Randomized Encoding of Functions

Yuval Ishai

Technion and UCLA
Overview

• Can we make a computation simpler by just encoding the output?
• Question originally motivated by secure computation
• Answers have found applications in other areas of cryptography and elsewhere
Garbled Circuit Construction

Yao, 1986
Garbled Circuit Construction

Circuit C

Garbled circuit C'

Pairs of short keys

$C(x)$

$C', K_i, x_i$
Even more abstractly...

dependence on x is “simple”

randomness
The General Question

- $g$ is a “randomized encoding” of $f$
  - Nontrivial relaxation of computing $f$
- Hope:
  - $g$ can be “simpler” than $f$
    (meaning of “simpler” determined by application)
  - $g$ can be used as a substitute for $f$
Applications

- Secure computation [Yao82…]
- Parallel cryptography [AIK04…]
- One-time programs [GKR08…]
- KDM-secure encryption [BHHI10…]
- Verifiable computation [GGP10…]
- Functional encryption [SS10…]

…
Rest of Tutorial

• Constructions of randomized encodings
  – Different notions of simplicity
  – Different mathematical tools
    • Finite groups
    • Linear algebra
    • Number theory
  – Focus on information-theoretic security
    • Not in this tutorial: “succinct” and “reusable” variants

• Applications
  – Secure multiparty computation
  – Parallel cryptography
Randomized Encoding - Syntax

\[ f(x) \text{ is encoded by } g(x, r) \]
Randomized Encoding - Semantics

• Correctness: $f(x)$ can be efficiently decoded from $g(x,r)$.
  $$f(x) \neq f(w) \Rightarrow$$

  ![Diagram](image1)

• Privacy: $\exists$ efficient simulator $\text{Sim}$ such that $\text{Sim}(f(x)) \equiv g(x,U)$
  – $g(x,U)$ depends only on $f(x)$
  $$f(x) = f(w) \Rightarrow$$

  ![Diagram](image2)
Notions of Simplicity

2-Decomposable encoding
\[ g((x,y),r) = (g_x(x,r), g_y(y,r)) \]

Decomposable encoding
\[ g((x_1, \ldots, x_n), r) = (g_1(x_1, r), \ldots, g_n(x_n, r)) \]

NC⁰ encoding
Output locality c

Low-degree encoding
Algebraic degree d over F

AKA: projective garbling scheme [BHR12]
2-Decomposable Encodings

- \( g((x_A, x_B), r) = (g_A(x_A, r), g_B(x_B, r)) \)

- **Application:** “minimal model for secure computation” [Feige-Kilian-Naor 94, …]
Example: sum

- \( f(x_A, x_B) = x_A + x_B \) (\( x_A, x_B \in \) finite group \( G \))

\[ m_A + m_B \]
Example: equality

- \( f(x_A, x_B) = \text{equality} \) \( (x_A, x_B \in \text{finite field } F) \)

Alice

- \( x_A \)
- \( r_1 x_A + r_2 \)

Bob

- \( x_B \)
- \( r_1 x_B + r_2 \)

Carol

- \( m_A = m_B ? \)

Conditions:
- \( r_1 \in \mathbb{F} \setminus \{0\}, r_2 \in \mathbb{F} \)
Example: ANY function

- $f(x_A, x_B) = x_A \land x_B$ $(x_A, x_B \in \{0, 1\})$
  - Reduction to equality: $x_A \not\rightarrow 1/0$, $x_B \not\rightarrow 2/0$

- **General boolean $f$: write as disjoint 2-DNF**
  - $f(x_A, x_B) = \bigvee_{(a,b):f(a,b)=1} (x_A = a \land x_B = b) = t_1 \lor t_2 \lor \ldots \lor t_m$

Exponential complexity
• **Decomposability:** \( g((x_1,\ldots,x_n),r) = (g_1(x_1,r),\ldots,g_n(x_n,r)) \)
  
  – Application: Basing 2-PC on OT [Kilian 88, ...]

