ \Leftrightarrow\ &\mathsf{true} \stackrel{\text{def}}{=} \Psi \end{aligned}$$

であるから，このホーア式が Rnd_f を用いて導かれる．

3.5.3 推論規則 Assn

次に，整数値の決定的代入の場合を見てみよう．決定的代入 $x \leftarrow e$ は，数式 e を変数 x に確率 1 で代入する確率的代入と見なせる．よって，規則 Rnd_f において e を e' に対応させる全単射 f を考えればよく，決定的代入の推論規則 Assn は以下のようになる．

$$\frac{\models \Psi \Leftrightarrow \Phi[e/x,\, e'/y]}{\models x \leftarrow e \sim y \leftarrow e' : \Psi \Rightarrow \Phi}\ [\mathrm{Assn}]$$

3.5.4 推論規則 Seq

次に，ゲームを逐次実行する場合を見てみよう．「c_1 を実行して c_1' を実行するゲーム $c_1; c_1'$」と「c_2 を実行して c_2' を実行するゲーム $c_2; c_2'$」の間に成り立つ性質を証明したいとき，ゲームをそれぞれ分割して，

- c_1 と c_2 の間の性質 $\models c_1 \sim c_2 : \Psi \Rightarrow \Theta$ の証明
- c_1' と c_2' の間の性質 $\models c_1' \sim c_2' : \Theta \Rightarrow \Phi$ の証明

に証明の作業を分解できる．そのための推論規則が以下に示す規則 [Seq] である．

$$\frac{\models c_1 \sim c_2 : \Psi \Rightarrow \Theta \qquad \models c_1' \sim c_2' : \Theta \Rightarrow \Phi}{\models c_1; c_1' \sim c_2; c_2' : \Psi \Rightarrow \Phi}\ [\mathrm{Seq}]$$

この推論規則を使うには，①ゲームをどの部分で分割するか，②どのような Θ を中間条件として使えばよいのか，をよく考える必要がある．

3.5.5 推論規則 Swap

次に，ゲームを変形する推論規則を見てみよう．具体例としてランダムな整数を 2 回生成するゲームを考えてみよう．$X = \{0,1\}$，$Y = \{2,3\}$ とおく．

$$\models x_1 \stackrel{\$}{\leftarrow} X; y_1 \stackrel{\$}{\leftarrow} Y \sim y_2 \stackrel{\$}{\leftarrow} Y; x_2 \stackrel{\$}{\leftarrow} X : \mathsf{true} \Rightarrow x_1 = x_2$$

このホーア式では，x_i と y_i のどちらを先に生成しても，x_1 の値の確率分布と x_2 の値の確率分布が同じであることを主張している．実際，x_i と y_i は独立に選ばれているから，このホーア式は成り立つ．

このホーア式を証明するためには，プログラムの順序を入れ替えるための推論規則を使う．プログラム c_2 と c_3 が独立である場合，すなわち

- c_2 が c_3 に現れる変数に書き込まず，
- c_3 も c_2 に現れる変数に書き込まず，
- c_2 も c_3 も共通の変数を持たない

場合を考えよう．（上の例だと $c_2 \stackrel{\text{def}}{=} x_1 \stackrel{\$}{\leftarrow} X$ と $c_3 \stackrel{\text{def}}{=} y_1 \stackrel{\$}{\leftarrow} Y$ は独立である．）このとき，ゲームの中で c_2 と c_3 の実行順序を入れ替えても，何も変わらない．そこで，実行順序の入れ替えの推論規則を適用できる．

$$\frac{\models c_1; c_3; c_2; c_4 \sim c' : \Psi \Rightarrow \Phi}{\models c_1; c_2; c_3; c_4 \sim c' : \Psi \Rightarrow \Phi}\ [\mathrm{Swap}]$$

順序を入れ替える c_2 と c_3 の独立性を確かめなけれ

†12 単純化のため，本稿では全単射 f が存在する場合を紹介している．例えば，集合 $\mathsf{supp}(d_1)$ と $\mathsf{supp}(d_2)$ の要素数が異なる場合には，$\mathsf{supp}(d_1)$ から $\mathsf{supp}(d_2)$ への全単射が存在しないため，規則 Rnd_f を適用できない．一般の場合の推論規則は [9] を参照．

ば，この推論規則を適用できないことに注意したい．なお，右側のゲームで順序を入れ替える規則もある．

3.5.6 推論規則 Skip

他にも非常に多くの推論規則があるが，本節では最後に，何もしないゲーム skip についての規則

$$\frac{\models \Phi \Rightarrow \Psi}{\models \mathsf{skip} \sim \mathsf{skip} : \Phi \Rightarrow \Psi}\ [\mathrm{Skip}]$$

を見てみよう．この規則は，横線の下のホーア式を証明するためには，一階述語論理式 $\Phi \Rightarrow \Psi$ を証明すれば十分であることを表している．

3.6 確率関係ホーア論理で安全性証明を形式化

前節までは確率関係ホーア論理における証明について見てきた．道具がすべてそろったから，いよいよ暗号系の安全性証明を形式化する方法を紹介しよう．2.4 節で紹介した ElGamal 暗号の IND-CPA 安全性の証明を例にとって説明する．

3.6.1 証明したいことの形式化

暗号系の IND-CPA 安全性は，2.2 節で述べたように，ゲームを用いて定義する．IND-CPA ゲームを pWhile 言語（3.1 節）で記述したものを図 9 に示す．

