

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

Reynald Affeldt¹ Miki Tanaka² Nicolas Marti³

¹Research Center for Information Security, National Institute of Advanced
Industrial Science and Technology

²Information Security Research Center, National Institute of Information and
Communications Technology

³Department of Computer Science, University of Tokyo

JSIAM-FAIS, 15 September 2007

Outline

- 1 Motivation and Background
 - Provable Security and Game-playing
 - Code-based Game-playing
- 2 Game-playing Framework in Coq
 - A Probabilistic Imperative Language
 - Fundamental Lemma
- 3 PRP/PRF Switching Lemma
 - Switching Lemma as Games
 - The Proof Sketch

Outline

- 1 Motivation and Background
 - Provable Security and Game-playing
 - Code-based Game-playing
- 2 Game-playing Framework in Coq
 - A Probabilistic Imperative Language
 - Fundamental Lemma
- 3 PRP/PRF Switching Lemma
 - Switching Lemma as Games
 - The Proof Sketch

Outline

- 1 Motivation and Background
 - Provable Security and Game-playing
 - Code-based Game-playing
- 2 Game-playing Framework in Coq
 - A Probabilistic Imperative Language
 - Fundamental Lemma
- 3 PRP/PRF Switching Lemma
 - Switching Lemma as Games
 - The Proof Sketch

How can you trust your cryptographic primitives?

How can you trust your cryptographic primitives?

- By non-existence of attacks (so far)

How can you trust your cryptographic primitives?

- By non-existence of attacks (so far)
- By well-demonstrated rigorous proofs

How can you trust your cryptographic primitives?

- By non-existence of attacks (so far)
- By well-demonstrated rigorous proofs

“A proof of security will increase confidence in the security of a cryptographic scheme”

How can you trust your cryptographic primitives?

- By non-existence of attacks (so far)
- By well-demonstrated rigorous proofs

“A proof of security will increase confidence in the security of a cryptographic scheme”

Provable Security Paradigm

- Proofs based on complexity theory
- Reduction to well-studied hard problems (RSA, DL etc.)
- In some form of games, asymptotic

How rigorous? How well-demonstrated?

Bellare & Rogaway, “Code-based game-playing proofs and the security of triple encryption”, Eurocrypt 2006 (full version)

How rigorous? How well-demonstrated?

Bellare & Rogaway, “Code-based game-playing proofs and the security of triple encryption”, Eurocrypt 2006 (full version)

- “In our opinion, many proofs in cryptography have become essentially unverifiable.

How rigorous? How well-demonstrated?

Bellare & Rogaway, “Code-based game-playing proofs and the security of triple encryption”, Eurocrypt 2006 (full version)

- “In our opinion, many proofs in cryptography have become essentially unverifiable. Our field may be approaching a crisis of rigor. ...

How rigorous? How well-demonstrated?

Bellare & Rogaway, “Code-based game-playing proofs and the security of triple encryption”, Eurocrypt 2006 (full version)

- “In our opinion, many proofs in cryptography have become essentially unverifiable. Our field may be approaching a crisis of rigor. ... game-playing may play a role in the answer”.

How rigorous? How well-demonstrated?

Bellare & Rogaway, “Code-based game-playing proofs and the security of triple encryption”, Eurocrypt 2006 (full version)

- “In our opinion, many proofs in cryptography have become essentially unverifiable. Our field may be approaching a crisis of rigor. ... game-playing may play a role in the answer”.
- Proofs by *Code-based Game-playing*

How rigorous? How well-demonstrated?

Bellare & Rogaway, “Code-based game-playing proofs and the security of triple encryption”, Eurocrypt 2006 (full version)

- “In our opinion, many proofs in cryptography have become essentially unverifiable. Our field may be approaching a crisis of rigor. ... game-playing may play a role in the answer”.
- Proofs by *Code-based Game-playing*
- “the game-playing approach can lead to proofs that are less error-prone and more easily verifiable, even mechanically verifiable”

Outline

- 1 Motivation and Background
 - Provable Security and Game-playing
 - Code-based Game-playing
- 2 Game-playing Framework in Coq
 - A Probabilistic Imperative Language
 - Fundamental Lemma
- 3 PRP/PRF Switching Lemma
 - Switching Lemma as Games
 - The Proof Sketch