---

**Diagram:**

- **Alice:**
  - \( g_A(x_A,r) \)
  - \( x_A \)
- **Bob:**
  - \( x_B \)
  - \( f(x_A,x_B) \)
- **Malicious Bob?**
  - \( g_n(0,r) \)
  - \( g_n(1,r) \)
  - \( x_n \)
  - \( g_n(x_n,r) \)
Decomposable Encodings

- **Decomposability:** $g((x_1, \ldots, x_n), r) = (g_1(x_1, r), \ldots, g_n(x_n, r))$

  - Application: Basing 2-PC on OT [Kilian 88, ...]

![Diagram of 2-PC based on OT with malicious checks for Alice and Bob]
Example: iterated group product

- **Abelian case**
  - \( f(x_1, \ldots, x_n) = x_1 + x_2 + \ldots + x_n \)
  - \( g(x, (r_1, \ldots, r_{n-1})) = x_1 + r_1 x_2 + r_2 \ldots + x_{n-1} + r_{n-1} x_n \)

- **General case** [Kilian 88]
  - \( f(x_1, \ldots, x_n) = x_1 x_2 \ldots x_n \)
  - \( g(x, (r_1, \ldots, r_{n-1})) = x_1 r_1 r_2^{-1} x_2 r_3^{-1} x_3 \ldots r_{n-2}^{-1} x_{n-1} r_{n-1}^{-1} x_n \)
Example: iterated group product

Thm [Barrington 86]
Every boolean \( f \in \text{NC}^1 \) can be computed by a poly-length, width-5 branching program.

\[ f(x_1, \ldots, x_n) \text{ reduces to } \pi_1 \cdot \pi_2 \cdots \pi_m \text{ where:} \]

- Each \( \pi_i \) depends on a single \( x_j \)
- Distinct \( \sigma_0, \sigma_1 \in S_5 \) s.t. \( x_1^{\sigma_0} x_2^{\sigma_0} \cdots x_m^{\sigma_0} = f(x) \)

Encoding iterated group product
- Every output bit of \( g \) depends on just a single bit of \( x \)
  - Efficient decomposable encoding for every \( f \in \text{NC}^1 \)
Low-Degree Encodings

• Low degree: \( g(x,r) = \text{vector of degree-d poly in } x,r \text{ over } F \)
  – aka “Randomizing Polynomials” [I-Kushilevitz 00,…]
  – Application: round-efficient MPC

• Motivating observation:
  Low-degree functions are easy to distribute!
  – Round complexity of MPC protocols [GMW87,BGW88,CCD88,…]
    • Semi-honest (passive) adversary:
      – \( t<n \) using ideal OT \( \Rightarrow O(\log d) \) rounds
      – \( t<n/d \) \( \Rightarrow 2 \) rounds
      – \( t<n/2 \) \( \Rightarrow \) multiplicative depth + 1 = \( \lceil \log d \rceil +1 \) rounds
    • Malicious (active) adversary:
      – Optimal \( t \) \( \Rightarrow O(\log d) \) rounds
Examples

• What’s wrong with previous examples?
  – Great degree in $x$ ($\deg_x=1$), bad degree in $r$

• Coming up:
  – Degree-3 encoding for every $f$
  – Efficient in size of branching program
Local Encoding

• Small output locality:
  
  
  ![Diagram showing small output locality]

  – Application: parallel cryptography!

• Coming up: encodings with output locality 4
  – degree 3, decomposable
  – efficient in size of branching program
Parallel Cryptography

How low can we get?

poly-time

NC

log-space

NC¹

AC⁰

NC⁰
Cryptography in $\text{NC}^0$?

- Real-life motivation: fast cryptographic hardware

- Tempting conjecture:
Surprising Positive Result \cite{AIK04}

Compile primitives in a “relatively high” complexity class (e.g., NC\(^1\), NL/poly, \(\oplus L/poly\)) into ones in \(\text{NC}^0\).