$(pk, sk) \leftarrow \mathcal{KG}(\eta);$
$(m_0, m_1) \leftarrow A_0(pk);$
$b \xleftarrow{\$} \{0,1\};$
$c \leftarrow \mathcal{E}(pk, m_b);$
$b' \leftarrow A_1(c);$
$res \leftarrow (b = b');$

図 9: 言語 pWhile で記述したゲーム $\mathsf{CPA}(\mathcal{S}, \mathcal{A})$．
ただし $\mathcal{S} = (\mathcal{KG}, \mathcal{E}, \mathcal{D})$ は公開鍵暗号，$\mathcal{A} = (A_0, A_1)$．

ElGamal 暗号の IND-CPA 安全性について証明したいことは，任意の PPT 攻撃者 $\mathcal{A}$ に対して，

$$|\Pr[\mathsf{CPA}(\mathsf{ElGamal}, \mathcal{A}) : res] - 1/2|$$

が無視できるほど小さいことである（2.4.1 節）．

確率関係ホーア論理の枠組みでは，攻撃の確率が「無視できるほど小さい」こと自体を証明する代わりに，「攻撃の確率を表現した式」が成り立つことを証明する．例えば，ElGamal 暗号の IND-CPA 安全性の証明では，「IND-CPA 安全性を破る確率」が「DDH 仮定を破ってしまう（無視できるほど小さな）確率」に等しいことを証明する．すなわち，任意の PPT 攻撃者 $\mathcal{A}$ に対して，ある PPT 攻撃者 $\mathsf{Inv}_{\mathcal{A}}$ が存在し，

$$\begin{aligned}&|\Pr[\mathsf{CPA}(\mathsf{ElGamal}, \mathcal{A}) : res] - 1/2| \\ &= |\Pr[\mathsf{DDH}_0(\mathsf{Inv}_{\mathcal{A}}) : res] - \Pr[\mathsf{DDH}_1(\mathsf{Inv}_{\mathcal{A}}) : res]|\end{aligned}$$

が成り立つことを証明する．

3.6.2 安全性証明における攻撃者の形式化

安全性証明で考えたい攻撃者は，特定の攻撃者ではなくて，**任意の** PPT 攻撃者である．そこで，攻撃者は，具体的なゲームとしてではなく，ゲームの型（**モジュール型**）として形式化する．

IND-CPA 安全性の証明では，図 10 のとおり，攻撃者 $\mathcal{A}$ を二つの手続き A_0 と A_1 からなるゲームの型として形式化する．ここで，手続き A_0 と A_1 の記

module type $\mathcal{A} = \{$
  proc $A_0(pk : pkey) : plaintext \times plaintext$
  proc $A_1(c : ciphertext) : bool$
$\}$

図 10: IND-CPA 攻撃者を記述するゲームの型 $\mathcal{A}$

述では，引数と返り値の型だけを記述し，具体的なプログラムを与えない．A_0 は，公開鍵を受け取って，平文の対を返す任意の手続きであり，A_1 は，暗号文を受け取って，ブール値を返す任意の手続きである．

手続き A_0 と A_1 は，ひとつのモジュール型 $\mathcal{A}$ の宣言に記述され，攻撃者の内部状態 $\mathsf{glob}\mathcal{A}$ を共有している．そのため，図 9 と図 10 の A_0 と A_1 の引数や返り値には $\mathsf{glob}\mathcal{A}$ の受け渡しを記述していない．

最後に，注意したい点として，「攻撃者が確率的多項式時間（PPT）の計算しかできない」という条件は，じつは攻撃者 $\mathcal{A}$ の記述自体には現れていない．実際のところ，ElGamal 暗号の IND-CPA 安全性を証明するには，「攻撃者が DDH 問題を解けない」という仮定だけを使えば十分である．言い換えれば，「DDH 問題を解けない任意の攻撃者」というより強い攻撃者のもとで安全性を証明できる．したがって，攻撃者の計算能力の限界に関しては，「DDH 問題を解けない」という仮定だけを記述している †13．

†13 オラクルにアクセスする攻撃者を扱う場合だったら，多項式回のアクセスという条件が必要になってくる．

3.6.3 安全性証明の形式化の概要

確率関係ホーア論理で形式化したい安全性証明は，2.4.2節の図5のようなゲーム列を用いた証明である．

確率関係ホーア論理を用いて証明を形式化する際には，3.4節で説明したように，証明図を作成する．証明図では，証明のすべての推論に対して，それぞれいかなる推論規則を用いて導いたのかを，推論を飛躍させることなく厳密に記述する．

しかし，一般に暗号系の安全性証明は非常に煩雑であるから，証明のすべての推論を同じレベルで記述すると，証明図が膨大で複雑になり，読みにくくなる．

そこで，暗号系の安全性証明を記述する際には，証明を次の二つの階層に分けて記述する．

(A) 確率ホーア論理の枠組みの中で，「ゲーム変換の各ステップに関する補題」をそれぞれ示す．(2.4.3節の各補題 (1)〜 (5) の証明の形式化.)

(B) 確率ホーア論理の枠組みの外で，「ゲーム変換の各ステップに関する補題」を組み合わせて，確率の議論を行い，証明の全体を完成させる．(2.4.2節の証明の全体構造の記述の形式化.)