Proofs by *Code-based* Game-playing

Game-playing Proofs are like :



- Sequence of “Games”
- Security definitions are games
- Proof steps are game transformations

Proofs by *Code-based* Game-playing

Game-playing Proofs are like :

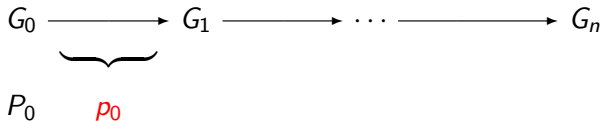


P_0

- Sequence of “Games”
- Security definitions are games
- Proof steps are game transformations

Proofs by *Code-based* Game-playing

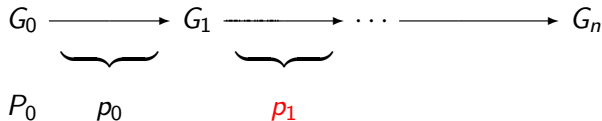
Game-playing Proofs are like :



- Sequence of “Games”
- Security definitions are games
- Proof steps are game transformations

Proofs by *Code-based* Game-playing

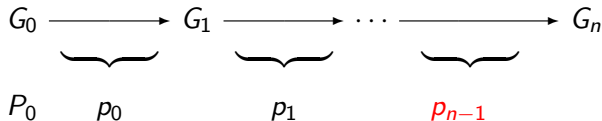
Game-playing Proofs are like :



- Sequence of “Games”
- Security definitions are games
- Proof steps are game transformations

Proofs by *Code-based* Game-playing

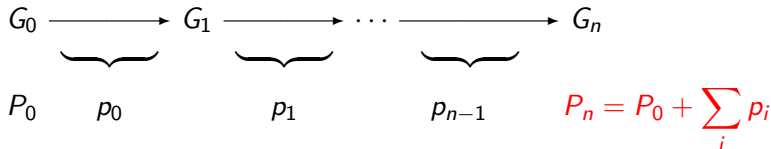
Game-playing Proofs are like :



- Sequence of “Games”
- Security definitions are games
- Proof steps are game transformations

Proofs by *Code-based* Game-playing

Game-playing Proofs are like :



- Sequence of “Games”
- Security definitions are games
- Proof steps are game transformations

Proofs by *Code-based* Game-playing

Game-playing Proofs are like :

$$\begin{array}{ccccccc}
 G_0 & \longrightarrow & G_1 & \longrightarrow & \dots & \longrightarrow & G_n \\
 & \underbrace{\hspace{2cm}} & & \underbrace{\hspace{2cm}} & & \underbrace{\hspace{2cm}} & \\
 P_0 & & p_0 & & p_1 & & p_{n-1} & & P_n = P_0 + \sum_i p_i
 \end{array}$$

- Sequence of “Games”
- Security definitions are games
- Proof steps are game transformations

What are “Games”??

Proofs by *Code-based* Game-playing

Code-based Game-playing Proofs are like :

$$\begin{array}{ccccccc}
 G_0 & \longrightarrow & G_1 & \longrightarrow & \dots & \longrightarrow & G_n \\
 & \underbrace{\hspace{2cm}} & & \underbrace{\hspace{2cm}} & & \underbrace{\hspace{2cm}} & \\
 P_0 & & p_0 & & p_1 & & p_{n-1} & & P_n = P_0 + \sum_i p_i
 \end{array}$$

- Sequence of “Games”
- Security definitions are games
- Proof steps are game transformations

What are “Games”??

Proofs by *Code-based* Game-playing

Code-based Game-playing Proofs are like :

$$\begin{array}{ccccccc}
 G_0 & \longrightarrow & G_1 & \longrightarrow & \dots & \longrightarrow & G_n \\
 & \underbrace{\hspace{2cm}} & & \underbrace{\hspace{2cm}} & & \underbrace{\hspace{2cm}} & \\
 P_0 & & p_0 & & p_1 & & p_{n-1} & & P_n = P_0 + \sum_i p_i
 \end{array}$$

- Sequence of "**probabilistic programs**"
- Security definitions are games
- Proof steps are game transformations

