Secure Computation ORAM
The Case of 3-Party Computation

Stanislaw Jarecki, UC Irvine
Cryptography in the RAM Model Workshop,
Cambridge, MA,
June 2016

AC’15: Sky Faber, S.J., Sotirios Kentros, Boyang Wei
New Work: S.J., Boyang Wei
Secure Computation of (O)RAM Access (SC-ORAM)

SC-ORAM = Sec. Comp of $F_{ORAM}$: sharing of $D, x \rightarrow$ sharing of $D[x]$

(for $\text{<write>}$: additional shared input $v$ and $D \rightarrow D'$ s.t. $D'[x] = v$)
Secure Computation of (O)RAM Access (SC-ORAM)

Application: n-Server Private Database (≈ n-Server SPIR)

SC-ORAM = Sec.Comp of $F_{ORAM}$: sharing of $D, x \rightarrow$ sharing of $D[x]$

Diagram:
- $D_1, D_2, \ldots, D_n$
- $X_1, X_2, \ldots, X_n$
- $S_1, S_2, \ldots, S_n$
- Client
- $X \rightarrow s = D[x]$
Secure Computation of (O)RAM Access (SC-ORAM)

Application: Sec. Comp. of RAM Program [OS’97, DMN’11, GKKKMRV’12]

SC-ORAM = Sec. Comp of $F_{ORAM}$: sharing of $D, x$ $\rightarrow$ sharing of $D[x]$

Each instruction computed by standard MPC (Yao, BGW, ...)

Sec. Comp. of RAM programs with polylog($|D|$) overhead
Generic SC-ORAM Construction [OS’97, GKKKMVR’12]
ORAM Scheme + Secure Comp. of Client’s Code → SC-ORAM

ORAM:

Client
MK, x

Server
Enc_{MK}(D; r)

D[x]

SC-ORAM:

Client_{1}
MK_{1}, x_{1}

2PC

Client_{2}
MK_{2}, x_{2}

Enc_{MK}(D; r)

Server

D[x] = v_{1} \oplus v_{2}
Generic SC-ORAM Construction \cite{OS'97,GKKKMRV'12}

ORAM Scheme + Secure Comp. of Client’s Code

For efficient MPC of RAM programs we need ORAM whose Client is “Secure-Computation Friendly”

\[D[x] = v_1 \oplus v_2\]

\[\text{[GKKKMRV'12a]}: \ GO'96 \ ORAM \ + \ Yao \ + \ PK-based \ SS-OPRF \ gadget\]

\[\text{[GKKKMRV'12b]}: \ Path-ORAM \ [\text{Shi+’11}] \ + \ Yao\]

\[\text{[WHCSS’14]}: \ Path-ORAM \ modified \ + \ Yao \ \Rightarrow \ small \ circuits\]
Our Question:
Could SC-ORAM be faster given 3 players with honest majority?

3 Parties = 2 Parties with correlated randomness

Example: Oblivious Transfer with Precomputation [Bea’95]

2 Parties: OT needs PK crypto ops [IR’89]
3 Parties: OT costs 4 xor’s
SC for Path-ORAM [Shi+’11]
Path-ORAM Access: Recursive Tree+Array Lookup

Split address space of m bits, h chunks of $\tau=m/h$ bits

\[ N = [N_1 \mid N_2 \mid \ldots \mid N_h] \]

$T_i$ is a binary tree of depth $d_i = i \cdot \tau$, tree nodes are buckets of size $w$

ORAM = $(T_0, T_1, T_2, \ldots, T_h)$
SC for Path-ORAM [Shi+’11]
Path-ORAM Access: Recursive Tree+Array Lookup

Split address space of m bits, h chunks of $\tau = m/h$ bits
$N = [N_1 \mid N_2 \mid \ldots \mid N_h]$

$T_i$ is a binary tree of depth $d_i = i \cdot \tau$, tree nodes are buckets of size $w$

Client’s code is a sequence of *array* or *dictionary* list look-ups...