実際のところ，「ゲーム変換の各ステップに関する補題」の証明も，すべての推論を同じレベルで記述すると，煩雑になりがちである．そのため，安全性証明の形式化の詳細は，4.2節において，検証ツール EasyCrypt を紹介しながら説明する．

4 コンピュータ上で安全性を証明

前節で紹介した確率関係ホーア論理では，安全性証明の各推論を飛躍させることなく厳密に記述するため，証明の正しさを厳密に確認できそうである．

しかし，安全性証明の導出を一切省略せずに，人手で書くわけにもいかない．長い証明を書くのは大変だし，それを読む人も退屈する．また，人の行いに間違いはつきものであり，長い証明を紙の上に書けば，思わぬミスをしてしまうものである．

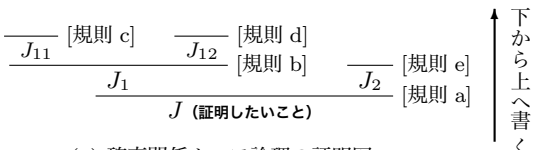
人手で安全性証明を書くたびに，間違いを見つけられては，人手で新しい証明を書き直す，ということの繰り返しを防ぐには，**コンピュータを用いて証明を記述し，機械的に正しさを確認する**ことが有効である．

コンピュータ上で証明を記述する利点は，証明の厳密さだけではない．例えば，確率関係ホーア論理を拡張したプログラミング言語を用いると，機能ごとにモジュールに分割できて，複合的で階層的に証明を記述できるため，証明を読む人（特に暗号の研究者や技術者）にとって，より分かりやすくなる．また，ツールによる支援により，証明の瑣末な部分を自動化したり，証明にかかる時間や労力を減らしたりできる．

4.1 検証ツール EasyCrypt

検証ツール EasyCrypt [11] は，確率関係ホーア論理に基づいて，暗号系の安全性証明の記述を支援し，証明の正しさを機械的に確認するためのソフトウェアである．このツールは，暗号系の計算量的安全性（や情報理論的安全性）の検証ツールの中で，最も注目を集めているものである．2011年に Barthe らによって発表され，2016年現在も開発が続き，進化し続けている．本節では EasyCrypt の概要を紹介する．

まず，EasyCrypt における証明記述のやり方を，確率関係ホーア論理の証明図と比べながら，説明しよう．



(a) 確率関係ホーア論理の証明図

```
proof.
  a.
  e.
  b.
  c.
  d.
qed.
```

上から下へ書く

(b) EasyCrypt の証明記述

図 11: 証明図に対応する EasyCrypt の証明記述

証明図では，図11のとおり，一番下に「証明したいこと」を書き，推論規則を次々に適用しながら，下から上へ書いていき，すべての葉が公理であることを確認できると，証明図が完成するのであった．EasyCrypt における証明記述は，大雑把にいうと，こうして適用していった推論規則の列である．証明図と EasyCrypt の記述の間の対応は，図11のとおりである．

厳密に言うと，EasyCrypt の証明記述には推論規則の名前ではなく，**タクティック**（tactic）の名前を書く．タクティックは，必ずしも一つの推論規則ではなく，一つ以上の推論規則の列に対応する †14．

EasyCrypt の見かけや使い方は，定理証明支援系 Coq[29] を参考に作られている．EasyCrypt のユーザは，テキストエディタ Emacs の画面上で，EasyCrypt の出力表示を見てタクティックを入力するという作業を繰り返すことで，証明を**対話的に**作成できる．

証明を対話的に作成する手順は以下のとおりである．

1. 最初にユーザが「証明したいこと」（**ゴール**）を EasyCrypt に入力する．
2. EasyCrypt が表示した「次に証明すべきこと」（その時点でのゴール）に対して，ユーザは「次に適用するタクティック」を入力する．
3. EasyCrypt は，タクティックがゴールに適用できるとき，「次に証明すべきこと」（ゴール）を表示する．そうでないときはエラーを表示する．
4. これらを繰り返すうちに，証明すべきこと（ゴール）がなくなり，証明が完成する．

4.2 EasyCrypt による安全性証明の例

では，ElGamal 暗号の IND-CPA 安全性の証明を例にとり，EasyCrypt で証明を記述する方法を見てみよう．記述したい証明は，2.4.2 節の図 5 のゲーム列による証明だが，本節ではその一部分を解説する．

4.2.1 暗号系の記述

EasyCrypt では，専用のプログラミング言語を用いて，暗号系の仕様と安全性定義をプログラムとして記述する．詳細には立ち入らないが，例えば，ElGamal 暗号は図 12 のように記述される．冒頭ではライブラリをインポートし，データ型を定義している．ElGamal 暗号を記述するゲームを，公開鍵暗号方式を表す型 `Scheme` のモジュール `ElGamal` として定義している．

モジュールの記述は抽象化されており，巡回群 $\mathbb{Z}_q^*$ の詳細はライブラリに含まれる．そのため，例えば q の選び方の条件が，図 12 の定義にも安全性証明にも現れず，DDH 仮定に隠れていることに注意したい．

```
require import DiffieHellman FMap.
type pkey       = group.
type skey       = F.t.
type plaintext  = group.
type ciphertext = group * group.
require PKE.
clone import PKE as PKE_ with
  type pkey <- pkey,
  type skey <- skey,
  type plaintext <- plaintext,
  type ciphertext <- ciphertext.

module ElGamal : Scheme = {
  proc kg(): pkey * skey = {
    var sk;
    sk = $FDistr.dt;
    return (g^sk, sk);
  }
  proc enc(pk:pkey, m:plaintext) :
       ciphertext = {
    var y;
    y = $FDistr.dt;
    return (g^y, (pk^y) * m);
  }
  proc dec(sk:skey, c:ciphertext) :
       plaintext option = {
    var gy,gxym;
    (gy, gxym) = c;
    return Some(gxym * gy^(-sk));
  }
}.
```

