How to Treat Programs in a Cryptographic Scenario?

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

\[ P \in \{0, 1\}^* \rightarrow H_s \rightarrow R \in \{0, 1\}^* \]
How to Treat Programs in a Cryptographic Scenario?

- Programs, e.g. when hashed, are usually treated as strings:
  \[ P \in \{0, 1\}^* \xrightarrow{H_s} R \in \{0, 1\}^* \]

- If two programs \( P_1 \) and \( P_2 \) are perfectly equivalent but distinct, they are thus seen as distinct strings, and mapped to distinct hashes:
  \[ P_1 \in \{0, 1\}^* \xrightarrow{H_s} R_1 \in \{0, 1\}^* \]
  \[ P_2 \in \{0, 1\}^* \xrightarrow{H_s} R_2 \in \{0, 1\}^* \]
  \[ P_1 \equiv P_2 \]
  \[ P_1 \neq P_2 \]
  \[ R_1 \neq R_2 \]
How to Treat Programs in a Cryptographic Scenario?

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

  \[ P \in \{0, 1\}^* \xrightarrow{H_s} R \in \{0, 1\}^* \]

- If two programs \( P_1 \) and \( P_2 \) are perfectly equivalent but distinct, they are thus seen as distinct strings, and mapped to distinct hashes:

  \[ P_1 \equiv P_2 \quad P_1 \neq P_2 \quad P_1 \in \{0, 1\}^* \xrightarrow{H_s} R_1 \in \{0, 1\}^* \]
  \[ P_2 \in \{0, 1\}^* \xrightarrow{H_s} R_2 \in \{0, 1\}^* \]

- 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:
  \[ P \in \{0, 1\}^* \rightarrow H_s \rightarrow R \in \{0, 1\}^* \]

- If two programs \( P_1 \) and \( P_2 \) are perfectly equivalent but distinct, they are thus seen as distinct strings, and mapped to distinct hashes:
  \[
  \begin{align*}
  P_1 &\in \{0, 1\}^* \rightarrow H_s \rightarrow R_1 \in \{0, 1\}^* \\
  P_2 &\in \{0, 1\}^* \rightarrow H_s \rightarrow R_2 \in \{0, 1\}^*
  \end{align*}
\]

- 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 \( \{0, 1\}^* \rightarrow \{0, 1\}^* \rightarrow \{0, 1\}^* \) (rather than \( \{0, 1\}^* \rightarrow \{0, 1\}^* \)).
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

\[
S = (S_n : \{0,1\}^n \rightarrow X_n)_{n \in \mathbb{N}}
\]

is pseudo-random

key deterministic function

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

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

eignegligible function
Pseudorandomness in Cryptography

\[ S = (S_n : \{0, 1\}^n \rightarrow X_n)_{n \in \mathbb{N}} \]

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: \( X_n = \{0, 1\}^{r(n)} \rightarrow \{0, 1\}^{l(n)} \). 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 exist;
- PRNG exist \( \Rightarrow P \neq NP \);
- PRNG exists \( \Rightarrow \) PRF exist.
Collision Resistance

**Definition**

A scheme

\[ S = (S_n : \{0, 1\}^n \rightarrow (Y_n \rightarrow Z_n))_{n \in \mathbb{N}} \]

is collision-resistant

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

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

is negligible.

**Fact**

As soon as \( (n \mapsto \frac{\text{card}(Y_n)}{2 \cdot \text{card}(Z_n)}) \) is negligible, a truly random \( S \) is collision resistant.

E.g. \( Y_n = (\{0, 1\}^{p(n)} \rightarrow \{0, 1\}) \), \( Z_n = \{0, 1\}^{q(n)} \) is collision-resistant when \( p(n) \leq q(n) \).
Higher-Order Pseudorandomness?

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

\[
S : \{0, 1\}^n \rightarrow ((\{0, 1\}^n \rightarrow \{0, 1\}) \rightarrow \{0, 1\}^n) 
\]

which “look random”.