Proofs by *Code-based* Game-playing

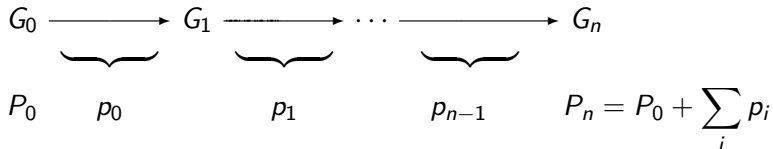
Code-based Game-playing Proofs are like :

$$\begin{array}{ccccccc}
 G_0 & \longrightarrow & G_1 & \longrightarrow & \dots & \longrightarrow & G_n \\
 & \underbrace{\hspace{2cm}} & & \underbrace{\hspace{2cm}} & & \underbrace{\hspace{2cm}} & \\
 P_0 & & p_0 & & p_1 & & p_{n-1} & & P_n = P_0 + \sum_i p_i
 \end{array}$$

- Sequence of "probabilistic programs"
- Security definitions are **probabilistic programs**
- Proof steps are game transformations

Proofs by *Code-based* Game-playing

Code-based Game-playing Proofs are like :



- Sequence of "probabilistic programs"
- Security definitions are probabilistic programs
- Proof steps are **program** transformations

Towards Automation of Game-playing

CryptoVerif, Blanchet & Pointcheval, 2006

- Games as processes
- Transformations as rewrite rules based on indistinguishability
- Some of the important rules justified by hand

Towards Automation of Game-playing

CryptoVerif, Blanchet & Pointcheval, 2006

- Games as processes
- Transformations as rewrite rules based on indistinguishability
- Some of the important rules justified by hand

Our approach

- Games as programs
- A probabilistic imperative language implemented on Coq
- All transformations justified inside Coq (fully mechanized)

Towards Automation of Game-playing

CryptoVerif, Blanchet & Pointcheval, 2006

- Games as processes
- Transformations as rewrite rules based on indistinguishability
- Some of the important rules justified by hand

Our approach

- Games as programs
- A probabilistic imperative language implemented on Coq
- All transformations justified inside Coq (fully mechanized)

An Example

PRP/PRF Switching Lemma using Fundamental Lemma

Outline

- 1 Motivation and Background
 - Provable Security and Game-playing
 - Code-based Game-playing
- 2 Game-playing Framework in Coq
 - A Probabilistic Imperative Language
 - Fundamental Lemma
- 3 PRP/PRF Switching Lemma
 - Switching Lemma as Games
 - The Proof Sketch

Outline

- 1 Motivation and Background
 - Provable Security and Game-playing
 - Code-based Game-playing
- 2 Game-playing Framework in Coq
 - A Probabilistic Imperative Language
 - Fundamental Lemma
- 3 PRP/PRF Switching Lemma
 - Switching Lemma as Games
 - The Proof Sketch

A Probabilistic Imperative Language I

Probabilistic Execution States

- Deterministic states : lists of (variable, value)
- Oracles : indexed tables of (key, value)
- Probabilistic states : Distributions of (dstate, oracle)
- Events : boolean valued functions on dstates
- Operations on distributions : sum, filter, map, scale, fork
- Probability : $\Pr e d = \text{sum}(\text{filter } e \ d)$

A Probabilistic Imperative Language I

Probabilistic Execution States

- Deterministic states : lists of (variable, value)
- Oracles : indexed tables of (key, value)
- **Probabilistic states : Distributions of (dstate, oracle)**
- Events : boolean valued functions on dstates
- Operations on distributions : sum, filter, map, scale, fork
- Probability : $\Pr e d = \text{sum}(\text{filter } e d)$

A Probabilistic Imperative Language I

Probabilistic Execution States

- Deterministic states : lists of (variable, value)
- Oracles : indexed tables of (key, value)
- Probabilistic states : Distributions of (dstate, oracle)
- Events : boolean valued functions on dstates
- Operations on distributions : sum, filter, map, scale, fork
- **Probability** : $\text{Pr } e \text{ d} = \text{sum}(\text{filter } e \text{ d})$

A Probabilistic Imperative Language I

Probabilistic Execution States

- Deterministic states : lists of (variable, value)
- Oracles : indexed tables of (key, value)
- Probabilistic states : Distributions of (dstate, oracle)
- Events : boolean valued functions on dstates
- Operations on distributions : sum, filter, map, scale, fork
- Probability : $\Pr e d = \text{sum}(\text{filter } e d)$

