On Higher-Order Cryptography

Ugo Dal Lago
(Based on joint work with Boaz Barak and Raphaëlle Crubillé)

ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA
Inria informatiques mathématiques

Meeting of the IFIP Working Group 2.2
September 20th, 2021

Cryptographic Reductions and Higher-Order Computation

- Security of Ψ in the computational model:

$$(\forall A \in \mathbf{PPT}.\neg(A\ \mathbf{Breaks}\ \Phi)) \Longrightarrow (\forall B \in \mathbf{PPT}.\neg(B\ \mathbf{Breaks}\ \Psi))$$

Cryptographic Reductions and Higher-Order Computation

- Security of Ψ in the computational model:

$$(\forall A \in \mathbf{PPT}.\neg(A\ \mathbf{Breaks}\ \Phi)) \Longrightarrow (\forall B \in \mathbf{PPT}.\neg(B\ \mathbf{Breaks}\ \Psi))$$

- By contraposition, this amounts to prove that:

$$(\exists B \in \mathbf{PPT}.(B\ \mathbf{Breaks}\ \Psi)) \Longrightarrow (\exists A \in \mathbf{PPT}.(A\ \mathbf{Breaks}\ \Phi))$$

Cryptographic Reductions and Higher-Order Computation

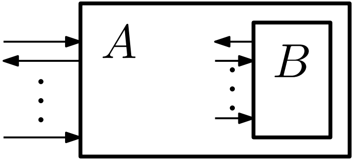
- Security of Ψ in the computational model:

$$(\forall A \in \mathbf{PPT}.\neg(A\ \mathbf{Breaks}\ \Phi)) \Longrightarrow (\forall B \in \mathbf{PPT}.\neg(B\ \mathbf{Breaks}\ \Psi))$$

- By contraposition, this amounts to prove that:

$$(\exists B \in \mathbf{PPT}.(B\ \mathbf{Breaks}\ \Psi)) \Longrightarrow (\exists A \in \mathbf{PPT}.(A\ \mathbf{Breaks}\ \Phi))$$

- Proofs are usually *constructive*:



$$\texttt{Reduction} : \mathit{TypeOf}(B) \to \mathit{TypeOf}(A)$$

Cryptographic Reductions and Higher-Order Computation

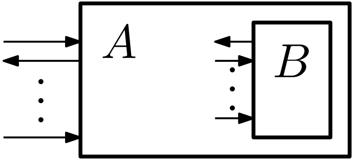
- Security of Ψ in the computational model:

$$(\forall A \in \mathbf{PPT}.\neg(A \textbf{ Breaks } \Phi)) \implies (\forall B \in \mathbf{PPT}.\neg(B \textbf{ Breaks } \Psi))$$

- By contraposition, this amounts to prove that:

$$(\exists B \in \mathbf{PPT}.(B \textbf{ Breaks } \Psi)) \implies (\exists A \in \mathbf{PPT}.(A \textbf{ Breaks } \Phi))$$

- Proofs are usually *constructive*:



$$\texttt{Reduction} : \mathit{TypeOf}(B) \to \mathit{TypeOf}(A)$$

- `Reduction` is a *complexity preserving* higher-order function.

An Example from [KatzLindell2008]

DEFINITION 3.25 *Let $F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ be an efficient, length-preserving, keyed function. F is a* pseudorandom function *if for all probabilistic polynomial-time distinguishers D, there is a negligible function* negl *such that:*

$$\left| \Pr[D^{F_k(\cdot)}(1^n) = 1] - \Pr[D^{f(\cdot)}(1^n) = 1] \right| \leq \mathsf{negl}(n),$$

where the first probability is taken over uniform choice of $k \in \{0,1\}^n$ and the randomness of D, and the second probability is taken over uniform choice of $f \in \mathsf{Func}_n$ and the randomness of D.