**Intuition:**
- the input function can be accessed only polynomially many times by the efficient algorithm \(S\);
- a truly random \(F\) in \((\{0, 1\}^n \rightarrow \{0, 1\} \rightarrow \{0, 1\}^n)\) 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 $X_n = \{0, 1\}^n \rightarrow \{0, 1\}^n$, then the distinguisher is of order 2: can be taken as a polytime (in $n$) oracle randomized Turing machine.
- If $X_n = (\{0, 1\}^n \rightarrow \{0, 1\}^1) \rightarrow \{0, 1\}^n$, 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

\[ !\left( \{0, 1\}^* \rightarrow \{0, 1\}^* \right) \rightarrow \{0, 1\}^* \]
An Example: the Game Semantics of a Second-Order Function

\[ \forall \{0,1\}^* \rightarrow \{0,1\}^* \rightarrow \{0,1\}^* \]

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

\( !\left( \{0,1\}^* \to \{0,1\}^* \right) \to \{0,1\}^* \)

O
P

O
P

\( (1, ?) \)
\( (1, s_1) \)

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

$\forall \{{0, 1}\}^* \to \{0, 1\}^* \to \{0, 1\}^*$

$O$ ?

$P$ $(1, ?)$

$O$ $(1, ?)$

$P$ $(1, s_1)$

$O$ $(1, t_1)$
An Example: the Game Semantics of a Second-Order Function

\[ \forall(\{0, 1\}^\ast \to \{0, 1\}^\ast) \to \{0, 1\}^\ast \]

\[
\begin{align*}
O & \quad (1, '?') \\
P & \quad (1, s_1) \\
O & \quad (1, t_1) \\
\vdots \\
P & \quad (m, '?') \\
O & \quad (m, '?') \\
P & \quad (m, s_m) \\
O & \quad (m, t_m)
\end{align*}
\]
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

\[ !\{0,1\}^* \rightarrow \{0,1\}^* \rightarrow \{0,1\}^* \]

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: \( G = (O_G, P_G, (L^n_G)_{n \in \mathbb{N}}) \)
- Strategies: \( f : \mathbb{N} \times (L^n_G \cap \text{Odd}) \rightarrow P_G \)
Cryptographic Game Semantics - I

Games Parametrized by a Security Parameter

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

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

\[ S[p] = (\{?, \{0, 1\}^*, (L^n_S[p])_{n \in \mathbb{N}}) \text{ with } \]
\[ L^n_S[p] = \{\epsilon, ?\} \cup \{?s \mid |s| \leq p(n)\} \]
Cryptographic Game Semantics - I

Games Parametrized by a Security Parameter

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

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

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

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 $f(n, s)$.  

Boaz Barak,  Raphaëlle Crubillé,  Ugo Dal Lago  On Higher-Order Cryptography
Cryptographic Game Semantics - I

Games Parametrized by a Security Parameter
- Games: \( G = (O_G, P_G, (L_G^n)_{n \in \mathbb{N}}) \)
- Strategies: \( f : \mathbb{N} \times (L_G^n \cap \text{Odd}) \to P_G \)

Example: Strings of Length \( \leq p(n) \)
\( S[p] = (\{?, \}, \{0, 1\}^*, (L_S^{S[p]}_n)_{n \in \mathbb{N}}) \) with
\( L_S^{S[p]}_n = \{\epsilon, ?\} \cup \{?s \mid |s| \leq p(n)\} \)

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 \( f(n, s) \).

Constructing Games
From the games \( G, H \), we can construct more complex games such as:
- \( G \rightarrow H \), modeling functions from \( G \) to \( H \);
- \( G \otimes H \), modeling pairs of elements from \( G \) and \( H \);
- \(!_q G \) modeling \( q(n) \) copies of \( G \).
Proposition (Composing Strategies)

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

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

- How about randomization?
Proposition (Composing Strategies)

If $f, g$ polytime strategies on $G \rightarrow H$ and $H \rightarrow K$ (respectively), one can form $g \circ f$ as a strategy on $G \rightarrow K$. Moreover, strategy 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 $f : \mathbb{N} \times (L^n_G \cap \text{Odd}) \rightarrow \text{DISTR}(P_G)$
Cryptographic Game Semantics — II

Proposition (Composing Strategies)

If \( f, g \) polytime strategies on \( G \rightarrow H \) and \( H \rightarrow K \) (respectively), one can form \( g \circ f \) as a strategy on \( G \rightarrow K \). Moreover, strategy 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 \( f : \mathbb{N} \times (L^n_G \cap \text{Odd}) \rightarrow \text{DISTR}(P_G) \)

NO!

Randomized polytime strategies are not stable by composition.
Proposition (Composing Strategies)

If \( f, g \) polytime strategies on \( G \to H \) and \( H \to K \) (respectively), one can form \( g \circ f \) as a strategy on \( G \to K \). Moreover, strategy 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 \( f : \mathbb{N} \times (L^n_G \cap \text{Odd}) \to \text{DISTR}(P_G) \)

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 \( !_pB \to G \) (where \( B \) is the boolean game).