Boolean sampling using fork

$$((p_0, (x, 0)) :: (p_1, (x, 1))) :: \text{nil}$$

A Probabilistic Imperative Language I

Probabilistic Execution States

- Deterministic states : lists of (variable, value)
- Oracles : indexed tables of (key, value)
- Probabilistic states : Distributions of (dstate, oracle)
- Events : boolean valued functions on dstates
- Operations on distributions : sum, filter, map, scale, fork
- Probability : $\Pr e d = \text{sum}(\text{filter } e d)$

Boolean sampling using fork

$$\begin{aligned}
 & ((p_0, (x, 0)) :: (p_1, (x, 1)) :: \text{nil}) \\
 & \quad \downarrow \\
 & (pp_0, (x, 0) :: (y, 0)) :: (pp_1, (x, 1) :: (y, 0)) :: \\
 & \quad ((1-p)p_0, (x, 0) :: (y, 1)) :: ((1-p)p_1, (x, 1) :: (y, 1)) :: \text{nil}
 \end{aligned}$$

A Probabilistic Imperative Language I

Probabilistic Execution States

- Deterministic states : lists of (variable, value)
- Oracles : indexed tables of (key, value)
- Probabilistic states : Distributions of (dstate, oracle)
- Events : boolean valued functions on dstates
- Operations on distributions : sum, filter, map, scale, fork
- Probability : $\Pr e d = \text{sum}(\text{filter } e d)$

Boolean sampling using fork

$$\begin{aligned}
 & ((p_0, (x, 0)) :: (p_1, (x, 1)) :: \text{nil}) \\
 & \quad \downarrow \\
 & ((pp_0, (x, 0)) :: (y, 0)) :: ((pp_1, (x, 1)) :: (y, 0)) :: \\
 & ((1-p)p_0, (x, 0)) :: (y, 1)) :: ((1-p)p_1, (x, 1)) :: (y, 1)) :: \text{nil}
 \end{aligned}$$

A Probabilistic Imperative Language I

Probabilistic Execution States

- Deterministic states : lists of (variable, value)
- Oracles : indexed tables of (key, value)
- Probabilistic states : Distributions of (dstate, oracle)
- Events : boolean valued functions on dstates
- Operations on distributions : sum, filter, map, scale, fork
- Probability : $\Pr e d = \text{sum}(\text{filter } e d)$

Boolean sampling using fork

$$\begin{aligned}
 & ((p_0, (x, 0)) :: (p_1, (x, 1)) :: \text{nil}) \\
 & \quad \downarrow \\
 & (pp_0, (x, 0)) :: (y, 0) :: (pp_1, (x, 1)) :: (y, 0) :: \\
 & \quad ((1-p)p_0, (x, 0)) :: (y, 1) :: ((1-p)p_1, (x, 1)) :: (y, 1) :: \text{nil}
 \end{aligned}$$

A Probabilistic Imperative Language II

Commands

```
Inductive cmd : Set :=
| skip : cmd
| assign : var -> expr -> cmd
| sample_n : var -> nat -> cmd
| sample_b : var -> R -> cmd
| find_value : var -> expr -> cmd
| insert : expr -> expr -> cmd
| ifte : expr -> cmd -> cmd -> cmd
| seq : cmd -> cmd -> cmd
| call : fun_id -> cmd
```


Example of a Program

```
Definition PRF' bad A 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).
```

Example of a Program

```
Definition PRF' bad A 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).
```

- assign a value to x

Example of a Program

```

Definition PRF' bad A 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).
  
```

- assign a value to x
- random sampling out of n for y

Example of a Program

```

Definition PRF' bad A 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).
    
```

- assign a value to x
- random sampling out of n for y
- look up in the oracle if value y is already there

Example of a Program

```

Definition PRF' bad A 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).
    
```

- assign a value to x
- random sampling out of n for y
- look up in the oracle if value y is already there
- if yes then assing 1 to bad else do nothing

Example of a Program

```

Definition PRF' bad A 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).
    
```

- assign a value to x
- random sampling out of n for y
- look up in the oracle if value y is already there
- if yes then assing 1 to bad else do nothing
- add entry (x, y) to the oracle