図 12: EasyCrypt における ElGamal 暗号の記述

4.2.2 補題の記述

ゲーム変換のひとつのステップについての補題を記述する方法を紹介する．図 13 に再掲するゲーム G_1 から G_2 への変換を例にとって説明する．なお，G_1 と G_2 の違いは，破線で囲まれた部分（ビット b の生成箇所と，手続き A_1 が受け取るメッセージ）である．

証明したいことは，任意の PPT 攻撃者 $\mathcal{A}$ に対し，

$$\Pr[\mathsf{G}_1(\mathcal{A}) : res] = \Pr[\mathsf{G}_2(\mathcal{A}) : res]$$

が成り立つこと（2.4.2 節の補題 (5)）である．EasyCrypt では，この補題を以下のように記述する．

```
lemma lem_G1_G2 (A<:Adversary) &m :
   Pr[G1(A).main() @ &m : res]
 = Pr[G2(A).main() @ &m : res].
```

なお，`&m` はゲームを実行する前のメモリ状態を表す．

†14 証明の記述と作業を簡潔にするために，定理証明支援系 Coq と同様，タクティカル（他のタクティックを引数に取るタクティック）も用意されている．

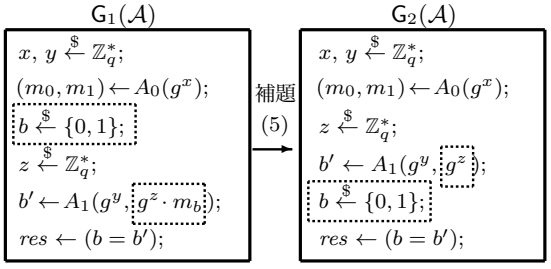


図 13: 補題 (5) のゲーム変換

```
proof.
 byequiv => //.
 proc.
 swap{2} 10 -4.
 call (_:true).
 wp.
 rnd
  (fun z, z+log(if b then m1 else m0){2})
  (fun z, z-log(if b then m1 else m0){2}).
 rnd.
 call (_:true).
 wp; rnd; rnd.
 skip.
 progress; smt.
qed.
```

図 14: EasyCrypt による lem_G1_G2 の証明

4.2.3 証明の記述と方針

確率関係ホーア論理に基づく証明は，定義（例えば暗号系の仕様や IND-CPA 安全性の定義）や証明したい補題と同じファイルに記述できる．EasyCrypt による証明の記述は，証明図を作成するのに用いたタクティックの列となっている．例えば，補題 lem_G1_G2 の証明は，図 14 のように書ける．

証明の方針を説明しよう．最初に，補題 lem_G1_G2 の確率の等式をホーア式に変形する．次に，$\models \mathsf{G}_1 \sim \mathsf{G}_2 : \mathsf{pre} \Rightarrow \mathsf{post}$ という形のホーア式に対し，推論規則 Seq と他の規則から構成されるタクティックを次々に用いて，ゲーム G_1 と G_2 の末尾から各々ひとつずつ命令文を削っていく．両ゲームから命令文を削る度に，削られた命令文の間の関係が事後条件 post に加わっていく．最終的に二つのゲームは空っぽになって，$\models \mathsf{skip} \sim \mathsf{skip} : \mathsf{true} \Rightarrow \mathsf{post}'$ という形のホーア式が得られ，一階述語論理式 post' を証明すれば十分となる．post' は人に読みにくい巨大な論理式になりがちなので，外部ツールで自動証明を試みる．

4.2.4 図 14 の証明の各行の説明

証明 2 行目：確率の等式からホーア式へ

補題 lem_G1_G2 の証明では，最初に，推論規則 PrEq（3.5.1 節）に対応するタクティック byequiv => //（図 14 の証明 2 行目）を適用する．これにより，確率の議論を確率ホーア論理の推論に持ち込むことができ，次に証明したいことが以下のとおり表示される．

```
Current goal
Type variables: <none>
A : Adversary
&m: memory
------------------------------------
pre  = (glob A){2} = (glob A){m} /\
       (glob A){1} = (glob A){m}
   G1(A).main ~ G2(A).main
post = ={res}
```

$\mathcal{A}$ は任意の攻撃者，m は任意の初期メモリ状態を表す．pre はホーア式の事前条件，post は事後条件を表す．

二つのゲームに現れる同じ変数を区別するために，ゲーム G_i の各変数 α を $\alpha\{i\}$ で表そう．G_1 の変数 $\mathsf{glob}\,\mathcal{A}$ は $\mathsf{glob}\,\mathcal{A}\{1\}$，$\mathsf{G}_2$ の変数 $\mathsf{glob}\,\mathcal{A}$ は $\mathsf{glob}\,\mathcal{A}\{2\}$ と書ける．={res} は両ゲームの変数 *res* の値が等しいこと，すなわち $\mathtt{res}\{1\} = \mathtt{res}\{2\}$ の略記である．ホーア式の意味は，「ゲーム G_1 と G_2 の実行前に攻撃者の状態 $\mathsf{glob}\,\mathcal{A}$ が等しいとき，G_1 と G_2 の実行後に得られる変数 *res* の値の確率分布が等しい」である．