NO!
Randomized polytime strategies are not stable by composition.
Proposition (Composing Strategies)

If \( f, g \) polytime strategies on \( G \rightarrow H \) and \( H \rightarrow K \) (respectively), one can form \( g \circ f \) as a strategy on \( G \rightarrow K \). Moreover, strategy 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 \( f : \mathbb{N} \times (L^n_G \cap \text{Odd}) \rightarrow \text{DISTR}(P_G) \)

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 \( !_p B \rightarrow G \) (where \( B \) is the boolean game).

NO!

Randomized polytime strategies are not stable by composition.

YES!

The whole sequence of probabilistic choices is available, and strategies compose.
Randomized Strategies—A First Try

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

**Fact:**

Polytime randomized strategies do not compose.

<table>
<thead>
<tr>
<th>$S[n] \rightarrow !^2S[n]$</th>
<th>$S[n]$</th>
<th>$!^2S[n]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>(1, ?s[n])</td>
<td>O</td>
</tr>
<tr>
<td>P ?s[n]</td>
<td></td>
<td>(1, ?s[n])</td>
</tr>
<tr>
<td>O s</td>
<td></td>
<td>O</td>
</tr>
<tr>
<td>P (1, h(s))</td>
<td></td>
<td>P $l_1$</td>
</tr>
<tr>
<td>O (2, ?s[n])</td>
<td></td>
<td>l_2</td>
</tr>
<tr>
<td>P 2, s</td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>$f_h$: polytime strategy</td>
<td>Unif: polytime prob. strategy</td>
<td>$f_h \circ \text{Unif}$: not-polytime prob. strategy</td>
</tr>
</tbody>
</table>

$h$: one-way permutation
Definition

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

$$(q_2, f_2) \circ (q_1, f_1) : (q_2 + q_1, !_qB \times !_qB \to !_qB \times !_qB \to !_qB \otimes G \xrightarrow{f_2} H)$$

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

$$\text{Prob}_f^n(b) = \sum_{(b_1, \ldots, b_k) \in B^k} \frac{1}{2^k} \quad \text{for } f : !_pB \to B, b \in B.$$
An Example: a Simple Randomized Strategy

\[ \!_1 B \rightarrow \!_2 (S[n] \rightarrow B) \rightarrow B \]
An Example: a Simple Randomized Strategy

$$!_1 \mathcal{B} \rightarrow !_2 (S[n] \rightarrow \mathcal{B}) \rightarrow \mathcal{B}$$

$$O$$

$$P \quad (1, ?)$$

$$O \quad (1, b)$$
An Example: a Simple Randomized Strategy

$!_1 B \leadsto !_2 (S[n] \leadsto B) \leadsto B$

O

P $(1, ?)$

O $(1, b)$

P $(1, ?)$

O $(1, ?)$

P $(1, b^n)$

O $(1, c)$
An Example: a Simple Randomized Strategy

$$\begin{align*}
!_1 B & \rightarrow !_2 (S[n] \rightarrow B) \rightarrow B \\
O & \\
P & (1, ?) \\
O & (1, b) \\
P & (1, b^n) \\
O & (1, c) \\
P & (2, ?) \\
O & (2, ?) \\
P & (2, (\neg b)^n) \\
O & (2, d)
\end{align*}$$
An Example: a Simple Randomized Strategy

\[ !_1 B \leadsto !_2 (S[n] \leadsto B) \leadsto B \]

\begin{align*}
  O & \quad (1, ?) \\
  P & \quad (1, ?) \\
  O & \quad (1, b) \\
  P & \quad (1, b^n) \\
  O & \quad (1, c) \\
  P & \quad (2, ?) \\
  O & \quad (2, ?) \\
  P & \quad (2, (\neg b)^n) \\
  O & \quad (2, d) \\
  P & \quad \neg c \land d
\end{align*}
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

\[
\text{SOF} = \mathbb{S}[n] \to!_p (\mathbb{S}[q] \to \mathbb{B}) \to \mathbb{S}[r],
\]

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

\[
\text{ADV} = !_s \mathbb{B} \to !_t (!_p (\mathbb{S}[q] \to \mathbb{B}) \to \mathbb{S}[r]) \to \mathbb{B}
\]
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

  \[ SOF = S[n] \circ !_p(S[q] \circ B) \circ S[r], \]

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

  \[ ADV = !_s B \circ !_t(!_p(S[q] \circ B) \circ S[r]) \circ B \]