A Probabilistic Imperative Language III

Operational Semantics in terms of probabilistic states

Defined as a relation on

- environments
- probabilistic states (start state)
- commands
- probabilistic states (end state)

written as `prg ||- start -- cmd --> end`

A Probabilistic Imperative Language III

Operational Semantics in terms of probabilistic states

Defined as a relation on

- environments
- probabilistic states (start state)
- commands
- probabilistic states (end state)

written as `prg ||- start -- cmd --> end`

A Probabilistic Imperative Language III

Operational Semantics in terms of probabilistic states

Defined as a relation on

- environments
- probabilistic states (start state)
- commands
- probabilistic states (end state)

written as `prg ||- start -- cmd --> end`

Boolean sampling `sample_b`

```
nil ||- st0 -- y <-b- p -->
      fork ((p, update y 0)::(1-p, update y 1)::nil) st
```

Outline

- 1 Motivation and Background
 - Provable Security and Game-playing
 - Code-based Game-playing
- 2 Game-playing Framework in Coq
 - A Probabilistic Imperative Language
 - Fundamental Lemma
- 3 PRP/PRF Switching Lemma
 - Switching Lemma as Games
 - The Proof Sketch

The Fundamental Lemma of Game-playing

Lemma

Let E_1 , E_2 , F_1 and F_2 be probabilistic events such that

The Fundamental Lemma of Game-playing

Lemma

Let E_1, E_2, F_1 and F_2 be probabilistic events such that

- $Pr(E_1 \wedge \bar{F}_1) = Pr(E_2 \wedge \bar{F}_2)$; and

The Fundamental Lemma of Game-playing

Lemma

Let E_1, E_2, F_1 and F_2 be probabilistic events such that

- $Pr(E_1 \wedge \bar{F}_1) = Pr(E_2 \wedge \bar{F}_2)$; and
- $Pr(F_1) = Pr(F_2) = r$ for some $0 \leq r \leq 1$.

The Fundamental Lemma of Game-playing

Lemma

Let E_1 , E_2 , F_1 and F_2 be probabilistic events such that

- $Pr(E_1 \wedge \bar{F}_1) = Pr(E_2 \wedge \bar{F}_2)$; and
- $Pr(F_1) = Pr(F_2) = r$ for some $0 \leq r \leq 1$.

Then

$$| Pr(E_1) - Pr(E_2) | \leq r$$

The Fundamental Lemma of Game-playing

Lemma

Let E_1 , E_2 , F_1 and F_2 be probabilistic events such that

- $Pr(E_1 \wedge \bar{F}_1) = Pr(E_2 \wedge \bar{F}_2)$; and
- $Pr(F_1) = Pr(F_2) = r$ for some $0 \leq r \leq 1$.

Then

$$| Pr(E_1) - Pr(E_2) | \leq r$$

The Fundamental Lemma of Game-playing

Lemma

Let E_1 , E_2 , F_1 and F_2 be probabilistic events such that

- $Pr(E_1 \wedge \bar{F}_1) = Pr(E_2 \wedge \bar{F}_2)$; and
- $Pr(F_1) = Pr(F_2) = r$ for some $0 \leq r \leq 1$.

Then

$$| Pr(E_1) - Pr(E_2) | \leq r$$

$$| Pr(E_1) - Pr(E_2) |$$

The Fundamental Lemma of Game-playing

Lemma

Let E_1, E_2, F_1 and F_2 be probabilistic events such that

- $Pr(E_1 \wedge \bar{F}_1) = Pr(E_2 \wedge \bar{F}_2)$; and
- $Pr(F_1) = Pr(F_2) = r$ for some $0 \leq r \leq 1$.

Then

$$| Pr(E_1) - Pr(E_2) | \leq r$$

$$\begin{aligned} & | Pr(E_1) - Pr(E_2) | \\ &= | (Pr(E_1 \wedge F_1) + Pr(E_1 \wedge \bar{F}_1)) - (Pr(E_2 \wedge F_2) + Pr(E_2 \wedge \bar{F}_2)) | \end{aligned}$$

The Fundamental Lemma of Game-playing

Lemma

Let E_1 , E_2 , F_1 and F_2 be probabilistic events such that

- $Pr(E_1 \wedge \bar{F}_1) = Pr(E_2 \wedge \bar{F}_2)$; and
- $Pr(F_1) = Pr(F_2) = r$ for some $0 \leq r \leq 1$.