\(\text{NC}^1\) cryptography implied by factoring, discrete-log, lattices…

\(\Rightarrow\) essentially settles the existence of cryptography in \(\text{NC}^0\)
Remaining Challenge

How to encode “complex” \( f \) by \( g \in \mathbb{F} \)?

- **Observation**: enough to obtain const. degree encoding

- **Locality Reduction**:
  degree 3 poly over GF(2) \( \Rightarrow \) locality 4 rand. encoding

\[
\begin{align*}
  f(x) &= T_1(x) + T_2(x) + \ldots + T_k(x) \\
g(x,r,s) &= T_1(x)+r_1 + T_2(x)+r_2 + \ldots + T_k(x)+r_k \\
&= -r_1+s_1 -s_1-r_2+s_2 -s_{k-1}-r_k
\end{align*}
\]
3 Ways to Degree 3

1. Degree-3 encoding using a circuit representation

\[ f(x) = 1 \iff \exists y_1, y_2, y_3 \]

\[ y_1 = \text{NAND}(x_1, x_2) = x_1(1-x_2) + (1-x_1)x_2 + (1-x_1)(1-x_2) \]

\[ y_2 = \text{NAND}(x_3, x_4) \]

\[ y_3 = \text{NAND}(y_1, y_2) \]

\[ 1 = \text{NAND}(y_3, x_5) \]

Note: \[ \Rightarrow \exists! y_1, y_2, y_3 \]
Using circuit representation (contd.)

\[
\begin{align*}
q_1(x,y) &= 0 \\
q_2(x,y) &= 0 \\
& \vdots \\
q_s(x,y) &= 0
\end{align*}
\]

\[
g(x, y, r) = \sum r_i q_i(x, y) \}
\]

\[
f(x) = 0 \implies g(x, y, r) \text{ is uniform}
\]

\[
f(x) = 1 \implies g(x, y, r) \equiv 0 \text{ given } y=y_0, \text{ otherwise it is uniform}
\]

Statistical distance amplified to 1/2 by \(2^{\Theta(s)}\) repetitions.

- works over any field
- complexity exponential in circuit size
2. Degree-3 encoding using quadratic characters

Fact from number theory:

\[\forall N \ \forall \text{bit-sequence } b \in \{0,1\}^N \]

\[\exists \text{prime } q(=2^{O(N)}) \ \exists d > 0 \text{ such that } b = \chi_q(d)\chi_q(d+1)\cdots\chi_q(d+N-1)\]

• Let \(N=2^n, b = \text{length-}N \text{ truth-table of } f, \ F=\text{GF}(q)\)

• Define \(p(x_1,\ldots,x_n, r) = \left(d + \sum_{i=1}^{n} 2^{i-1}x_i\right) \cdot r^2\)

• one polynomial

• huge field size
3. Perfect Degree-3 Encoding from Branching Programs

BP = \((G, s, t, \text{edge-labeling})\) \hspace{1cm} G_x = \text{subgraph induced by } x

mod-\(q\) NBP: \(f(x) = \# s\text{-}t \text{ paths in } G_x \pmod{q}\)

- \text{size} = \# \text{ of vertices}
- \text{circuit-size} \leq \text{BP-size} \leq \text{formula-size}
- Boolean case: \(q=2\).
  - Captures complexity class \(\oplus L/poly\)
3. Perfect Degree-3 Encoding from Branching Programs

\[ BP = (G, s, t, \text{edge-labeling}) \]

\[ G_x = \text{subgraph induced by } x \]

- \( BP(x) = \text{det}(L(x)) \), where \( L \) is a degree-1 mapping which outputs matrices of a special form.

- Encoding:

\[ g(x, r_1, r_2) = R_1(r_1) \cdot L(x) \cdot R_2(r_2) \]
Perfect Degree-3 Encoding of BPs

BP = (G, s, t, edge-labeling)  

Gₓ = subgraph induced by x

Encoding based on Lemma:

\[ g(x, r₁, r₂) = R₁(r₁) \cdot L(x) \cdot R₂(r₂) \mod q \]

mod-q BP: \( f(x) = \# s \rightarrow t \) paths in \( Gₓ \mod q \).