証明 3 行目：ゲームの手続きを展開する

次に，タクティック proc（図 14 の証明 3 行目）で，手続き main を展開すると，次のホーア式を得る．

```
pre  = (glob A){2} = (glob A){m} /\
       (glob A){1} = (glob A){m}

 x =$ FDistr.dt    (1) x =$ FDistr.dt
 y =$ FDistr.dt    (2) y =$ FDistr.dt
 gx = g^x          (3) gx = g^x
 gy = g^y          (4) gy = g^y
 (m0,m1)=A.A0(gx)  (5) (m0,m1)=A.A0(gx)
 b =${0,1}         (6) z =$FDistr.dt
 z =$FDistr.dt     (7) gz = g^z
 gz = g^z          (8) c = (gy, gz)
 c = (gy, gz * (b? m1:m0))
                   (9) b' = A.A1(c)
 b' = A.A1(c)     (10) b =${0,1}

post = (b{1} = b'{1}) = (b{2} = b'{2})
```

行番号の左側が G_1，右側が G_2 のプログラムである．

証明 4 行目：命令文の順番を入れ替える

次に，タクティック `swap{2} 10 -4`（図 14 の証明 4 行目）を適用すると，ゲーム G_2 の (10) `b =${0,1}` が 4 行上に移動し，ゲーム G_1 の (6) と一致する．

```
   ......
 b =${0,1}          (6) b =${0,1}
 z =$FDistr.dt      (7) z =$FDistr.dt
 gz = g^z           (8) gz = g^z
 c = (gy, gz * (b? m1 : m0))
                    (9) c = (gy, gz)
 b' = A.A1(c)      (10) b' = A.A1(c)

post = (b{1} = b'{1}) = (b{2} = b'{2})
```

証明 5 行目：攻撃者の手続き呼び出しを取り除く

さて，次に，左右のゲームで両方とも (10) が `b' = A.A1(c)` であることに着目し，攻撃者の性質に関する以下のホーア式を用いることを考える．

$$\models \mathtt{b'} \leftarrow \mathcal{A}.A_1(\mathtt{c}) \sim \mathtt{b'} \leftarrow \mathcal{A}.A_1(\mathtt{c}):$$
$$(\mathsf{glob}\mathcal{A}\{1\} = \mathsf{glob}\mathcal{A}\{2\}) \wedge (\mathtt{c}\{1\} = \mathtt{c}\{2\})$$
$$\Rightarrow (\mathsf{glob}\mathcal{A}\{1\} = \mathsf{glob}\mathcal{A}\{2\}) \wedge (\mathtt{b'}\{1\} = \mathtt{b'}\{2\})$$

このホーア式では，左右のゲームで攻撃者の内部状態 $\mathsf{glob}\mathcal{A}\{1\}$ と $\mathsf{glob}\mathcal{A}\{2\}$ が等しいときに，攻撃者の手続き A_1 を同一の入力 `c` でそれぞれ実行すると，A_1 実行後の内部状態 $\mathsf{glob}\mathcal{A}\{1\}$ と $\mathsf{glob}\mathcal{A}\{2\}$ も等しく，A_1 の返り値 `b'{1}` と `b'{2}` も等しくなることを表す．

このホーア式の適用に対応するタクティックが `call (_:true)`（図 14 の証明 5 行目）である．これにより，両方のゲームから (10) `b' = A.A1(c)` が除かれ，事後条件に $(\mathsf{glob}\mathcal{A}\{1\} = \mathsf{glob}\mathcal{A}\{2\}) \wedge (\mathtt{c}\{1\} = \mathtt{c}\{2\})$ が追加される．

証明 6 行目：決定的代入を取り除く

次に，推論規則 Assn（3.5.3 節）に対応するタクティック `wp`（図 14 の証明 6 行目）を用いる．すると，(8) と (9) の代入で，事後条件が書きかわり，左右のゲームから (8) と (9) が取り除かれる．

```
   ......
 (m0,m1) = A.A0(gx) (5) (m0,m1) = A.A0(gx)
 b =${0,1}          (6) b =${0,1}
 z =$FDistr.dt      (7) z =$FDistr.dt

post = ......
```

証明 7〜9 行目：確率的代入を取り除く

このとき，左右のゲームの最終行 (7) が確率的代入 `z =$FDistr.dt`（すなわち $\mathtt{z} \xleftarrow{\$} \mathbb{Z}_q^*$）であるから，推論規則 `rnd`（3.5.2 節）を適用できる．

ここで注意すべき点は，どのような全単射 f を用いるかである．事後条件の中には，「左側のゲームの $g^{\mathtt{z}\{1\}} * (\mathsf{if}\ \mathtt{b}\{1\}\ \mathsf{then}\ \mathtt{m}_1\{1\}\ \mathsf{else}\ \mathtt{m}_0\{1\})$ の値の確率分布」と「右側のゲームの $g^{\mathtt{z}\{2\}}$ の値の確率分布」が等しいという条件が現れてくる．そのため，3.5.2 節の議論に従うと，全単射

$$g^{\mathtt{z}\{2\}} \mapsto g^{\mathtt{z}\{1\}} * (\mathsf{if}\ \mathtt{b}\{1\}\ \mathsf{then}\ \mathtt{m}_1\{1\}\ \mathsf{else}\ \mathtt{m}_0\{1\})$$