Then

$$| Pr(E_1) - Pr(E_2) | \leq r$$

$$\begin{aligned} & | Pr(E_1) - Pr(E_2) | \\ &= | (Pr(E_1 \wedge F_1) + Pr(E_1 \wedge \bar{F}_1)) - (Pr(E_2 \wedge F_2) + Pr(E_2 \wedge \bar{F}_2)) | \\ &= | Pr(E_1 \wedge F_1) - Pr(E_2 \wedge F_2) | \end{aligned}$$

The Fundamental Lemma of Game-playing

Lemma

Let E_1, E_2, F_1 and F_2 be probabilistic events such that

- $Pr(E_1 \wedge \bar{F}_1) = Pr(E_2 \wedge \bar{F}_2)$; and
- $Pr(F_1) = Pr(F_2) = r$ for some $0 \leq r \leq 1$.

Then

$$| Pr(E_1) - Pr(E_2) | \leq r$$

$$\begin{aligned} & | Pr(E_1) - Pr(E_2) | \\ &= | (Pr(E_1 \wedge F_1) + Pr(E_1 \wedge \bar{F}_1)) - (Pr(E_2 \wedge F_2) + Pr(E_2 \wedge \bar{F}_2)) | \\ &= | Pr(E_1 \wedge F_1) - Pr(E_2 \wedge F_2) | \\ &\leq r \end{aligned}$$

Identical-until-bad Games

Two games are “**identical-until-bad**” if they are syntactically the same except the branches after `bad` \leftarrow `int_e 1`

Identical-until-bad Games

Two games are “**identical-until-bad**” if they are syntactically the same except the branches after `bad <- int_e 1`

```
g1
a;
if b then
  (bad <- int_e 1; c)
else d
```

Identical-until-bad Games

Two games are “**identical-until-bad**” if they are syntactically the same except the branches after `bad` \leftarrow `int_e 1`

g1

```
a;  
if b then  
  (bad  $\leftarrow$  int_e 1; c)  
else d
```

g2

```
a;  
if b then  
  (bad  $\leftarrow$  int_e 1; c')  
else d
```

Identical-until-bad Games

Two games are **“identical-until-bad”** if they are syntactically the same except the branches after `bad` \leftarrow `int_e 1`

g1

```
a;
if b then
  (bad  $\leftarrow$  int_e 1; c)
else d
```

g2

```
a;
if b then
  (bad  $\leftarrow$  int_e 1; c')
else d
```

Corollary

Let E_1, E_2 be probabilistic events. Then

$$| Pr_{g1}(E_1) - Pr_{g2}(E_2) | \leq Pr_{g1}(\text{“bad is set 1”})$$

Outline

- 1 Motivation and Background
 - Provable Security and Game-playing
 - Code-based Game-playing
- 2 Game-playing Framework in Coq
 - A Probabilistic Imperative Language
 - Fundamental Lemma
- 3 PRP/PRF Switching Lemma
 - Switching Lemma as Games
 - The Proof Sketch

PRP/PRF Switching Lemma

PRP/PRF Switching Lemma

- $Func(n) : \{0, 1\}^n \rightarrow \{0, 1\}^n$, the set of functions

PRP/PRF Switching Lemma

- $Func(n) : \{0, 1\}^n \rightarrow \{0, 1\}^n$, the set of functions
- $Perm(n) : \{0, 1\}^n \rightarrow \{0, 1\}^n$, the set of permutations

PRP/PRF Switching Lemma

- $Func(n) : \{0, 1\}^n \rightarrow \{0, 1\}^n$, the set of functions
- $Perm(n) : \{0, 1\}^n \rightarrow \{0, 1\}^n$, the set of permutations
- $\rho \leftarrow \$- Func(n)$, $\sigma \leftarrow \$- Perm(n)$, where ρ and σ are randomly chosen

PRP/PRF Switching Lemma

- $Func(n) : \{0, 1\}^n \rightarrow \{0, 1\}^n$, the set of functions
- $Perm(n) : \{0, 1\}^n \rightarrow \{0, 1\}^n$, the set of permutations
- $\rho \leftarrow \$- Func(n)$, $\sigma \leftarrow \$- Perm(n)$, where ρ and σ are randomly chosen
- \mathcal{A} , an adversary with an oracle access to ρ or σ

