On Higher-Order Cryptography

Boaz Barak **Raphaëlle Crubillé** **Ugo Dal Lago**

informatiques mathématiques Inria

ICALP 2020, Track B

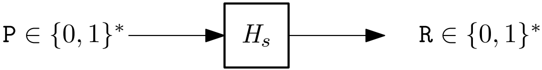
How to Treat Programs in a Cryptographic Scenario?

- Programs, e.g. when hashed, are usually treated as strings:

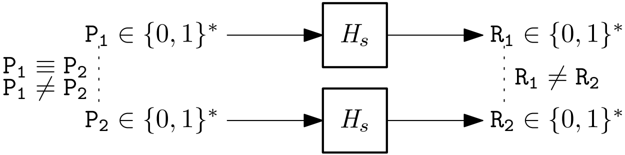
$$\mathtt{P} \in \{0,1\}^* \longrightarrow \boxed{H_s} \longrightarrow \mathtt{R} \in \{0,1\}^*$$

How to Treat Programs in a Cryptographic Scenario?

- Programs, e.g. when hashed, are usually treated as strings:

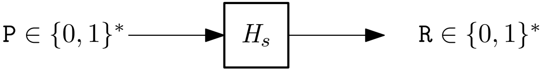


- If two programs $\mathtt{P}_1$ and $\mathtt{P}_2$ are perfectly equivalent but distinct, they are thus seen as distinct strings, and mapped to distinct hashes:

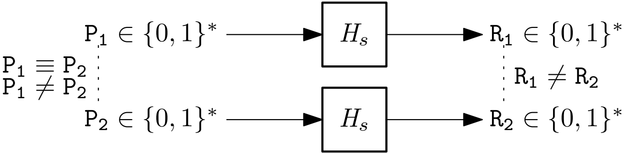


How to Treat Programs in a Cryptographic Scenario?

- Programs, e.g. when hashed, are usually treated as strings:



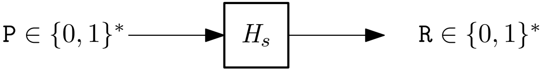
- If two programs $\mathtt{P}_1$ and $\mathtt{P}_2$ are perfectly equivalent but distinct, they are thus seen as distinct strings, and mapped to distinct hashes:



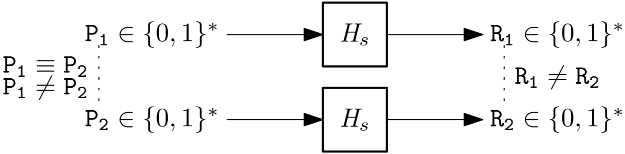
- The same argument holds when H_s is replaced by Enc_k (i.e. encryption) or Mac_k (i.e. authentication).

How to Treat Programs in a Cryptographic Scenario?

- Programs, e.g. when hashed, are usually treated as strings:



- If two programs $\mathtt{P}_1$ and $\mathtt{P}_2$ are perfectly equivalent but distinct, they are thus seen as distinct strings, and mapped to distinct hashes:



- The same argument holds when H_s is replaced by Enc_k (i.e. encryption) or Mac_k (i.e. authentication).
- Would it be possible to define any cryptographic primitive in such a way as to make it *equivalence preserving*?
  - That somehow amounts to turning H_s into a program of type $(\{0,1\}^* \to \{0,1\}^*) \to \{0,1\}^*$ (rather than $\{0,1\}^* \to \{0,1\}^*$).

Contributions in This Talk

1. A **New Model** of Complexity-Bounded Higher-Order Computation Based on *Game Semantics*.
    - Second-order adversaries are everywhere in cryptography.
    - Defining the concept of an *efficient adversary* at third-order (or above!) instead requires some care.
    - Game semantics [AJM00,HO00] offers a way to reduce higher-order computation to first-order computation.

Contributions in This Talk

1. A **New Model** of Complexity-Bounded Higher-Order Computation Based on *Game Semantics*.
    - Second-order adversaries are everywhere in cryptography.
    - Defining the concept of an *efficient adversary* at third-order (or above!) instead requires some care.
    - Game semantics [AJM00,HO00] offers a way to reduce higher-order computation to first-order computation.
2. Some *Negative* and *Positive* Results on the Feasibility of **Higher-Order Cryptography**.
    - Results about influential variables in decision trees imply that second-order pseudorandomness and collision-resistance are not attainable.
    - Some positive results can be obtained, but there is an high price to pay.

Pseudorandomness

A family of distributions $\{\mathcal{D}_n\}_{n\in\mathbb{N}}$, each having "type" X_n, is said to be **pseudorandom** iff $\mathcal{D}_n$ is indistinguishable from a genuinely uniform random distribution of the same type, by distinguishers working in polynomial time (in n).