An Example from [KatzLindell2008]

DEFINITION 3.25 *Let* $F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ *be an efficient, length-preserving, keyed function.* F *is a* pseudorandom function *if for all probabilistic polynomial-time distinguishers* D*, there is a negligible function* negl *such that:*

$$\left| \Pr[D^{F_k(\cdot)}(1^n) = 1] - \Pr[D^{f(\cdot)}(1^n) = 1] \right| \le \mathsf{negl}(n),$$

D **Breaks** F

where the first probability is taken over uniform choice of $k \in \{0,1\}^n$ *and the randomness of* D*, and the second probability is taken over uniform choice of* $f \in \mathsf{Func}_n$ *and the randomness of* D*.*

An Example from [KatzLindell2008]

DEFINITION 3.25 *Let* $F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ *be an efficient, length-preserving, keyed function.* F *is a* pseudorandom function *if for all probabilistic polynomial-time distinguishers* D*, there is a negligible function* negl *such that:*

$$\left| \Pr[D^{F_k(\cdot)}(1^n) = 1] - \Pr[D^{f(\cdot)}(1^n) = 1] \right| \leq \mathsf{negl}(n),$$

D **Breaks** F

where the first probability is taken over uniform choice of $k \in \{0,1\}^n$ *and the randomness of* D*, and the second probability is taken over uniform choice of* $f \in \mathsf{Func}_n$ *and the randon*

D accesses f as an oracle

… $\times \{0,1\}^* \to \{0,1\}^*$ *be an efficient, length-preserving, keyed function. F is a* pseudorandom function *if for all probabilistic polynomial-time distinguishers D, there is a negligible function* negl *such that:*

$$\left| \Pr[D^{F_k(\cdot)}(1^n) = 1] - \Pr[D^{f(\cdot)}(1^n) = 1] \right| \leq \mathsf{negl}(n),$$

where the first probability is taken over uniform choice of $k \in \{0,1\}^n$ and the randomness of D, and the second probability is taken over uniform choice of $f \in \mathsf{Func}_n$ and the random …

The first parameter is fixed to be k

D **Breaks** F

D accesses f as an oracle

The Cryptographer's Notion of Efficiency [KatzLindell2008]

We equate "efficient adversaries" with randomized (i.e., probabilistic) algorithms running in time *polynomial in* n. This means there is some polynomial p such that the adversary runs for time at most $p(n)$ when the security parameter is n. We also require—for real-world efficiency—that honest parties run in polynomial time, although we stress that the adversary may be much more powerful (and run much longer than) the honest parties.

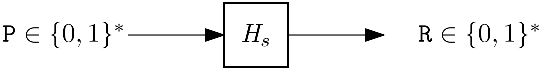
Equivalence-Preserving Cryptography?

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

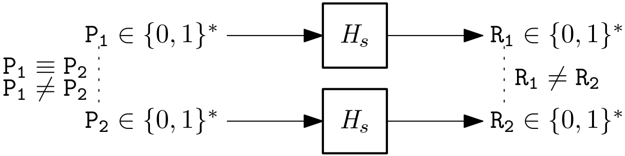
$$\mathtt{P} \in \{0,1\}^* \longrightarrow \boxed{H_s} \longrightarrow \mathtt{R} \in \{0,1\}^*$$

Equivalence-Preserving Cryptography?

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

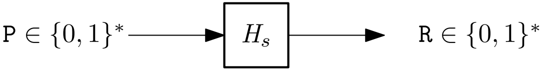


- If two programs P and Q are perfectly equivalent but distinct, they are thus seen as distinct strings, and mapped to distinct hashes:

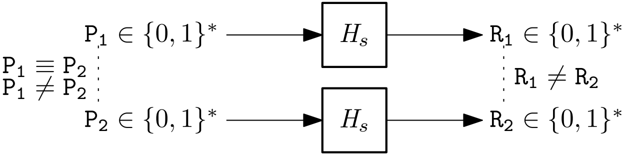


Equivalence-Preserving Cryptography?

