SESSION ID: CRYP-F02

Cut-and-Choose for Garbled RAM

Peihan Miao
Research Scientist
Visa Research
Secure Two-Party Computation

$f(x, y)$
Secure Two-Party Computation for Circuits

C(x, y)
Yao’s Garbled Circuit [Yao’86]

\[ C(x, y) = \tilde{C}(\tilde{x}, \tilde{y}) \]

\( Input\ keys \)

\( \perp \)

\( \tilde{C} \)

\( \tilde{x} \)

\( \tilde{C} \)

\( \tilde{y} \)

\( y \)
Yao’s Garbled Circuit [Yao’86]

Input: 0 1 0 1 1

Garble

\[ C \]

\[ \tilde{C} \]
Yao’s Garbled Circuit [Yao’86]

Input: 0 1 0 1 1

C

Garble

\(C(01011)\)
Secure Two-Party Computation for Circuits
RAM (Random-Access Machine) Computation?

Program $P$

$P^D(x, y)$
Secure Two-Party RAM Computation

- Convert RAM program into a circuit?
  - RAM program with running time $T$
  - Turing machine with running time $O(T^3)$
  - Circuit with size $O(T^3 \log T)$
Secure Two-Party RAM Computation

- Convert RAM program into a circuit?

Circuit size could be *exponentially* larger than running time $T$!
Can we do it more efficiently?

Yes, Garbled RAM [LO’13]!

Secure RAM computation [LO’13, GHLORW’14, GLOS’15, GLO’15, GGMP’16, LO’17, KY’18, HY’19, CQ’19, ...]
Secure RAM Computation over Persistent Memory
Secure RAM Computation over Persistent Memory

\[ P_2^{D_1}(x_2, y_2) \]

\[ x_2, y_2 \]
Secure RAM Computation over Persistent Memory

\[ D_2 \]

\[ x_3 \rightarrow P_3 \rightarrow y_3 \]
Garbled RAM [LO’13]

\[\tilde{D} \quad \tilde{P}_1 \quad \tilde{x}_1, \tilde{y}_1\]

\[P_1^D (x_1, y_1) = \tilde{P}_1 \tilde{D} (\tilde{x}_1, \tilde{y}_1)\]
Garbled RAM [LO’13]

Size of garbled memory $\tilde{D} = \tilde{O}(|D|)$
Size and evaluation time of garbled program $\tilde{P} = \tilde{O}(T)$

$\tilde{O}$ ignores $\text{poly}(\lambda) \cdot \text{polylog}(|D|, T)$
Can we do it from the weakest cryptographic assumption?

Yes, black-box garbled RAM [GLO’15]!

black-box use of OWFs, but only semi-honest secure
Can we make it maliciously secure?

Yes, [GMW’87] compiler: semi-honest → malicious

requires generic zero-knowledge proofs, non-black-box use of OWFs
Can we make it maliciously secure while still making black-box use of OWFs?

Yes, this work!
Outline

• Secure Two-Party RAM Computation
  – Convert RAM program into a circuit?

• Garbled RAM [LO’13]

• Black-Box Garbled RAM [GLO’15]

• This Work: Malicious Security
  – Consistency Checks by Commitments
  – Cut-and-Choose on Gates
Black-Box Garbled RAM [GLO’15]
Black-Box Garbled RAM [GLO’15]
Malicious Alice?

\[ \tilde{D} \]

\[ \tilde{P} \]
Malicious Alice?

\[ \tilde{D} \]

\[ \tilde{P} \]

\[ \text{garbage} \]

\[ \text{label}_a^0 \]

\[ \text{label}_a^1 \]
How to avoid Alice cheating?

Cut-and-Choose!
Cut-and-Choose Technique
Cut-and-Choose for Yao’s Garbled Circuit [LP’07]

Open 2, 3, 5

Evaluate over the rest garbled circuits, take majority
Cut-and-Choose for Garbled RAM

Evaluate over the rest garbled RAMs, take majority
Cut-and-Choose for Garbled RAM

Evaluate over the rest garbled RAMs, take majority
Consistency

\[ \text{Circuit } X \]

\[ \text{Circuit } Y \]
Consistency

\[ \text{Circuit } X \]

\[ \text{Circuit } Y \]
Consistency

How to enforce Alice to provide $\text{label}_w^0$ without revealing the bit 0?

Circuit $X$

$\text{label}_a^0 = 0110$
$\text{label}_a^1$

Circuit $Y$

$\text{label}_w^0$
$\text{label}_w^1$
Consistency Check by Commitments

XOR-homomorphic commitment [FJNT'15]

\[ [a] \oplus [b] = [a \oplus b] \]
\[ [a] \oplus [a] = [0] \]
Consistency Check by Commitments

\[ [b_0], [b_1], [b_2], [b_3] \]

\( \text{label}_a^0 = b_0b_1b_2b_3 \)

\( \text{label}_a^1 \)

Circuit X

Circuit Y

\[ [\text{label}_w^0]||[0] \]

\[ [\text{label}_w^1]||[1] \]

\[ [\text{label}_w^r]||[r] \]

\[ [\text{label}_w^\bar{r}]||[\bar{r}] \]

Open \( \text{label}_w^r \)

\[ [b_0] \oplus [r] \rightarrow [0] \]?
Outline

- Secure Two-Party RAM Computation
  - Convert RAM program into a circuit?
- Garbled RAM [LO’13]
- Black-Box Garbled RAM [GLO’15]
- This Work: Malicious Security
  - Consistency Checks by Commitments
  - Cut-and-Choose on Gates
Cut-and-Choose on Circuits?

Open 2, 3, 5

Evaluate over the rest garbled circuits, take majority
How to guarantee that Alice has committed correctly?
Issue 2

$\text{label}^0_{a,1}$  
$\text{label}^1_{a,1}$

$\text{label}^0_{a,2}$  
$\text{label}^1_{a,2}$

$\text{label}^0_{a,3}$  
$\text{label}^1_{a,3}$

$X_1$  \hspace{1cm} $X_2$  \hspace{1cm} $X_3$

Circuit $Y$

$\text{label}^0_a$, $\text{label}^1_a$
Issue 2

Input size may grow $\textit{exponentially}$ in the number of circuits!
Cut-and-Choose on Gates [NO’09]

Soldering information

Open 2, 3, 5, 8, 11
Summary

- Secure Two-Party RAM Computation
  - Convert RAM program into a circuit?
- Garbled RAM [LO’13]
- Black-Box Garbled RAM [GLO’15]
- This Work: Malicious Security
  - Consistency Checks by Commitments
  - Cut-and-Choose on Gates
Thank you!