Definition

A scheme

$$S = (S_n : \{0,1\}^n \longrightarrow X_n)_{n\in\mathbb{N}} \qquad \text{is pseudo-random}$$

key — deterministic function

when for every **efficient**, randomized distinguisher $\mathcal{A} = (\mathcal{A}_n :!X_n \to \{0,1\})_{n\in\mathbb{N}}$,

$$\left| \operatorname*{Prob}_{k_1,k_2,\ldots\leftarrow \mathrm{Unif}(\{0,1\}^n)} [\mathcal{A}_n(S_n(k_1), S_n(k_2), \ldots)) \downarrow 1] - \operatorname*{Prob}_{x_1,x_2,\ldots\leftarrow \mathrm{Unif}(X_n)} [\mathcal{A}_n(x_1, x_2, \ldots) \downarrow 1] \right| \leq \epsilon(n)$$

negligeable function

Pseudorandomness in Cryptography

$$S = (S_n : \{0,1\}^n \longrightarrow X_n)_{n \in \mathbb{N}}$$

Order 0: $X_n = \{0,1\}^{r(n)}$. Pseudo-Random Number Generator (PRNG)

- take a few random bits and produce a longer string of pseudo-random bits.
- used e.g for key-generation, encryption...

Order 1: $X_n = \{0,1\}^{r(n)} \to \{0,1\}^{l(n)}$. Pseudo-Random Function (PRF)

- from a random key k, build deterministically a function that associates to any message m a tag t, indistinguishable from a random mapping from messages to tags.
- used e.g as MAC (message authentication code)

Existence: widely accepted

- PRNG exist iff **one-way functions** exists;
- PRNG exist $\Rightarrow P \neq NP$;
- PRNG exists $\Rightarrow$ PRF exist.

Collision Resistance

Definition

A scheme

$$S = (S_n : \{0,1\}^n \longrightarrow (Y_n \to Z_n))_{n\in\mathbb{N}} \qquad \text{is collision-resistant}$$

key — deterministic functions

when for every **efficient**, randomized adversary $\mathcal{A} = (\mathcal{A}_n : (Y_n \to Z_n) \to Y_n \times Y_n)_{n\in\mathbb{N}}$,

$$\underset{k \leftarrow \mathtt{Unif}(\{0,1\}^n)}{\text{Prob}} [\mathcal{A}_n(S_n(k)) = (y_1, y_2) \wedge y_1 \neq y_2 \wedge S_n(k)(y_1) = S_n(k)(y_2)] \leq \epsilon(n)$$

negligeable function

Fact

As soon as $(n \mapsto \frac{card(Y_n)}{2^{2\cdot card(Z_n)}})$ *is negligeable, a truly random* S *is collision resistant*
e.g. $Y_n = (\{0,1\}^{p(n)} \to \{0,1\})$, $Z_n = \{0,1\}^{q(n)}$ *is collision-resistant when* $p(n) \leq q(n)$.

Higher-Order Pseudorandomness?

Intuitively, it is **impossible** to build deterministic polytime objects of type

$$S : \underbrace{\{0,1\}^n}_{\text{key}} \to \overbrace{\big(\underbrace{(\{0,1\}^n \to \{0,1\}^1)}_{\text{input function}} \to \underbrace{\{0,1\}^n}_{\text{tag}}\big)}^{X_n}$$

which "look random".

Intuition:

- the input function can be accessed only polynomially many times by the efficient algorithm S;
- a truly random F in $(\{0,1\}^n \to \{0,1\}^1 \to \{0,1\}^n$ would a priori depends on exponentially many answers of the input function.

Question:

How to turn this into a formal argument?

Higher-Order **Randomized**, **Efficient** Adversaries ?

Efficients Distinguishers

- If $X_n = \{0,1\}^{r(n)}$, then the distinguisher is of order 1, i.e. just a polytime randomized algorithm.
- If $X_n = \{0,1\}^n \to \{0,1\}^n$, then the distinguisher is of order 2: can be taken as a polytime (in n) oracle randomized Turing machine.
- If $X_n = (\{0,1\}^n \to \{0,1\}^1) \to \{0,1\}^n$, then the distinguisher is of **order 3**.

Fact

Third-order adversaries have not been considered, at least so far, by the crypto community.

Question:

How should we account for the time it takes to "cook" an argument function?

Are we Looking at a Form of Higher-Order Complexity?

Yes!...

- **No** *modern* cryptographic construction is secure against **unbounded adversaries**, so limiting the computational capabilities of the adversary is necessary.
- Adversaries could be third-order.
- Efficiency should be captured by **polynomial time** computability (in the value of the security parameter).

Are we Looking at a Form of Higher-Order Complexity?

Yes!...