---

Server

Client

$T_0$ $T_1$ $T_2$ $\ldots$ $T_h$

$L_1$ $L_2$ $L_3$

$N_1$ $N_2$ $\ldots$ $N_h$

$T_i$ is a binary tree of depth $d_i = i \cdot \tau$, tree nodes are buckets of size $w$
SC for Path-ORAM [Shi+’11]
The other half: Path-ORAM Eviction

Eviction: 1) put the (modified) retrieved entry on top  
2) move all* entries down towards their targets labels

SC-ORAM: To reduce circuit size, use **constrained eviction strategy**
SC for Tree-ORAM
Three Steps

Access: Retrieve data assoc. with searched-for address N
SS[ X, N ] → d s.t. (N,d) ∈ X

Eviction.1: Compute movement logic, T : [n] → [n]
SS[ X_\mid N ] → SS[ T ]

Eviction.2: Permute path X according to T
SS[ X, T ] → SS[ T(X) ] s.t. T(X) = X_{T(1)},...,X_{T(n)}

2PC-ORAM costs: online, passive adv (last tree, w/o small constants)

bndw: |X|·k
comp: |C_A|+|C_T|+|C_M| ciphers (+ k OT’s)

X = (X_1,...,X_n) : tree path
k: sec.par.
3PC for Tree-ORAM

Access Step: \( \text{SS}[X, N] \rightarrow d \) s.t. \((N, d) \in X\)

Client’s code is a sequence of array look-ups...

3PC idea:  
- secret-share all data (T_i’s and N) between \( P_1 \) & \( P_2 \)
- send matching entry to \( P_3 \) via *Conditional SS-OT*

\[
\begin{align*}
\begin{array}{c|c}
N_1^*, & \\
\vdots & \\
N_i^1 & d_1^1 \\
\vdots & \\
N_n & d_n \\
\end{array}
\quad & \quad \quad
\begin{array}{c|c}
N^*, & \\
\vdots & \\
N_i & d_i \\
\vdots & \\
N_n & d_n \\
\end{array}
\quad & \quad \quad
\begin{array}{c|c}
N_2^*, & \\
\vdots & \\
N_i^2 & d_i^2 \\
\vdots & \\
N_n & d_n \\
\end{array}
\end{align*}
\]

\[
\begin{align*}
P_3 & \rightarrow d_i^1 \oplus d_i^2 \quad \text{for } i \text{ s.t. } N_i^1 \oplus N_1^* = N_i^2 \oplus N_2^*
\end{align*}
\]
### 3PC for Tree-ORAM

**Access Step:** \( SS[ X, N ] \rightarrow d \) s.t. \((N,d) \in X\)

<table>
<thead>
<tr>
<th>(P_1)</th>
<th>(N_1^*, \cdots )</th>
</tr>
</thead>
<tbody>
<tr>
<td>(N_i^1, d_i^1)</td>
<td>(\vdots, \vdots)</td>
</tr>
<tr>
<td>(\vdots, \vdots)</td>
<td>(\vdots, \vdots)</td>
</tr>
</tbody>
</table>

- **String Equality Problem:**

<table>
<thead>
<tr>
<th>(P_2)</th>
<th>(\vdots, \vdots)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(N_i^2, d_i^2)</td>
<td>(\vdots, \vdots)</td>
</tr>
<tr>
<td>(\vdots, \vdots)</td>
<td>(\vdots, \vdots)</td>
</tr>
</tbody>
</table>

**2PC, Yao’s GC:** \( knD \) bndw (+k exp’s)

**2PC, arith.circ.:** bndw--, rounds++

**2PC, DH-KE:** \( n \) exp’s

**3PC:** *Conditional Disclosure of Secrets*

- [GIKM00], IT: \( 4nD \) bndw
- [AC’15], crypt: \( 2n(m+D) \) bndw \( \approx 2x \) plain Client-Server ORAM

**3PC, 2-PIR:** +1 round, \( 2nm+\sqrt{n}D \) bnds

**k:** sec.par.

**n:** # tuples in path

**D:** record size

**m:** address size
3PC for Tree-ORAM

Problem: $P_3$ learns position $i$ where the $a^i = b^i$ match occurs...