PRP/PRF Switching Lemma

- $Func(n) : \{0, 1\}^n \rightarrow \{0, 1\}^n$, the set of functions
- $Perm(n) : \{0, 1\}^n \rightarrow \{0, 1\}^n$, the set of permutations
- $\rho \leftarrow \$- Func(n)$, $\sigma \leftarrow \$- Perm(n)$, where ρ and σ are randomly chosen
- \mathcal{A} , an adversary with an oracle access to ρ or σ

Lemma

If the adversary makes at most q queries, then

$$| Pr_{\rho}(\mathcal{A} = 1) - Pr_{\sigma}(\mathcal{A} = 1) | \leq \frac{q(q-1)}{2^{n+1}}$$

Outline

- 1 Motivation and Background
 - Provable Security and Game-playing
 - Code-based Game-playing
- 2 Game-playing Framework in Coq
 - A Probabilistic Imperative Language
 - Fundamental Lemma
- 3 PRP/PRF Switching Lemma
 - Switching Lemma as Games
 - The Proof Sketch

Switching Lemma : Game0 (PRP Game)

A game that simulates a random permutation :

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

If the randomly selected value is already in the oracle, **bad** is set. Then execute **retry** to select a fresh value.

Game0

```
Definition PRP bad q A := loop q (PRP' bad A)
```


Switching Lemma : Game1 (PRF Game)

A game that simulates a random function :

```

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

No action after setting
bad.

Behaves as a function, not
 permutation (Collisions
 are allowed).

Game1

```

Definition PRF bad q A := loop q (PRF' bad A)
    
```

Outline

- 1 Motivation and Background
 - Provable Security and Game-playing
 - Code-based Game-playing
- 2 Game-playing Framework in Coq
 - A Probabilistic Imperative Language
 - Fundamental Lemma
- 3 PRP/PRF Switching Lemma
 - Switching Lemma as Games
 - The Proof Sketch

Sketch of the Proof : Step 1

Switching Lemma

If the adversary makes at most q queries, then

$$| Pr_{Game0}(\mathcal{A} = 1) - Pr_{Game1}(\mathcal{A} = 1) | \leq \frac{q(q-1)}{2^{n+1}}$$

Sketch of the Proof : Step 1

Step 1 : Application of a version of Fundamental Lemma to Game0 and Game1

Corollary

$$| Pr_{Game0}(\mathcal{A} = 1) - Pr_{Game1}(\mathcal{A} = 1) | \leq Pr_{Game0}(\text{"bad is set 1"})$$

Sketch of the Proof : Step 2

Step 2 : Evaluate the bound Pr_{Game0} (“bad is set 1”) by induction on q

Sketch of the Proof : Step 2

Step 2 : Evaluate the bound Pr_{Game0} (“bad is set 1”) by induction on q

Lemma

$$Pr_{Game0}(\text{“bad is set 1”}) \leq \frac{q(q-1)}{2^{n+1}}$$

Sketch of the Proof : Step 2

Step 2 : Evaluate the bound Pr_{Game0} (“bad is set 1”) by induction on q

Lemma

$$Pr_{Game0}(\text{“bad is set 1”}) \leq \frac{q(q-1)}{2^{n+1}}$$

Switching Lemma

If the adversary makes at most q queries, then

$$| Pr_{Game0}(\mathcal{A} = 1) - Pr_{Game1}(\mathcal{A} = 1) | \leq \frac{q(q-1)}{2^{n+1}}$$

Summary

- A **probabilistic imperative language** for Game-playing proofs

Summary

- A **probabilistic imperative language** for Game-playing proofs
- A **fully mechanized proof** of PRP/PRF switching lemma

Summary

- A **probabilistic imperative language** for Game-playing proofs
- A **fully mechanized proof** of PRP/PRF switching lemma
- Application to many more interesting examples to come

Summary

- A **probabilistic imperative language** for Game-playing proofs
- A **fully mechanized proof** of PRP/PRF switching lemma
- Application to many more interesting examples to come

Reference



R. Affeldt, M. Tanaka, N. Marti.

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

ProvSec 2007, LNCS 4784, pages 151–168.