- **No** *modern* cryptographic construction is secure against **unbounded adversaries**, so limiting the computational capabilities of the adversary is necessary.
- Adversaries could be third-order.
- Efficiency should be captured by **polynomial time** computability (in the value of the security parameter).

...but Not Really!

- There is no aim at **classifying functions** as for their inherent difficulty following, e.g., the work by Cook et al. [CU1988,CK1992].
- The *size* of the input function is not a crucial parameter.

An Example: the Game Semantics of a Second-Order Function

$$!(\{0,1\}^* \quad \multimap \quad \{0,1\}^*) \quad \multimap \quad \{0,1\}^*$$

An Example: the Game Semantics of a Second-Order Function

$$!(\{0,1\}^* \quad \multimap \quad \{0,1\}^*) \quad \multimap \quad \{0,1\}^*$$

O $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad ?$

An Example: the Game Semantics of a Second-Order Function

	$!(\{0,1\}^*$	$\multimap$	$\{0,1\}^*)$	$\multimap$	$\{0,1\}^*$
O					$?$
P			$(1, ?)$		

An Example: the Game Semantics of a Second-Order Function

	$!(\{0,1\}^*$	$\multimap$	$\{0,1\}^*)$	$\multimap$	$\{0,1\}^*$
O					$?$
P			$(1, ?)$		
O	$(1, ?)$				
P	$(1, s_1)$				

An Example: the Game Semantics of a Second-Order Function

	$!(\{0,1\}^*$	$\multimap$	$\{0,1\}^*)$	$\multimap$	$\{0,1\}^*$
O					$?$
P			$(1, ?)$		
O	$(1, ?)$				
P	$(1, s_1)$				
O			$(1, t_1)$		

An Example: the Game Semantics of a Second-Order Function

	$!(\{0,1\}^*$	$\multimap$	$\{0,1\}^*)$	$\multimap$	$\{0,1\}^*$
O					$?$
P			$(1,?)$		
O	$(1,?)$				
P	$(1,s_1)$				
O			$(1,t_1)$		
		$\vdots$			
P			$(m,?)$		
O	$(m,?)$				
P	(m,s_m)				
O			(m,t_m)		

An Example: the Game Semantics of a Second-Order Function

	$!(\{0,1\}^*$	$\multimap$	$\{0,1\}^*)$	$\multimap$	$\{0,1\}^*$
O					$?$
P			$(1, ?)$		
O	$(1, ?)$				
P	$(1, s_1)$				
O			$(1, t_1)$		
		$\vdots$			
P			$(m, ?)$		
O	$(m, ?)$				
P	(m, s_m)				
O			(m, t_m)		
P					v

An Example: the Game Semantics of a Second-Order Function

	$!(\{0,1\}^*$	$\multimap$	$\{0,1\}^*)$	$\multimap$	$\{0,1\}^*$
O					$?$
P			$(1, ?)$		
O	$(1, ?)$				
P	$(1, s_1)$				
O			$(1, t_1)$		
		$\vdots$			
P			$(m, ?)$		
O	$(m, ?)$				
P	(m, s_m)				
O			(m, t_m)		
P					v

Missing Ingredients to Model Cryptographic Primitives

- The Player determines the next move without any *complexity* constraint.
- The *length* of the interaction is in principle arbitrary (and can even be infinite).
- Strategies are deterministic, and do not have access to any source of *randomness*.

Cryptographic Game Semantics - I

Games Parametrized by a Security Parameter

- Games: $G = (O_G, P_G, (L_G^n)_{n \in \mathbb{N}})$
- Strategies: $f : \mathbb{N} \times (L_G^n \cap \mathrm{Odd}) \to P_G$

Cryptographic Game Semantics - I

Games Parametrized by a Security Parameter

- Games: $G = (O_G, P_G, (L_G^n)_{n \in \mathbb{N}})$
- Strategies: $f : \mathbb{N} \times (L_G^n \cap \mathrm{Odd}) \to P_G$

Example: Strings of Length $\leq p(n)$

$\mathbf{S}[p] = (\{?\}, \{0,1\}^\star, (L_n^{\mathbf{S}[p]})_{n \in \mathbb{N}})$ with
$L_n^{\mathbf{S}[p]} = \{\epsilon, ?\} \cup \{?s \mid |s| \leq p(n)\}$

Cryptographic Game Semantics - I

Games Parametrized by a Security Parameter

- Games: $G = (O_G, P_G, (L_G^n)_{n\in\mathbb{N}})$
- Strategies: $f : \mathbb{N} \times (L_G^n \cap \mathrm{Odd}) \to P_G$

Example: Strings of Length $\leq p(n)$

$\mathbf{S}[p] = (\{?\}, \{0,1\}^\star, (L_n^{\mathbf{S}[p]})_{n\in\mathbb{N}})$ with $L_n^{\mathbf{S}[p]} = \{\epsilon, ?\} \cup \{?s \mid |s| \leq p(n)\}$

Restricted Classes of Games and Strategies

Polynomially Bounded Games:
G such that there exists a polynomial P with positive coefficients, such that: $\forall n \in \mathbb{N}, \forall s \in L_G^n,\ |s| \leq P(n)$.