を考えればよい．両辺の対数をとると，全単射

$$f : \mathtt{z}\{2\} \mapsto \mathtt{z}\{1\} + \log_g(\mathsf{if}\ \mathtt{b}\{1\}\ \mathsf{then}\ \mathtt{m}_1\{1\}\ \mathsf{else}\ \mathtt{m}_0\{1\})$$

を考えることになる．

こうしてタクティック `rnd` f f^{-1}（図 14 の証明 7〜9 行目）を適用すると，事後条件が代入によって書き換わり，両方のゲームから (7) が取り除かれる．

証明 10〜12 行目：空になるまで命令文を取り除く

次に，左右のゲームの (6) $\mathtt{b} \xleftarrow{\$} \{0,1\}$ も確率的代入であるから，タクティック `rnd` を適用して，(6) を取り除く（図 14 の証明 10 行目）．今度は全単射として恒等写像を考えればよい．

```
pre  = (glob A){2} = (glob A){m} /\
       (glob A){1} = (glob A){m}

 x =$ FDistr.dt   (1) x =$ FDistr.dt
 y =$ FDistr.dt   (2) y =$ FDistr.dt
 gx = g^x         (3) gx = g^x
 gy = g^y         (4) gy = g^y
 (m0,m1) = A.A0(gx) (5) (m0,m1) = A.A0(gx)

post = ......
```

次に，左右のゲームの (5) がどちらも `(m0,m1) = A.A0(gx)` だから，再びタクティック `call (_:true)` を適用し（図 14 の証明 11 行目），(5) を取り除く．

その後も，決定的代入に対してタクティック `wp` を適用し，確率的代入に対してタクティック `rnd` を適用すると，左右のゲームは空っぽのプログラムとなる．

```
pre  = (glob A){2} = (glob A){m} /\
       (glob A){1} = (glob A){m}

post = ......
```

証明 13 行目：ホーア式から一階述語論理式へ

空のプログラムに対しては，推論規則 Skip（3.5.6 節）に対応するタクティック `skip` を適用する．すると，`pre` → `post` という形の（確率を含まない）一階述語論理式を得る．証明を完成させるには，この論理式が成り立つことを示せば十分である．

証明 14 行目：一階述語論理式の自動証明

この一階述語論理式 `pre` → `post` を証明するために，タクティックの列 `progress; smt` を適用する．すると，`progress` が論理式をより小さな論理式に自動的に分解し，`smt` がこれらを Why3 プラットフォーム [19] のフォーマットに変換して外部ツール（SMT ソルバ）を呼び出し，論理式が成り立つかを判定する．

EasyCrypt では，このように外部ツールを用いて証明の一部を自動化しており，外部ツールの信頼性に依存しているが，定理証明支援系 Coq [29] の上に作られた検証ツール CertiCrypt より使いやすくなっている．

5 関連研究

5.1 ゲーム列による安全性証明の形式化

暗号系の安全性証明を形式化する枠組みやツールは，EasyCrypt 以外にも様々なものが研究されてきた．

5.1.1 最初の検証ツール CryptoVerif

ゲーム列による安全性証明のための最初の検証ツールは CryptoVerif [13] である．CryptoVerif では，**確率プロセス計算**のプロセスとしてゲームをモデル化し，プロセスの間の双模倣性を用いてゲーム変換を形式化する．CryptoVerif を用いて，FDH（Full Domain Hash）署名，Kerberos 認証，SSH トランスポート層プロトコルなどの検証が行われている．

5.1.2 ホーア論理を用いた初期の研究

ゲーム列による安全性証明を**確率ホーア論理**で初めて形式化したのは Corin と den Hartog [16] である．しかし，表現力が不十分で，証明には形式的でない部分もあった．Courant ら [17] は，ホーア論理を拡張し，公開鍵暗号の一般的構成法の IND-CPA 安全性を検証するための（不完全な）自動ツールを開発した．

5.1.3 深い埋め込みによる定式化

定理証明支援系を用いた形式化の研究も，同時期に始まった．Affeldt ら [2] は，初めて定理証明支援系 Coq を用いて，深い埋め込み（deep embedding）によりゲームを定式化し，ゲーム列による PRP/PRF Switching Lemma の証明を形式化した．

Barthe ら [9] は，深い埋め込みによりゲーム列を用いて安全性を証明するための検証ツール CertiCrypt を Coq の上に開発し，確率関係ホーア論理の枠組みで FHD 署名や RSA-OAEP の安全性証明を形式化した．しかし，CertiCrypt は，実数演算の形式化の負担も大きく，自動化もないため，小さな証明であっても膨大な時間と労力がかかるもので，暗号研究者には使えそうにないツールだった．

そこで，Barthe らは CertiCrypt よりも簡単に扱える検証ツールとして，EasyCrypt の開発を 2009 年に開始し，プロトタイプ EasyCrypt 0.2 [11] を発表した．EasyCrypt は Coq の上に作られておらず，CertiCrypt より実行時間が短く，証明の行数も少なくて済む．SMT ソルバを呼び出すことにより，証明の瑣末な部分の自動化が可能で，CertiCrypt よりも使いやすい．

2012 年からは，EasyCrypt 0.2 の記述力を高めた EasyCrypt 1.0 [6] を実装し直している [7]．抽象化のメカニズムにより，複合的で階層的に証明を記述でき，大規模な暗号系の安全性証明を記述しやすくなっている．これにより，検証可能計算や認証付き鍵交換プロトコルの安全性証明も記述できるようになっている．

Barthe らは，この他に，パディングを用いる公開鍵暗号の自動解析ツール ZooCrypt [5] や，ジェネリック群モデルの計算量的仮定を解析するためのツール GenericGroupAnalyzer [8] を開発した．