We say that a polytime strategy \( f \) for the game \( SOF \) is **pseudorandom** iff for any polytime strategy \( A \) for the game \( ADV \) it holds that

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

where \( \varepsilon \) is a negligible function and \( rand_G \) is a random strategy for the game \( G \).
The Negative Result: Summary

- Consider a strategy $f$ for $SOF = S[n] \xrightarrow{p} (S[q] \xrightarrow{B} S[r])$, 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 $\text{SOF} = S[n] \rightarrow_p (S[q] \rightarrow \text{B}) \rightarrow S[r]$, 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 \( SOF = S[n] \rightarrow_p (S[q] \rightarrow B) \rightarrow S[r] \), 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 $j \in [N] \setminus \text{Dom}(g)$ such that

$$\Pr_{x \to_U g}[F(x) \neq F(x \oplus ej)] \geq \frac{\text{Var}_{Ug}(F)}{q}.$$
Looking for Collisions with Influential Variables-I

Tool: known result on influential variables

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

$$\Pr_{x \rightarrow U_g}[F(x) \neq F(x \oplus ej)] \geq \frac{\text{Var}_{U_g}(F)}{q}.$$ 

approximable in poly time

Next step for first-order functions

Given a function $F : S[N] \rightarrow S[L]$, we build a polytime algorithm that returns a short set of bits variables $J = \{j_1, \ldots, j_m\}$, and an associated function $g : J \rightarrow B$, such that as soon as $x$ is on the bits $J$ as specified by $F$, the variance of $F$ is negligible.
Looking for Collisions with Influential Variables-I

Tool: known result on influential variables

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

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

approximable in poly time

Next step for first-order functions

Given a function $F : S[N] \to S[L]$, we build a polytime algorithm that returns a short set of bits variables $J = \{j_1, \ldots, j_m\}$, and an associated function $g : J \to B$, 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 $F : (S[n] \to B) \to S[L]$, $N$ polynomial in $n \leq 2^n$:

$\tilde{F} : S[N] \to S[L]$  

$x \mapsto F(\text{block-function}(x)).$
Looking for Collisions with Influential Variables- II

Theorem

For every $\delta$ there is a strategy $\text{coll}_\delta$ on the game

$$!_t(!_p(S[n] \rightarrow B) \rightarrow S[r]) \rightarrow (S[n] \rightarrow B) \otimes (S[n] \rightarrow B)$$

such that for every deterministic strategy $f$, the composition $(!_s f) \circ \text{coll}_\delta$, 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 $\text{coll}_\delta$ 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* $\delta$ *there is a strategy* $\text{coll}_\delta$ *on the game

$$\!_t(\!_p(S[n] \rightarrow B) \rightarrow S[r]) \rightarrow (S[n] \rightarrow B) \otimes (S[n] \rightarrow B)$$

*such that for every deterministic strategy* $f$, *the composition* $(\!_s f) \circ \text{coll}_\delta$, *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 $\text{coll}_\delta$ 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 $(\!_p(S[n] \rightarrow B) \rightarrow S[r])$;
- thus there are no pseudo-random function for $X_n = (\!_p(S[n] \rightarrow B) \rightarrow S[r])$. 
The Positive Result

- Now, consider the type \( \text{SOF} = \text{S}[n] \rightarrow!_p(\text{S}[q] \rightarrow \text{B}) \rightarrow \text{S}[r] \), where \( q(n) \leq \log_2(n) \), and \( p(n) \geq n \).
The Positive Result

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

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

- A deterministic strategy for \( S[q] \rightarrow B \) can thus be seen as a binary string of length at most \( n \).
  - Well, more or less: the string is accessed interactively, the access pattern 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 $SOF = S[n] \rightarrow !_p(S[q] \rightarrow B) \rightarrow S[r]$, where $q(n) \leq \log_2(n)$, and $p(n) \geq n$.
- A deterministic strategy for $S[q] \rightarrow B$ can thus be seen as a binary string of length at most $n$.
  - Well, more or less: the string is accessed interactively, the access pattern 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 $S[n] \rightarrow !_n(S[\log_2(n)] \rightarrow B) \rightarrow S[r]$.***
The Positive Result

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

- A deterministic strategy for $S[q] \to B$ can thus be seen as a binary string of length at most $n$.
  - Well, more or less: the string is accessed interactively, the access pattern 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 $S[n] \to!_n (S[\log_2(n)] \to B) \to S[r]$.***

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

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 second-order pseudorandom 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 second-order pseudorandom 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!