Polytime Computable Strategies:
There exists a polynomial time Turing machine which on input $(1^n, s)$ returns $f(n, s)$.

Cryptographic Game Semantics - I

Games Parametrized by a Security Parameter

- Games: $G = (O_G, P_G, (L_G^n)_{n\in\mathbb{N}})$
- Strategies: $f : \mathbb{N} \times (L_G^n \cap \mathrm{Odd}) \to P_G$

Example: Strings of Length $\leq p(n)$

$\mathbf{S}[p] = (\{?\}, \{0,1\}^\star, (L_n^{\mathbf{S}[p]})_{n\in\mathbb{N}})$ with
$L_n^{\mathbf{S}[p]} = \{\epsilon, ?\} \cup \{?s \mid |s| \leq p(n)\}$

Restricted Classes of Games and Strategies

Polynomially Bounded Games:
G such that there exists a polynomial P with positive coefficients, such that: $\forall n \in \mathbb{N}, \forall s \in L_G^n,\ |s| \leq P(n)$.

Polytime Computable Strategies:
There exists a polynomial time Turing machine which on input $(1^n, s)$ returns $f(n, s)$.

Constructing Games

From the games G, H, we can construct more complex games such as:

- $G \multimap H$, modeling functions from G to H;
- $G \otimes H$, modeling pairs of elements from G and H;
- $!_q G$ modeling $q(n)$ copies of G.

Cryptographic Game Semantics — II

Proposition (Composing Strategies)

If f, g polytime strategies on $G \multimap H$ and $H \multimap K$ (respectively), one can form $g \circ f$ as a strategy on $G \multimap K$. Moreover, strategy composition is associative.

Cryptographic Game Semantics — II

Proposition (Composing Strategies)

If f, g polytime strategies on $G \multimap H$ and $H \multimap K$ (respectively), one can form $g \circ f$ as a strategy on $G \multimap K$. Moreover, strategy composition is associative.

- How about randomization?

Cryptographic Game Semantics — II

Proposition (Composing Strategies)

If f, g polytime strategies on $G \multimap H$ and $H \multimap K$ (respectively), one can form $g \circ f$ as a strategy on $G \multimap K$. Moreover, strategy composition is associative.

- How about randomization?

Randomized Strategies—A First Try

- Games: $G = (O_G, P_G, (L_G^n)_{n \in \mathbb{N}})$
- Randomized Strategies: polytime computable functions $f : \mathbb{N} \times (L_G^n \cap \mathrm{Odd}) \to \mathsf{DISTR}(P_G)$

Cryptographic Game Semantics — II

Proposition (Composing Strategies)

If f, g polytime strategies on $G \multimap H$ and $H \multimap K$ (respectively), one can form $g \circ f$ as a strategy on $G \multimap K$. Moreover, strategy composition is associative.

- How about randomization?

Randomized Strategies—A First Try

- Games: $G = (O_G, P_G, (L_G^n)_{n \in \mathbb{N}})$
- Randomized Strategies: polytime computable functions $f : \mathbb{N} \times (L_G^n \cap \mathrm{Odd}) \to \mathsf{DISTR}(P_G)$

NO!

Randomized *polytime* strategies *are not* stable by composition.

Cryptographic Game Semantics — II

Proposition (Composing Strategies)

If f, g polytime strategies on $G \multimap H$ and $H \multimap K$ (respectively), one can form $g \circ f$ as a strategy on $G \multimap K$. Moreover, strategy composition is associative.

- How about randomization?

Randomized Strategies—A First Try

- Games: $G = (O_G, P_G, (L_G^n)_{n \in \mathbb{N}})$
- Randomized Strategies: polytime computable functions $f : \mathbb{N} \times (L_G^n \cap \mathrm{Odd}) \to \mathsf{DISTR}(P_G)$

NO!

Randomized *polytime* strategies *are not* stable by composition.

Randomized Games—Second Try

- Games: $G = (O_G, P_G, (L_G^n)_{n \in \mathbb{N}})$
- Randomized Strategies on G are taken as *deterministic* strategies on $!_p\mathbf{B} \multimap G$ (where $\mathbf{B}$ is the boolean game).

Cryptographic Game Semantics — II

Proposition (Composing Strategies)

If f, g polytime strategies on $G \multimap H$ and $H \multimap K$ (respectively), one can form $g \circ f$ as a strategy on $G \multimap K$. Moreover, strategy composition is associative.

