証明可能安全性の形式的証明：定理証明ツールによるゲームの手法

Formal Proof of Provable Security by Game-playing in a Proof Assistant

アフェルト レナルド* 田中三貴† ‡ マーティ ニコラ§

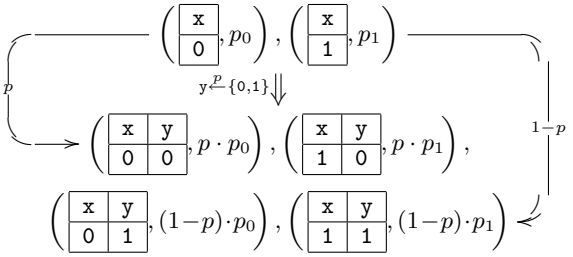
キーワード：安全性証明, 定理証明支援系, ゲーム手法
Keywords: Provable security, Proof assistant, Game-playing

1 初めに

ゲーム列による証明手法は, 証明し易い安全性証明を構成するための方法である. 安全性性質や計算量的問題はプログラムとして記述され, リダクションによる証明はプログラム書換の列として表現される. その明かなプログラム言語的指向性からコンパイラの構成手法の適用が想定されるが, その場合健全性の証明が問題となってくる. 本発表では, 安全性証明の構成ツールとして定理証明支援系 Coq[1] 上でのゲーム手法の形式化を提案する. 確率的プログラミング言語を形式的に定義し, 安全性証明に必要な性質を全て Coq 上で表現・証明することで健全性が保証できる. 具体的には, 確率的プログラミング言語を定義することにより Bellare と Rogaway によるゲーム手法の枠組み [2] を形式化し, 基本補題等の各補題の証明, さらにその応用例として PRP/PRF Switching Lemma の証明を紹介する. なお, 本稿は [3] の概要である. 詳細は [3] を参照されたい.

2 確率的状態の形式化

通常のプログラミング言語の意味論では, プログラムの状態は変数とその値のペアの集合として表される. これを決定性状態と呼ぶ. 確率的プログラミング言語における確率的状態は決定性状態の確率分布として表され, 形式的には決定性状態とその確率のペアのリストとして定義される. 確率的プログラミング言語において最も重要な命令はランダムサンプリングである. 下にブール値ランダムサンプリングの場合の確率的状態のふるまいを例として示す.



また, 確率事象の生起確率も形式的に定義される. 基本的に確率事象はブール関数として定義され, 確率的状態 `d` での事象 `e` の確率は, `e` を満たす決定性状態の生起確率の和の, リスト全体に対する比率として定義され, これを `Pr e d` と書く.

3 確率的プログラミング言語

本研究で定義するゲーム手法のための言語は, ランダムサンプリングと命令型プログラミング言語である [2]. また, ハッシュ関数をランダムオラクルとして実現している. プログラムの実行は, 確率状態上の遷移に相当する. 操作的意味論は, 1. 関数環境, 2. 始状態, 3. プログラム, 4. 終状態の四項関係として定義され, `prg ||- st -- c --> st'` と表記される.

*Reynald Affeldt, 産業技術総合研究所 情報セキュリティ研究センター, National Institute of Advanced Industrial Science and Technology
†Miki Tanaka, 情報通信研究機構 情報通信セキュリティ研究センター, National Institute of Information and Communications Technology
‡Email : miki.tanaka@nict.go.jp
§Nicolas Marti, 東京大学 情報理工学系研究科, University of Tokyo

例：PRF ゲーム 以下のプログラムを考える.

```
Definition PRF' bad (A:nat→nat) i :=
 x <- int_e (A i) ;
 y <-$- n ;
 find_value z (var_e y) ;
 ifte (var_e z)
  (bad <- int_e 1) skip ;
 insert (var_e x) (var_e y).
```

一行目は x に対する代入. 二行目はサイズ n の集合からのランダムサンプリングの結果を y に代入. 三行目は, y の値が既にオラクルに登録されいるかを調べ, その結果を z に代入. 四行目では, z（三行目の結果）が真ならば bad に 1 を代入, 偽ならば何もしない. 五行目では, (x,y) をオラクルに登録している.