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



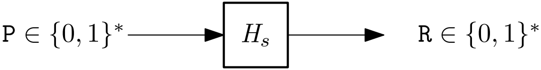
- If two programs P and Q are perfectly equivalent but distinct, they are thus seen as distinct strings, and mapped to distinct hashes:



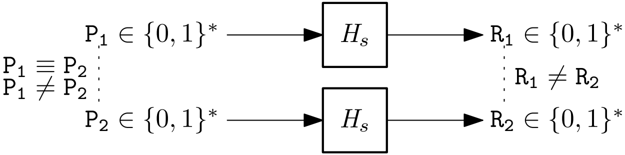
- The same argument holds when H_s is replaced by Enc_k (i.e. encryption) or Mac_k (i.e. authentication).

Equivalence-Preserving Cryptography?

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



- If two programs P and Q 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 $\{0,1\}^* \to \{0,1\}^*$).
  - E.g., hashing distinct but equivalent programs can be done *only once*.

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 order higher-than- 2 instead requires some care.
   - Game semantics [AJM00,HO00] offers a way to reduce higher-order computation to first-order computation.

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 order higher-than- 2 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.

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

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

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

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

O			?

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

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

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

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

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

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

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

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

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

	$(\{0,1\}^*$	$\to$	$\{0,1\}^*)$	$\to$	$\{0,1\}^*$
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

	$(\{0,1\}^*$	$\rightarrow$	$\{0,1\}^*)$	$\rightarrow$	$\{0,1\}^*$
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

	$(\{0,1\}^*$	$\rightarrow$	$\{0,1\}^*)$	$\rightarrow$	$\{0,1\}^*$
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

- The Player P must react to every move of the opponent O.
- It can do so based on *the whole* history, without any further constraint. This defines a strategy.
- The *length* of the interaction is in principle arbitrary.
- Multiple moves can be played *at the same site*, but they somehow need to be distinguished.

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 $\leq p(n)$)

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

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

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:
$\forall n \in \mathbb{N}, \forall s \in L^n_G, |s| \leq P(n)$.

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, respectively;
- $!_q G$ modeling $q(n)$ copies of G.

Cryptographic Game Semantics — II

Proposition (Compositing Strategies)

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

Cryptographic Game Semantics — II

Proposition (Compositing Strategies)

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

- How about randomization?

Cryptographic Game Semantics — II

Proposition (Compositing Strategies)

If f, g polytime strategies on $G \multimap H$ and $H \multimap K$ (respectively), one can form $g \circ f$ as a ***polytime*** *strategy on $G \multimap K$. 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 $f : \mathbb{N} \times (L_G^n \cap \mathrm{Odd}) \rightarrow \mathsf{DISTR}(P_G)$

Cryptographic Game Semantics — II

Proposition (Compositing Strategies)

If f, g polytime strategies on $G \multimap H$ and $H \multimap K$ (respectively), one can form $g \circ f$ as a ***polytime*** *strategy on $G \multimap K$. 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 $f : \mathbb{N} \times (L_G^n \cap \mathrm{Odd}) \rightarrow \mathsf{DISTR}(P_G)$

NO!

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

Cryptographic Game Semantics — II

Proposition (Compositing Strategies)

If f, g polytime strategies on $G \multimap H$ and $H \multimap K$ (respectively), one can form $g \circ f$ as a ***polytime*** *strategy on $G \multimap K$. 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 (Compositing Strategies)

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

- How about randomization?

Randomized Strategies—A First Try

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

NO!

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

Randomized Games—Second Try

- Games: $G = (O_G, P_G, (L^n_G)_{n \in \mathbb{N}})$
- 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.

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							$\neg c \wedge d$

An Example: Pseudorandomness

An family of "objects" $\{X_n\}_{n\in\mathbb{N}}$ is said to be **pseudorandom** iff X_n is indistinguishable from a genuinely random object of the same type, by distinguishers working in polynomial time (in n).

