Crypto for PRAM from iO (via Succinct Garbled PRAM)

Kai-Min Chung
Academia Sinica, Taiwan

Joint work with:
Yu-Chi Chen, Sherman S.M. Chow, Russell W.F. Lai,
Wei-Kai Lin, Hong-Sheng Zhou
Computation in Cryptography

• Examples:
  – Multiparty Computation (MPC)
  – Non-interactive Zero Knowledge Proof (NIZK)
  – Fully Homomorphic Enc. (FHE)
  – Functional Encryption (FE)
  – Delegation with Persistent Database
  – Indistinguishability Obfuscation (iO)

• Traditionally, modeled as circuits

• Feasibility in more powerful computation model?
Models of Computation

• **Circuits**
  - Large description size
  - Parallelizable

AND, OR, NOT gates

• **Turing Machines**
  - Small description size

• **RAM Machines**
  - Random data access

• **Parallel RAM**
  - Random data access
  - Parallelizable
## Efficiency Gap

<table>
<thead>
<tr>
<th>Problem</th>
<th>Comp. Model</th>
<th>Total Time</th>
<th>Parallel Time</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Binary search</strong> (input size $n$)</td>
<td>Circuit</td>
<td>$\Omega (n)$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RAM</td>
<td>$O(\log n)$</td>
<td></td>
</tr>
<tr>
<td><strong>Sorting</strong></td>
<td>Circuit</td>
<td>$O(\log n)$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RAM</td>
<td>$\Omega (n \log n)$</td>
<td></td>
</tr>
<tr>
<td><strong>Keyword search/Range query</strong> (output size $m$)</td>
<td>Circuit</td>
<td>$\Omega (n)$</td>
<td>$O(\log n)$</td>
</tr>
<tr>
<td></td>
<td>RAM</td>
<td>$O(m \log n)$</td>
<td>$\Omega(m \log n)$</td>
</tr>
<tr>
<td></td>
<td>PRAM</td>
<td>$O(m \log n)$</td>
<td>$O(\log n)$</td>
</tr>
</tbody>
</table>
Parallel Model in Practice

• Emerging frameworks to handle big data
  – MapReduce, GraphLab, Spark, etc.

• Leverage *massive parallelism* & *random data access*
  – Circuit & RAM are not expressive enough

• **PRAM**: clean & expressive model to capture efficiency (total & parallel time & space) of these frameworks
Feasibility via Succinct Garbling

Succinct Garbling for Model X

- Delegation for X w/ Persistent DB
- MPC for X
- NIZK for X
- Functional Enc. for X
- iO for X

[GHRW14, CHJV15, BGLPT15, KLW15]
Succinct Garbling for Model X

\[ \Pi = (P, x) \rightarrow \text{Garb}(\Pi) \]

- **Succinctness**: \( \text{Time}(\text{Garb}(\Pi)) = \text{poly}(|\Pi|) \)
- **Eval Efficiency**: Complexity in Model X of \( \text{Eval}(\text{Garb}(\Pi)) \approx \text{Eval}(\Pi) \) (up to \( \text{polylog} \) overhead)
- **Security**: \( \Pi, \Pi' \) same complexity & output \( \Rightarrow \)

\[ \text{Garb}(\Pi) \approx \text{Garb}(\Pi') \]
Feasibility via Succinct Garbling

iO for circuit + OWF

\[\text{[KLW15]}\]

Succinct Garbling for \(X = TM\)

\[\text{[GHRW14, CHJV15, BGLPT15, KLW15]}\]

- Delegation for \(X\) w/ Persistent DB
- MPC for \(X\)
- NIZK for \(X\)
- Functional Enc. for \(X\)
- iO for \(X\)
Our Contribution

iO for circuit + OWF

Succinct Garbling for X = PRAM

[\text{GHRW14,CHJV15,BGLPT15,KLW15}]

Delegation for X w/ Persistent DB
MPC for X
NIZK for X
Functional Enc. for X
iO for X
Concurrent Work [CH16]

iO for circuit + OWF

Modular Proof

Succinct Garbling for $X = \text{RAM}$

[CH16, GHRW14, CHJV15, BGLPT15, KLW15]

Delegation for $X$ w/ Persistent DB
MPC for $X$
NIZK for $X$
Functional Enc. for $X$
iO for $X$
Succinct Garbling for TM [KLVW15]
Abstraction of [KLW15]

iO for circuit + OWF

- Authentication Step
  - ST-Garbling for TM
  - Same-Trace Garbling
  - Hiding Step
  - Succinct Garbling for TM
Same-Trace Garbling for TM/RAM

Computation Trace =
(initial-value),
(st₁, addr₁, val₁),
(st₂, addr₂, val₂),
(st₃, addr₃, val₃),
...
(stₜ₋₁, addrₜ₋₁, valₜ₋₁),
(stₜ, addrₜ, valₜ)

• Security: Π, Π’ *same trace* (so same inp/out, complexity) ⇒

Garb(Π) ≈ Garb(Π’)
Indistinguishability Obfuscation (iO)

- Scramble program to make it “unintelligible”

\[ P \rightarrow \mathcal{O}(P) \]

- Maintain functionality: \( \mathcal{O}(P)(x) = P(x) \ \forall \ x \)
- Security: If \( P(x) = P'(x) \ \forall \ x \) & same size \( \Rightarrow \)

\[ \mathcal{O}(P) \approx \mathcal{O}(P') \]
Abstraction of [KLW15]

iO for circuit + OWF

ST-Garbling for TM

Authentication Step

ST-Garb(P, x) = (iO(P_{auth}), x_{auth})

Only generate comp. trace of P(x)