- How about randomization?

Randomized Strategies—A First Try

- Games: $G = (O_G, P_G, (L_G^n)_{n\in\mathbb{N}})$
- Randomized Strategies: polytime computable functions $f : \mathbb{N} \times (L_G^n \cap \mathrm{Odd}) \to \mathsf{DISTR}(P_G)$

NO!

Randomized *polytime* strategies *are not* stable by composition.

Randomized Games—Second Try

- Games: $G = (O_G, P_G, (L_G^n)_{n\in\mathbb{N}})$
- Randomized Strategies on G are taken as *deterministic* strategies on $!_p\mathbf{B} \multimap G$ (where $\mathbf{B}$ is the boolean game).

YES!

The whole sequence of probabilistic choices is available, and strategies compose.

Randomized Strategies—A First Try

- Games: $G = (O_G, P_G, (L_G^n)_{n \in \mathbb{N}})$
- Randomized Strategies: polytime computable functions
  $f : \mathbb{N} \times (L_G^n \cap \mathrm{Odd}) \to \mathsf{DISTR}(P_G)$

Fact:

Polytime randomized strategies **do not** compose.

	$\mathbf{S}[n]$	$\multimap$	$!^2\mathbf{S}[n]$
O			$(1, ?_{\mathbf{S}[n]})$
P	$?_{\mathbf{S}[n]}$		
O	s		
P			$(1, h(s))$
O			$(2, ?_{\mathbf{S}[n]})$
P			$(2, s)$

f_h: **polytime** strategy
(h: one-way permutation)

$\mathbf{S}[n]$

O $\quad ?_{\mathbf{S}[n]}$

(each with probability $\frac{1}{2^n}$)

P $\quad l_1 \quad l_2 \quad \cdots \quad l_{2^n}$

Unif: **polytime** prob. strategy

$!^2\mathbf{S}[n]$

O $\quad (1, ?_{\mathbf{S}[n]})$

(each with probability $\frac{1}{2^n}$)

P $\quad (1, h(l_1)) \quad (1, h(l_2)) \cdots (1, h(l_n))$

O $\quad (2, ?_{\mathbf{S}[n]}) \quad (2, ?_{\mathbf{S}[n]}) \quad (2, ?_{\mathbf{S}[n]})$

P $\quad (2, l_1) \quad (2, l_2) \quad (2, l_{2^n})$

$f_h \circ$ Unif: **not-polytime** prob. strategy

The Category of Parametrized games and probabilistic strategies

Definition

- Objects: polynomially bounded games $G, H, \ldots$
- Morphisms from G to H: pairs (q, f), where f is a strategy in $!_q\mathbf{B} \multimap (G \multimap H)$.
- Composition: for $(q_1, f_1) : G \multimap H$; $(q_2, f_2) : H \multimap J$,

$$(q_2, f_2) \circ (q_1, f_1) : (q_2 + q_1, !_{q_2+q_1}\mathbf{B} \rightarrow !_{q_2}\mathbf{B} \otimes !_{q_1}\mathbf{B} \xrightarrow{id_{!_{q_2}\mathbf{B}} \otimes f_1} !_{q_2}\mathbf{B} \otimes G \xrightarrow{f_2} H)$$

Proposition

This category is:

- *symmetric monoidal closed, forms an exponential bounded situation.*
- *polytime computable morphisms are stable by composition.*

Observing the probabilistic behavior of a strategy

$$\mathrm{Prob}_f^n(b) = \sum_{\substack{(b_1,\ldots,b_k) \in \mathbf{B}^k \\ \text{with } (?_{\mathbf{B}} \cdot ?_{!_p\mathbf{B}} \cdot b_1 \ldots ?_{!_p\mathbf{B}} \cdot b_k \cdot b) \in \overline{f}_n}} \frac{1}{2^k} \qquad \text{for } f : !_p\mathbf{B} \multimap \mathbf{B}, b \in \mathbf{B}.$$

An Example: a Simple Randomized Strategy

$$!_1\mathbf{B} \quad\multimap\quad !_2(\mathbf{S}[n] \quad\multimap\quad \mathbf{B}) \quad\multimap\quad \mathbf{B}$$

An Example: a Simple Randomized Strategy

	$!_1\mathbf{B}$	$\multimap$	$!_2(\mathbf{S}[n]$	$\multimap$	$\mathbf{B})$	$\multimap$	$\mathbf{B}$
O							$?$
P	$(1, ?)$						
O	$(1, b)$						

An Example: a Simple Randomized Strategy

	$!_1\mathbf{B}$	$\multimap$	$!_2(\mathbf{S}[n]$	$\multimap$	$\mathbf{B})$	$\multimap$	$\mathbf{B}$
O							$?$
P	$(1, ?)$						
O	$(1, b)$						
P					$(1, ?)$		
O			$(1, ?)$				
P			$(1, b^n)$				
O					$(1, c)$		