An Example: Pseudorandomness

An family of "objects" $\{X_n\}_{n\in\mathbb{N}}$ is said to be **pseudorandom** iff X_n is indistinguishable from a genuinely random object of the same type, by distinguishers working in polynomial time (in n).

Then:

- If $X_n = \{0,1\}^{r(n)}$, then the distinguisher is just a polytime TM.
- If $X_n = \{0,1\}^n \to \{0,1\}^n$, then the distinguisher is a polytime (in n) OTM.
- If $X_n = (\{0,1\}^n \to \{0,1\}^1) \to \{0,1\}^n$, then there was no clear answer to this question. But now we have a model!

An Example: Pseudorandomness

An family of "objects" $\{X_n\}_{n\in\mathbb{N}}$ is said to be **pseudorandom** iff X_n is indistinguishable from a genuinely random object of the same type, by distinguishers working in polynomial time (in n).

Then:

- If $X_n = \{0,1\}^{r(n)}$, then the distinguisher is just a polytime TM.
- If $X_n = \{0,1\}^n \to \{0,1\}^n$, then the distinguisher is a polytime (in n) OTM.
- If $X_n = (\{0,1\}^n \to \{0,1\}^1) \to \{0,1\}^n$, then there was no clear answer to this question. But now we have a model!

Higher-Order Pseudorandomness?

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

$$\{0,1\}^n \to (\{0,1\}^n \to \{0,1\}^1) \to \{0,1\}^n$$

  which "look random": a truly random function "query" its argument exponentially many times, namely on all the strings in $\{0,1\}^n$.
- How to turn this into a formal argument in the just introduced model?

Second-Order Pseudorandomness

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

Second-Order Pseudorandomness

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

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

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

  $$ADV_{s,t,p,q,r} = !_s\mathbf{B} \multimap !_t(!_p(\mathbf{S}[q] \multimap \mathbf{B}) \multimap \mathbf{S}[r]) \multimap \mathbf{B}$$

Second-Order Pseudorandomness

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

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

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

$$ADV_{s,t,p,q,r} = !_s\mathbf{B} \multimap !_t(!_p(\mathbf{S}[q] \multimap \mathbf{B}) \multimap \mathbf{S}[r]) \multimap \mathbf{B}$$

We say that a polytime strategy f for the game $SOF_{p,q,r}$ is **pseoudorandom** iff for any polytime strategy $\mathcal{A}$ for the game $ADV_{s,t,p,q,r}$ it holds that

$$|Pr[\mathcal{A} \circ (f \circ rand) \Downarrow 1] - Pr[\mathcal{A} \circ (rand) \Downarrow 1]| \leq \varepsilon(n)$$

where ε is a negligible function.

The Two Results

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 $H(e, h) \geq 0.1$.*

The Two Results

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 $H(e, h) \geq 0.1$.*

Theorem

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

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 not only *complexity preserving*, but even *correct*?

Thank You!

Questions?

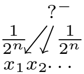
Example

$F : \{0,1\}^n \to \{0,1\}^{P(n)}$ a one-way function.

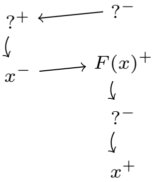
Example

$F : \{0,1\}^n \to \{0,1\}^{P(n)}$ a one-way function.

$g : \mathbf{1} \multimap \mathbf{S}[X]$



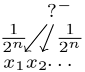
$f : \mathbf{S}[X] \multimap \mathbf{S}[P] \oslash \mathbf{S}[X]$



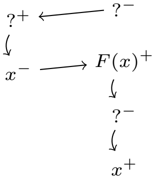
Example

$F : \{0,1\}^n \to \{0,1\}^{P(n)}$ a one-way function.