5.1.4 浅い埋め込みによる定式化

一方，Nowak [24] は，Coq 上で，浅い埋め込み（shallow embedding）によりゲームを定式化し，ElGamal 暗号の IND-CPA 安全性の証明をより簡潔に形式化した．この手法では攻撃者の計算量的限界を扱えなかったため，Nowak ら [25] は，確率ラムダ計算の一種 CSLR の拡張を提案し，Blum-Blum-Shub の擬似乱数生成器を形式化した．最近では，浅い埋め込みによってより少ない労力で証明を行うための新たな枠組みとして，Coq ライブラリ FCF [26] が提案され，検索可能暗号の証明 [27] が形式化されている．

5.2 形式手法の計算論的健全性

従来の形式手法[18]は，暗号プロトコルの設計ミスの発見に役立ってきたが，攻撃の確率や計算量を考慮しておらず，暗号プロトコルの計算量的安全性を検証できなかった．そこで，Abadi と Rogaway の研究[1]以来，「形式手法で攻撃が見つからなかった暗号プロトコルが，どのような条件のもとで計算量的安全性を満たすのか」についての研究（**形式手法の計算論的健全性**）も盛んに行われてきた[4][23][14][15]．

これらの計算論的健全性の研究は，既存の形式手法を計算量的安全性の検証に役立てる上で欠かせない．しかし，計算論的健全性の証明は，非常に煩雑であり，2016 年現在ほとんど注目を集めなくなっている．

6 むすびに

本稿では，暗号系の安全性証明を形式検証する方法，特に，検証ツール EasyCrypt について解説した．

EasyCrypt はまだ実用的とは言い難い．（本稿執筆時点では）C 言語のソースコードを検証することも，検証済みのソースコードを生成することもできない．

そもそも，EasyCrypt による証明が本当に絶対に正しいかどうかも分からない．EasyCrypt は比較的新しい開発中のツールで，バグが見つかることがある．人の行いに間違いはつきもので，キリがない．

それでも，EasyCrypt のような形式検証ツールを使うと，間違いを減らせることは確かである．暗号技術は社会の基盤であり，より信頼できる形で安全性を確かめておく必要があり，形式検証を行う意義がある．

暗号系の形式検証の他の枠組みについて知りたい読者は，日本語の書籍[30]を参照してほしい．

EasyCrypt のことをもっと知りたい読者は，公式ウェブサイト `https://www.easycrypt.info/` からツールを入手して使ってみてほしい．サイトにはチュートリアル，マニュアル，サマースクール等のスライドや練習問題もあり，メーリングリストもある．ただし，EasyCrypt は開発中のツールであり，マニュアルは十分でない．EasyCrypt を使いこなすには，定理証明支援系 Coq[29]を勉強するのが近道かもしれない．

謝辞

原稿完成を辛抱強く待ってくださった編集委員の真野健氏（NTT コミュニケーション科学基礎研究所）と有益なコメントをくださった査読者に感謝します．EasyCrypt Summer School 2013 の情報を提供してくださった Gergely Bana 氏（INRIA）と，有益な助言をくださった櫻田英樹氏（NTT コミュニケーション科学基礎研究所），Reynald Affeldt 氏と照屋唯紀氏（産業技術総合研究所）に感謝します．本稿執筆は JSPS 科研費 JP15H06886 の助成を受けたものです．

参 考 文 献

[1] Abadi, M. and Rogaway, P.: Reconciling Two Views of Cryptography (The Computational Soundness of Formal Encryption), *Journal of Cryptology*, Vol. 15, No. 2(2002), pp. 103 – 127.

[2] Affeldt, R., Tanaka, M., and Marti, N.: Formal Proof of Provable Security by Game-Playing in a Proof Assistant, *in Proc. First International Conference on Provable Security (ProvSec'07)*, 2007, pp. 151–168.

[3] Audebaud, P. and Paulin-Mohring, C.: Proofs of randomized algorithms in Coq, *Sci. Comput. Program.*, Vol. 74, No. 8(2009), pp. 568–589.

[4] Backes, M., Pfitzmann, B., and Waidner, M.: A composable cryptographic library with nested operations, *in Proc. 10th ACM Concerence on Computer and Communications Security (CCS'03)*, 2003, pp. 220–230.

[5] Barthe, G., Crespo, J. M., Grégoire, B., Kunz, C., Lakhnech, Y., Schmidt, B., and Béguelin, S. Z.: Fully automated analysis of padding-based encryption in the computational model, *in Proc. 20th ACM Conference on Computer and Communications Security (CCS'13)*, 2013, pp. 1247–1260.

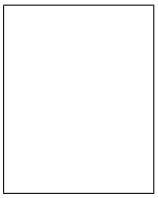
[6] Barthe, G., Dupressoir, F., Grégoire, B., Kunz, C., Schmidt, B., and Strub, P.-Y.: EasyCrypt: A Tutorial, *Foundations of Security Analysis and Design VII*, Springer, 2014, pp. 146–166.

[7] Barthe, G., Dupressoir, F., Grégoire, B., Schmidt, B., and Strub, P.-Y.: Computer-Aided Cryptography: Some Tools and Applications, *in Proc. All about Proofs, Proofs for All*, 2014.