An Example: a Simple Randomized Strategy

	$!_1\mathbf{B}$	$\multimap$	$!_2(\mathbf{S}[n]$	$\multimap$	$\mathbf{B})$	$\multimap$	$\mathbf{B}$
O							$?$
P	$(1, ?)$						
O	$(1, b)$						
P					$(1, ?)$		
O			$(1, ?)$				
P			$(1, b^n)$				
O					$(1, c)$		
P					$(2, ?)$		
O			$(2, ?)$				
P			$(2, (\neg b)^n)$				
O					$(2, d)$		

An Example: a Simple Randomized Strategy

	$!_1\mathbf{B}$	$\multimap$	$!_2(\mathbf{S}[n]$	$\multimap$	$\mathbf{B})$	$\multimap$	$\mathbf{B}$
O							$?$
P	$(1, ?)$						
O	$(1, b)$						
P					$(1, ?)$		
O			$(1, ?)$				
P			$(1, b^n)$				
O					$(1, c)$		
P					$(2, ?)$		
O			$(2, ?)$				
P			$(2, (\neg b)^n)$				
O					$(2, d)$		
P							$\neg c \wedge d$

Second-Order Pseudorandomness, Formally

- We are now in a position to finally **define** what second-order pseudorandomness could look like.

Second-Order Pseudorandomness, Formally

- We are now in a position to finally **define** what second-order pseudorandomness could look like.
- The *type* of a (candidate) **pseudorandom function** could be

$$SOF = \mathbf{S}[n] \multimap !_p(\mathbf{S}[q] \multimap \mathbf{B}) \multimap \mathbf{S}[r],$$

while the type of an **adversary** for it, being randomized, should be

$$ADV = !_s\mathbf{B} \multimap !_t(!_p(\mathbf{S}[q] \multimap \mathbf{B}) \multimap \mathbf{S}[r]) \multimap \mathbf{B}$$

Second-Order Pseudorandomness, Formally

- We are now in a position to finally **define** what second-order pseudorandomness could look like.
- The *type* of a (candidate) **pseudorandom function** could be

$$SOF = \mathbf{S}[n] \multimap !_p(\mathbf{S}[q] \multimap \mathbf{B}) \multimap \mathbf{S}[r],$$

while the type of an **adversary** for it, being randomized, should be

$$ADV = !_s\mathbf{B} \multimap !_t(!_p(\mathbf{S}[q] \multimap \mathbf{B}) \multimap \mathbf{S}[r]) \multimap \mathbf{B}$$

We say that a polytime strategy f for the game SOF is **pseudorandom** iff for any polytime strategy $\mathcal{A}$ for the game ADV it holds that

$$|Pr[\mathcal{A} \circ (f \circ rand_{\mathbf{S}[n]}) \Downarrow 1] - Pr[\mathcal{A} \circ (rand_{!_p(\mathbf{S}[q] \multimap \mathbf{B}) \multimap \mathbf{S}[r]}) \Downarrow 1]| \leq \varepsilon(n)$$

where ε is a negligible function and $rand_G$ is a random strategy for the game G.

The Negative Result: Summary

- Consider a strategy f for $SOF = \mathbf{S}[n] \multimap !_p(\mathbf{S}[q] \multimap \mathbf{B}) \multimap \mathbf{S}[r]$, where $q(n) \geq n$, and p is a polynomial. The intuition is that f *is far from being* pseudorandom, whatever this means.

The Negative Result: Summary

- Consider a strategy f for $SOF = \mathbf{S}[n] \multimap !_p(\mathbf{S}[q] \multimap \mathbf{B}) \multimap \mathbf{S}[r]$, where $q(n) \geq n$, and p is a polynomial. The intuition is that f *is far from being* pseudorandom, whatever this means.
  - The value of f depends on the value of its argument function on *polynomially* many coordinates $s_1, \ldots, s_m$, where $m \leq p(n)$
  - Once these are fixed, the value of the argument function on the other (exponentially many!) coordinates is irrelevant.
  - But **beware**: the values of $s_1, \ldots, s_m$ possibly depend on the key, and could be determined adaptively.