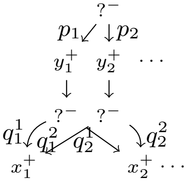
$g : \mathbf{1} \multimap \mathbf{S}[X]$



$f : \mathbf{S}[X] \multimap \mathbf{S}[P] \oslash \mathbf{S}[X]$

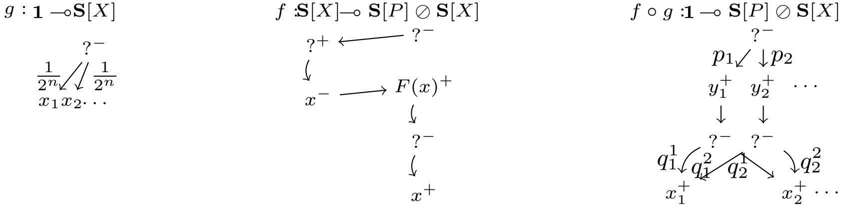


$f \circ g : \mathbf{1} \multimap \mathbf{S}[P] \oslash \mathbf{S}[X]$



Example

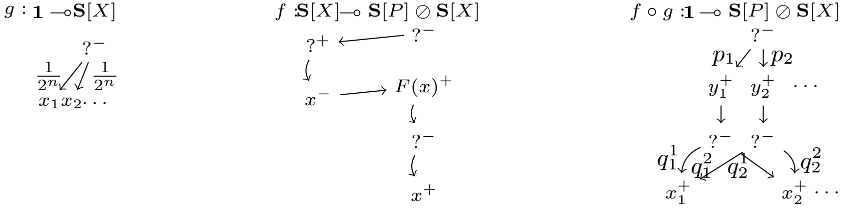
$F : \{0,1\}^n \to \{0,1\}^{P(n)}$ a one-way function.



Compute $f \circ g(?y_i)$ boils down to finding an element in $F^{-1}(y_i)$.

Example

$F : \{0,1\}^n \to \{0,1\}^{P(n)}$ a one-way function.



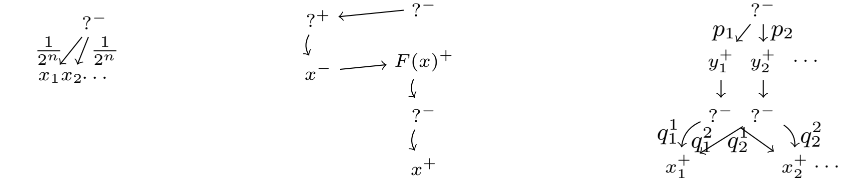
Compute $f \circ g(?y_i)$ boils down to finding an element in $F^{-1}(y_i)$.

- f, g are polytime computable functions on bounded games.

Example

$F : \{0,1\}^n \to \{0,1\}^{P(n)}$ a one-way function.

$g : \mathbf{1} \multimap \mathbf{S}[X]$ $\quad f : \mathbf{S}[X] \multimap \mathbf{S}[P] \oslash \mathbf{S}[X]$ $\quad f \circ g : \mathbf{1} \multimap \mathbf{S}[P] \oslash \mathbf{S}[X]$



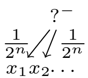
Compute $f \circ g(?y_i)$ boils down to finding an element in $F^{-1}(y_i)$.

- f, g are polytime computable functions on bounded games.

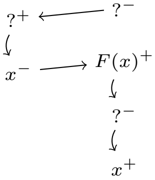
Example

$F : \{0,1\}^n \to \{0,1\}^{P(n)}$ a one-way function.

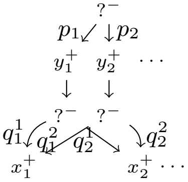
$g : \mathbf{1} \multimap \mathbf{S}[X]$



$f : \mathbf{S}[X] \multimap \mathbf{S}[P] \oslash \mathbf{S}[X]$



$f \circ g : \mathbf{1} \multimap \mathbf{S}[P] \oslash \mathbf{S}[X]$



Compute $f \circ g(?y_i)$ boils down to finding an element in $F^{-1}(y_i)$.

- f, g are polytime computable functions on bounded games.
- $f \circ g : \mathbf{1} \multimap \mathbf{S}[P] \oslash \mathbf{S}[X]$ is not polytime computable.