Lemma: \( \exists \) degree-1 mapping \( L : x \rightarrow \) s.t. \( \det(L(x)) = f(x) \).

Correctness: \( f(x) = det \ g(x, r₁, r₂) \)

Privacy:

\[ g(x, r₁, r₂) \equiv \]

\[ det \ L(x) (= f(x)) \]
Proof of Lemma

Lemma: \( \exists \) degree-1 mapping \( L : x \rightarrow A(x) \) s.t. \( \det(L(x)) = f(x) \).

Proof:

\[ A(x) = \text{adjacency matrix of } G_x \text{ (over } F=GF(q)) \]

\[ A^* = I + A + A^2 + \ldots = (I - A)^{-1} \]

\[ A^*_{s,t} = (-1)^{s+t} \cdot \det(I - A)_{|t,s} / \det(I - A) \]

\[ = \det(A-I)_{|t,s} \]

\[ L(x) = (A(x) - I)_{|t,s} \]
Thm. size-$s$ BP $\Rightarrow$ degree 3 encoding of size $O(s^2)$

- perfect encoding for mod-$q$ BP (capturing $\oplus L$/poly for $q=2$)
- imperfect for nondeterministic BP (capturing $NL$/poly)
The secure evaluation of an arbitrary function can be reduced to the secure evaluation of degree-3 polynomials.

Round complexity of information-theoretic MPC in semi-honest model:

• How many rounds for maximal privacy? 3 rounds suffice
• How much privacy in 2 rounds? \( t < n/3 \) suffices

• perfect privacy + correctness
• complexity \( O(\text{BP-size}^2) \)
Is 3 minimal?

Thm. [IK00]
A boolean function $f$ admits a *perfectly private* degree-2 encoding over $F$ if and only if either:

• $f$ or its negation test for a linear condition $Ax=b$ over $F$;
• $f$ admits standard representation by a degree-2 polynomial over $F$. 
Wrapping Up

Composition Lemma:

\[ f \]  \hspace{1cm} g \text{ encodes } f  \hspace{1cm} h \text{ encodes } g

Concatenation Lemma:

\[ g^{(1)} \text{ encodes } f^{(1)} \]  \hspace{1cm} \cdots  \hspace{1cm} g^{(l)} \text{ encodes } f^{(l)} \hspace{1cm} g \text{ encodes } f
From Branching Programs to Locality 4

poly-size BPs

degree 3

NC$^0_4$

locality 4
Summary

• Different flavors of randomized encoding
  – Motivated by different applications

• “Simplest” encodings: outputs of form $x_i r_j r_k + r_h$
  – Efficient perfect/statistical encodings for various complexity classes (NC$^1$, NL/poly, mod$q$L/poly)
  – (Efficient computationally private encodings for all P, assuming “Easy PRG”.)
## Open Questions

<table>
<thead>
<tr>
<th>Randomized encoding</th>
<th>MPC</th>
<th>Parallel crypto</th>
</tr>
</thead>
<tbody>
<tr>
<td>poly-size NC⁰ encoding for every ( f \in P )?</td>
<td>Unconditionally secure constant-round protocols for every ( f \in P )?</td>
<td>( \exists \text{OWF} \Rightarrow \exists \text{OWF in NC}^0 )?</td>
</tr>
<tr>
<td>locality 3 for every ( f )?</td>
<td>maximal privacy with minimal interaction?</td>
<td>( \exists \text{OWF in NC}^1 \Rightarrow \exists \text{OWF in NC}^0_3 )?</td>
</tr>
<tr>
<td>better encodings?</td>
<td>better constant-round protocols?</td>
<td>practical hardware?</td>
</tr>
</tbody>
</table>