The Negative Result: Summary

- Consider a strategy f for $SOF = \mathbf{S}[n] \multimap !_p(\mathbf{S}[q] \multimap \mathbf{B}) \multimap \mathbf{S}[r]$, where $q(n) \geq n$, and p is a polynomial. The intuition is that f *is far from being* pseudorandom, whatever this means.
  - The value of f depends on the value of its argument function on *polynomially* many coordinates $s_1, \ldots, s_m$, where $m \leq p(n)$
  - Once these are fixed, the value of the argument function on the other (exponentially many!) coordinates is irrelevant.
  - But **beware**: the values of $s_1, \ldots, s_m$ possibly depend on the key, and could be determined adaptively.
- How could an adversary **determine** the coordinates $s_1, \ldots, s_m$?
  - Since f can be seen as a decision tree with a relatively small depth (i.e., $p(n)$), We know [ODSSS2005] that f has influential variables.
  - We can thus proceed by querying f on randomly constructed block functions, evaluating their influences, until we find one with an high-influence.
  - This way, we iteratively fix $s_1, \ldots, s_m$ in such as way that the variance of f on any function behaving according to them is very low.

Looking for Collisions with Influential Variables-I

Tool: known result on influential variables

Suppose that $F : \mathbf{S}[N] \to \mathbf{B}$ is computable by a decision tree of depth at most q and $g : [N] \to \mathbf{B}$ is a partial function. Then there exists $j \in [N] \setminus Dom(g)$ such that

$$Pr_{x \to U_g}[F(x) \neq F(x \oplus ej)] \geq \frac{Var_{U_g}(F)}{q}.$$

Looking for Collisions with Influential Variables-I

Tool: known result on influential variables

Suppose that $F : \mathbf{S}[N] \to \mathbf{B}$ is computable by a decision tree of depth at most q and $g : [N] \to \mathbf{B}$ is a partial function. Then there exists $j \in [N] \setminus Dom(g)$ such that

$$Pr_{x \to U_g}[F(x) \neq F(x \oplus ej)] \geq \frac{Var_{U_g}(F)}{q}.$$

approximable in poly time

Next step for first-order functions

Given a function $F : \mathbf{S}[N] \to \mathbf{S}[L]$, we build a polytime algorithm that returns a short set of bits variables $J = \{j_1, \ldots, j_m\}$, and an associated function $g : J \to \mathbf{B}$, such that as soon as x is on the bits J as specified by F, the variance of F is negligeable.

Looking for Collisions with Influential Variables-I

Tool: known result on influential variables

Suppose that $F : \mathbf{S}[N] \to \mathbf{B}$ is computable by a decision tree of depth at most q and $g : [N] \to \mathbf{B}$ is a partial function. Then there exists $j \in [N] \setminus Dom(g)$ such that

$$Pr_{x \to U_g}[F(x) \neq F(x \oplus ej)] \geq \frac{Var_{U_g}(F)}{q}.$$

approximable in poly time

Next step for first-order functions

Given a function $F : \mathbf{S}[N] \to \mathbf{S}[L]$, we build a polytime algorithm that returns a short set of bits variables $J = \{j_1, \ldots, j_m\}$, and an associated function $g : J \to \mathbf{B}$, such that as soon as x is on the bits J as specified by F, the variance of F is negligeable.

Generalizing to second-order fonctions

for $F : (\mathbf{S}[n] \to \mathbf{B}) \to \mathbf{S}[L]$,
N polynomial in $n \leq 2^n$:

$$\tilde{F} : \mathbf{S}[N] \to \mathbf{S}[L]$$
$$x \mapsto F(\text{block-function}(x)).$$

Looking for Collisions with Influential Variables- II

Theorem

For every δ there is a strategy $coll_\delta$ on the game

$$!_t(!_p(\mathbf{S}[n] \multimap \mathbf{B}) \multimap \mathbf{S}[r]) \multimap (\mathbf{S}[n] \multimap \mathbf{B}) \otimes (\mathbf{S}[n] \multimap \mathbf{B})$$

such that for every deterministic strategy f, the composition $(!_s f) \circ coll_\delta$, with probability at least $1-\delta$, computes two functions g, h such that:

1. $H(g,h) \geq 0.1$;
2. $f \circ g$ *and* $f \circ h$ *behave the same;*
3. *For every function e on which $coll_\delta$ queries its argument, it holds that $H(e,g) \geq 0.1$ and $H(e,h) \geq 0.1$.*

Looking for Collisions with Influential Variables- II

Theorem

For every δ there is a strategy $coll_\delta$ on the game

$$!_t(!_p(\mathbf{S}[n] \multimap \mathbf{B}) \multimap \mathbf{S}[r]) \multimap (\mathbf{S}[n] \multimap \mathbf{B}) \otimes (\mathbf{S}[n] \multimap \mathbf{B})$$

such that for every deterministic strategy f, the composition $(!_s f) \circ coll_\delta$, with probability at least $1-\delta$, computes two functions g, h such that:

1. $H(g,h) \geq 0.1$;
2. $f \circ g$ *and* $f \circ h$ *behave the same;*
3. *For every function e on which $coll_\delta$ queries its argument, it holds that $H(e,g) \geq 0.1$ and $H(e,h) \geq 0.1$.*

Corollary

- *There are no collision resistant second-order scheme $(!_p(\mathbf{S}[n] \multimap \mathbf{B}) \multimap \mathbf{S}[r])$;*
- *thus there are no pseudo-random function for $X_n = (!_p(\mathbf{S}[n] \multimap \mathbf{B}) \multimap \mathbf{S}[r])$.*

The Positive Result

- Now, consider the type $SOF = \mathbf{S}[n] \multimap !_p(\mathbf{S}[q] \multimap \mathbf{B}) \multimap \mathbf{S}[r]$, where $q(n) \leq \log_2(n)$, and $p(n) \geq n$.