Hiding Step

Garb(P, x) = (ST-Garb(P_{hide}, x_{hide}))

Hide memory/CPU state content & memory access pattern
Authentication & Hiding in [KLW15]

• Authentication step: \( ST-Garb(P, x) = (iO(P_{auth}), x_{auth}) \)
  – \( iO \)-friendly authentication primitives
  – Enable program switching step by step in hybrids
Authentication & Hiding in [KLVW15]

- Authentication step: $\text{ST-Garb}(P, x) = (\text{iO}(P_{\text{auth}}), x_{\text{auth}})$
  - iO-friendly authentication primitives
  - Enable program switching step by step in hybrids

- Hiding step: $\text{Garb}(P, x) = (\text{ST-Garb}(P_{\text{hide}}, x_{\text{hide}}))$
  - Hide content by encryption
  - Hide access pattern by Oblivious TM [PF79]
  - Allow erasing computation step by step in hybrids
Succinct Garbling for RAM
Challenge: Hiding Access Pattern

\[ \text{Garb}(P, x) = (\text{ST-Garb}(P_{\text{hide}}, x_{\text{hide}})) \]

- Replace Oblivious TM by Oblivious RAM [GO96]

- Issue: Cannot use ORAM security
  - ORAM is inherently randomized, security hold only when ORAM randomness is hidden

- Idea: “Puncturing” ORAM
Puncturing ORAM

- Use tree-based ORAM [SLSC11], which is “puncturable”
  - $t$-th step access pattern is determined by single randomness $r_t$
  - if $r_t$ is punctured/erased from program, $t$-th step access pattern can be simulated by random

- Puncturing $r_t$
  - $r_t$ may appear multiple times (encrypted) in history
  - Carefully erase $r_t$ backward in time step by step
    - Modify program: “erase $r_t$ after step $s$” for $s = t, t-1, \ldots, 0$
Puncturing ORAM

• Use tree-based ORAM [SLSC11], which is “puncturable”
  – t-th step access pattern is determined by single randomness r_t
  – if r_t is punctured/erased from program, t-th step access pattern can be simulated by random

• Puncturing r_t
  – r_t may appear multiple times (encrypted) in history
  – Carefully erase r_t backward in time step by step
    • Modify program: “erase r_t after step s”

[CH16]: “2 tracks trick” w/ modular & simpler proof
Succinct Garbling for PRAM
Challenge: Authenticate Memory

\[
\text{ST-Garb}(P, x) = (iO(P_{\text{auth}}), x_{\text{auth}})
\]

- Memory authenticated by “Merkle tree”
  - root stored in CPU state
  - Locally updatable by given augment path
- Issue: Parallel CPU $\implies$ Parallel Update
  - Require CPU-to-CPU communication
Challenge: Authenticate Memory

\[
\text{ST-Garb}(P, x) = (\text{iO}(P_{\text{auth}}), x_{\text{auth}})
\]

• Memory authenticated by “Merkle tree”
  – root stored in CPU state
  – Locally updatable by given augment path

• Issue: Parallel CPU \implies Parallel Update
  – Require CPU-to-CPU communication

• Issue: Cannot afford \(\Omega(m)\) overhead in parallel time
  – Otherwise, void the gain of parallelism
Parallel Update Problem

addr_2
aug-path_2
addr_3
aug-path_3
addr_1, addr_m
... 
addr_4
aug-path_4
Challenge: Authenticate Memory

\[ \text{ST-Garb}(P, x) = (iO(P_{\text{auth}}), x_{\text{auth}}) \]

• Memory authenticated by “Merkle tree”
  – root stored in CPU state
  – Locally updatable by given augment path

• Issue: Parallel CPU \implies Parallel Update
  – Require CPU-to-CPU communication

• Issue: cannot afford \( \Omega(m) \) overhead in parallel time
  – Otherwise, void the gain of parallelism

• \( O(\log^2 m) \)-round parallel algorithm
  – Parallel update level-by-level from leaves to root
Security Issue: High **Pebble** Complexity

Put "**pebble**" on node to switch program

Put **pebble** on node require to hardwire input/output
Security Issue: High Pebble Complexity

Can use $2m$ pebbles to traverse graph, but not better
⇒ Need to hardwire $\Omega(m)$ information in $P_{auth}$
⇒ $\text{poly}(m)$ overhead
Branch & Combine Emulation

Change topology to reduce pebble complexity
• Combine \( m \) CPU states to 1 combined state
• Branch one step computation from it
Branch & Combine Emulation

Change topology to reduce pebble complexity
- Combine $m$ CPU states to 1 combined state
- Branch one step computation from it

Claim: pebble complexity = $O(\log m)$
Branch & Combine Emulation

Change topology to reduce pebble complexity
- Combine $m$ CPU states to 1 combined state
- Branch one step computation from it

Claim: pebble complexity = $O(\log m)$

• Combine step
  - Build “Merkle tree” on CPU states
  - Combined state = root

• Branch step
  - Authentication & one step computation
Hiding Step for PRAM

Garb(P, x) = (ST-Garb(P_{hide}, x_{hide}))

• Replace ORAM by Oblivious PRAM \cite{BCP16}
  – also puncturable
Summary and Open Problems

• Feasibility of crypto for **PRAM** based on iO via succinct garbled **PRAM**

• Adaptive succinct garbled **(Parallel) RAM** with persistent memory (next talk) [ACC+15,CCHR15]

• Open: FHE for **RAM/PRAM**?

• Open: Crypto for **PRAM** without iO
  – ABE for **RAM/PRAM** based on LWE?

• Other parallel model?
Thank you! Questions?