[8] Barthe, G., Fagerholm, E., Fiore, D., Mitchell, J. C., Scedrov, A., and Schmidt, B.: Automated Analysis of Cryptographic Assumptions in Generic Group Models, *Advances in Cryptology - proc. 34th Annual Cryptology Conference (CRYPTO'14) Part I*, 2014, pp. 95–112.

[9] Barthe, G., Grégoire, B., and Béguelin, S. Z.: Formal certification of code-based cryptographic

proofs, *in Proc. 36th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'09)*, 2009, pp. 90–101.
[10] Barthe, G., Grégoire, B., and Béguelin, S. Z.: Probabilistic Relational Hoare Logics for Computer-Aided Security Proofs, *in Proc. 11th International Conference on Mathematics of Program Construction (MPC'12)*, 2012, pp. 1–6.
[11] Barthe, G., Grégoire, B., Heraud, S., and Béguelin, S. Z.: Computer-Aided Security Proofs for the Working Cryptographer, *Advances in Cryptology - proc. 31st Annual Cryptology Conference (CRYPTO'11)*, 2011, pp. 71–90.
[12] Bellare, M. and Rogaway, P.: The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs, *Advances in Cryptology - proc. 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'06)*, 2006, pp. 409–426.
[13] Blanchet, B.: A Computationally Sound Mechanized Prover for Security Protocols, *IEEE Trans. Dependable Sec. Comput.*, Vol. 5, No. 4(2008), pp. 193–207.
[14] Comon-Lundh, H. and Cortier, V.: Computational soundness of observational equivalence, *in Proc. 15th ACM Conference on Computer and Communications Security (CCS'08)*, 2008, pp. 109–118.
[15] Comon-Lundh, H., Hagiya, M., Kawamoto, Y., and Sakurada, H.: Computational Soundness of Indistinguishability Properties without Computable Parsing, *in Proc. 8th International Conference on Information Security Practice and Experience (ISPEC'12)*, 2012, pp. 63–79.
[16] Corin, R. and den Hartog, J.: A Probabilistic Hoare-style Logic for Game-Based Cryptographic Proofs, *in Proc. 33rd International Colloquium on Automata, Languages and Programming (ICALP'06) Part II*, 2006, pp. 252–263.
[17] Courant, J., Daubignard, M., Ene, C., Lafourcade, P., and Lakhnech, Y.: Towards automated proofs for asymmetric encryption schemes in the random oracle model, *in Proc. 15th ACM Conference on Computer and Communications Security (CCS'08)*, 2008, pp. 371–380.
[18] Dolev, D. and Yao, A.: On the security of public key protocols, *IEEE Transactions on Information Theory*, Vol. 29, No. 2(1983), pp. 198–207.
[19] Filliâtre, J. and Paskevich, A.: Why3 - Where Programs Meet Provers, *in Proc. 22nd European Symposium on Programming on Programming Languages and Systems (ESOP'13)*, 2013, pp. 125–128.
[20] Gamal, T. E.: A public key cryptosystem and a signature scheme based on discrete logarithms, *IEEE Trans. Information Theory*, Vol. 31, No. 4(1985), pp. 469–472.
[21] Halevi, S.: A plausible approach to computer-aided cryptographic proofs, IACR Cryptology ePrint Archive, Report 2005/181, 2005.
[22] Jonsson, B., Yi, W., and Larsen, K. G.: Probabilistic extensions of process algebras, *Handbook of process algebra*, (2001), pp. 685–710.
[23] Micciancio, D. and Warinschi, B.: Soundness of Formal Encryption in the Presence of Active Adversaries, *in Proc. Theory of Cryptography Conference (TCC'04)*, Lecture Notes in Computer Science, Vol. 2951, 2004, pp. 133–151.
[24] Nowak, D.: A Framework for Game-Based Security Proofs, *in Proc. 9th International Conference on Information and Communications Security (ICICS'07)*, 2007, pp. 319–333.
[25] Nowak, D. and Zhang, Y.: A Calculus for Game-Based Security Proofs, *in Proc. 4th International Conference on Provable Security (ProvSec'10)*, 2010, pp. 35–52.
[26] Petcher, A. and Morrisett, G.: The Foundational Cryptography Framework, *in Proc. 4th International Conference on Principles of Security and Trust (POST'15)*, 2015, pp. 53–72.
[27] Petcher, A. and Morrisett, G.: A Mechanized Proof of Security for Searchable Symmetric Encryption, *in Proc. IEEE 28th Computer Security Foundations Symposium (CSF'15)*, 2015, pp. 481–494.
[28] Shoup, V.: Sequences of games: a tool for taming complexity in security proofs, IACR Cryptology ePrint Archive, Report 2004/332, 2004.
[29] The Coq Development Team: The Coq proof assistant reference manual: Version 8.5, 2016. `http://coq.inria.fr`.
[30] 萩谷昌己, 塚田恭章 (編): 数理的技法による情報セキュリティ, 日本応用数理学会監修, シリーズ応用数理, 第 1 巻, 共立出版, 2010 年.



川本裕輔

確率的プログラムや暗号プロトコルの検証を研究．2005 年東京大学理学部情報科学科卒業．2007 年同大学大学院情報理工学系研究科コンピュータ科学専攻修士課程修了．2010 年同専攻博士課程修了．博士（情報理工学）．仏国カシャン高等師範学校（ENS Cachan），英国バーミンガム大学，仏国国立情報学自動制御研究所 (INRIA)・理工科学校 (École Polytechnique) にて博士研究員．2015 年より国立研究開発法人 産業技術総合研究所 研究員．