The Positive Result

- Now, consider the type $SOF = \mathbf{S}[n] \multimap !_p(\mathbf{S}[q] \multimap \mathbf{B}) \multimap \mathbf{S}[r]$, where $q(n) \leq \log_2(n)$, and $p(n) \geq n$.
- A deterministic strategy for $\mathbf{S}[q] \multimap \mathbf{B}$ can thus be seen as a binary string of length at most n.
  - Well, **more or less**: the string is accessed *interactively*, the access patter being visible to the adversary.

The Positive Result

- Now, consider the type $SOF = \mathbf{S}[n] \multimap !_p(\mathbf{S}[q] \multimap \mathbf{B}) \multimap \mathbf{S}[r]$, where $q(n) \leq \log_2(n)$, and $p(n) \geq n$.
- A deterministic strategy for $\mathbf{S}[q] \multimap \mathbf{B}$ can thus be seen as a binary string of length at most n.
  - Well, **more or less**: the string is accessed *interactively*, the access patter being visible to the adversary.
- When building a pseudorandom strategy f for the game SOF above, one needs to be sure that the way f accesses its argument is **itself** indistinguishable from a random one.

The Positive Result

- Now, consider the type $SOF = \mathbf{S}[n] \multimap !_p(\mathbf{S}[q] \multimap \mathbf{B}) \multimap \mathbf{S}[r]$, where $q(n) \leq \log_2(n)$, and $p(n) \geq n$.
- A deterministic strategy for $\mathbf{S}[q] \multimap \mathbf{B}$ can thus be seen as a binary string of length at most n.
  - Well, **more or less**: the string is accessed *interactively*, the access patter being visible to the adversary.
- When building a pseudorandom strategy f for the game SOF above, one needs to be sure that the way f accesses its argument is **itself** indistinguishable from a random one.

Theorem

If there is a one-way function, then there is a pseudorandom strategy for $\mathbf{S}[n] \multimap !_n(\mathbf{S}[\log_2(n)] \multimap \mathbf{B}) \multimap \mathbf{S}[r]$.

The Positive Result

- Now, consider the type $SOF = \mathbf{S}[n] \multimap !_p(\mathbf{S}[q] \multimap \mathbf{B}) \multimap \mathbf{S}[r]$, where $q(n) \leq \log_2(n)$, and $p(n) \geq n$.
- A deterministic strategy for $\mathbf{S}[q] \multimap \mathbf{B}$ can thus be seen as a binary string of length at most n.
  - Well, **more or less**: the string is accessed *interactively*, the access patter being visible to the adversary.
- When building a pseudorandom strategy f for the game SOF above, one needs to be sure that the way f accesses its argument is **itself** indistinguishable from a random one.

Theorem

If there is a one-way function, then there is a pseudorandom strategy for $\mathbf{S}[n] \multimap !_n(\mathbf{S}[\log_2(n)] \multimap \mathbf{B}) \multimap \mathbf{S}[r]$.

- **Corollary**: “function-authenticating-codes” are indeed possible, but for functions of type $\mathbf{S}[\log_2(n)] \multimap \mathbf{B}$.

Conclusion

Main Contributions

- A novel game-theoretic framework for higher-order, randomized, complexity bounded computation.
- Impossibility of building second-order functions having the expected type, (i.e. taking in input characteristic functions on $\{0,1\}^n$) and having good cryptographic properties.
- Existence, under standard cryptographic assumptions, of secord-order pseudrandom functions taking in input characteristic functions on $\{0,1\}^{\log_2(n)}$.

Conclusion

Main Contributions

- A novel game-theoretic framework for higher-order, randomized, complexity bounded computation.
- Impossibility of building second-order functions having the expected type, (i.e. taking in input characteristic functions on $\{0,1\}^n$) and having good cryptographic properties.
- Existence, under standard cryptographic assumptions, of secord-order pseudrandom functions taking in input characteristic functions on $\{0,1\}^{\log_2(n)}$.

Future Work

- How about **encryption**?
- Is it that our game-semantic framework can be seen as a methodology for proving higher-order cryptographic **reduction arguments** to be *complexity preserving*, or even *correct*?

Thank you !