4 ゲームの手法の基本補題

ゲームの手法では, 安全性性質や計算量的問題はゲームとして記述され, リダクションによる証明はゲーム書換の列 $G_0 \to G_1 \to \cdots \to G_n$ として表現される. 基本補題は, ゲーム G_i と G_{i+1} 間での生起確率の差の上限を求める手法の一つである. 基本補題が適用できるのは,G_i と G_{i+1} が, "identical-until-bad"(iub) と呼ばれる, 「変数 bad に 1 が代入される時点までは全く同じ」というゲーム上の同値関係を満たす場合である.

```
Definition PRP' bad (A:nat→nat) i :=
 x <- int_e (A i) ;
 y <-$- n ;
 find_value z (var_e y) ;
 ifte (var_e z)
  (bad <- int_e 1; any) skip ;
 insert (var_e x) (var_e y).
```

any は bad への代入を含まない任意のプログラムを表す. この PRP' と先に示したゲーム PRF' は同値関係 iub を満たしている. PRF' と PRP' を拡張したゲームに関して, 下に示した補題が成立する. これは基本補題を適用することで証明できる.

```
Lemma from_PRF_to_PRP : ∀ q,
 ∀ A,(∀ x y, x ≠ y → A x ≠ A y) →
 ∀ st, coeff_pos st →
 ∀ st1, nil ||- st -- PRF bad q A --> st1 →
 ∀ st2, nil ||- st -- PRP bad q A --> st2 →
 ∀ e, Rabs (Pr e st2 - Pr e st1)
                ≤ Pr (sets bad 1) st1.
```

補題の意味するところは, 次のとおりである.PRF, PRP ゲームを実行したそれぞれの場合について, 任意の確率事象 e の生起確率の差は, いずれかゲームを実行した場合に bad の値が 1 であるという事象の生起確率で押えられる.

5 Switching Lemma

PRP/PRF Switching Lemma は, 暗号理論におけるブロック暗号に関する安全性証明の中でよく用いられる補題である. 本来, ブロック暗号は疑似乱数置換としてふるまうべきものであるが, この仮定を少し緩めて疑似乱数関数として扱うと証明が容易になる. そこで, 本補題は, 任意の確率事象についてこの仮定の変更に伴う生起確率の変化に上限を与えるものである. この補題については誤った証明が複数発表された経緯があるが [2], 本研究ではゲームの手法の形式化に基いた厳密に正確な証明を与える.

```
Lemma switching : ∀ q, q ≠ 0 →
 ∀ A, (∀ x y, x ≠ y → A x ≠ A y) →
 ∀ st, coeff_pos st → sum st > 0 →
 plength 0 st → Pr (sets bad 1) st = 0 →
 ∀ st1, nil ||- st -- PRF bad q A --> st1 →
 ∀ st2, nil ||- st -- PRP bad q A --> st2 →
 ∀ e, Rabs (Pr e st2 - Pr e st1)
       ≤ INR(q*(q-1))/INR(2*n) * sum st1.
```

証明の構成は, まず, 先に示した補題 from_PRF_to_PRP を適用した後, 得られた上限 Pr (sets bad 1) st1 を具体的に評価して INR(q*(q-1))/INR(2*n) * sum st1 という値を得る.

参考文献

[1] The LogiCal Project, INRIA. The Coq proof assistant: http://coq.inria.fr.

[2] M. Bellare and P. Rogaway. Code-Based Game-Playing Proofs and the Security of Triple Encryption. In *Advances in Cryptology (EuroCrypt '06)*, vol. 4004 of LNCS, p. 409–426. Springer.

[3] R. Affeldt, M. Tanaka, N. Marti. Formal Proof of Provable Security by Game-playing in a Proof Assistant. To appear in *Provable Security (ProvSec '07)*, LNCS, Springer.