3PC sol.: $P_1$ & $P_2$ shift their input lists by (the same) random offset $P_3$ can learn a pointer into the shifted list ($= \text{random in } [n]$)

$P_1$ & $P_2$ hold same PRF key $k$

\[
\begin{align*}
&[A^i, C^i] = \text{PRF}_k(a^i) \\
&\{[A^i, d_{i1} \oplus C^i]\}_i \\
&[B^i, D^i] = \text{PRF}_k(b^i) \\
&\{[B^i, d_{i2} \oplus D^i]\}_i \\
&d_{i1} \oplus d_{i2} \text{ for } i \text{ s.t. } a^i = b^i
\end{align*}
\]
Path-ORAM: from 2PC to 3PC
Access Step via CDS a.k.a. SS-COT

Access ($C_A$): $SS[ X, N ] \rightarrow d$ s.t. $(N,d) \in X$

Ev.1 ($C_T$): $SS[ X|N ] \rightarrow SS[ T ]$

Ev.2 ($C_M$): $SS[ X, T ] \rightarrow SS[ T(X) ]$ s.t. $T(X) = X_{T(1)}, \ldots, X_{T(n)}$

2PC-ORAM

Acc: $bndw: |X| \cdot k + ciph: |C_A| + OT's$

Ev.1: $ciph: |C_T|$

Ev.2: $ciph: |C_M|$

3PC-ORAM

2PC-ORAM

Acc: $bndw: |X|$

Ev.1: $ciph: |C_T|$

Ev.2: $ciph: |C_M|$

- 100x cheaper access
- Benefits:
  - response time (eviction in background)
  - access inherently sequential
  - batch access with postponed eviction

$X = (X_1, \ldots, X_n)$: tree path
$k$: sec.par.
3PC for Tree-ORAM
Eviction Steps

Ev.1 ($C_T$): $SS[ X_{|N} ] \rightarrow SS[ T ]$
Ev.2 ($C_M$): $SS[ X, T ] \rightarrow SS[ T(X) ]$ s.t. $T(X) = X_{T(1)}, \ldots, X_{T(n)}$

3PC idea: Use Yao for $C_T$, but make transition table $T$ “uniform” s.t.:
Ev.1: If $P_1$ and $P_2$ locally permute secret-shared list $X$ then $P_3$ can learn $T$ in the clear
Ev.2: $SS[ X, T ] \rightarrow SS[ T(X) ]$ is a simple variant of OT

Uniform Transition Table $T$ [AC’15]:
- $T$ moves 2 items from node $i$ to $i+1$
- $P_1$ & $P_2$ shuffle each bucket
- $P_3$ learns 2 first movable/empty items after 2 random shifts
Path-ORAM: from 2PC to 3PC.

Access (SS-COT), Ev.1 (Yao), Ev.2 (SS-OT)

Access (C_A): \( \text{SS}[X, N] \rightarrow d \) s.t. \((N, d) \in X\)

Ev.1 (C_T): \( \text{SS}[X|N] \rightarrow \text{SS}[T] \)

Ev.2 (C_M): \( \text{SS}[X, T] \rightarrow \text{SS}[T(X)] \) s.t. \( T(X) = X_{T(1)}, \ldots, X_{T(n)} \)

2PC-ORAM

Acc: \( \text{bndw}: |X| \cdot k + \text{ciph}: |C_A| + \text{OT’s} \)

Ev.1: \( \text{ciph}: |C_T| \)

Ev.2: \( \text{ciph}: |C_M| \)

\( \text{bndw}: |X| \cdot k = n(m+|d|)k \approx m^2w \cdot \alpha k \)

\( \text{ciph}: m^2w \cdot (\alpha + \alpha_{CT} + \alpha \cdot \alpha_{CM}) + \text{OT’s} \)

3PC-ORAM

\( \text{bndw}: |X| \)

\( \text{ciph}: |C_T| + \text{bndw}: nm \cdot k \)

\( \text{bndw}: |X| \)

\( m^2w \cdot (\alpha + k) \)

\( m^2w \cdot \alpha_{CT} \)

---

\( X = (X_1, \ldots, X_n) \): tree path
\( n = m \cdot w \)
\( \alpha = \max(2^T, D/m) \)
\( |d| \approx m \cdot \alpha \)

\( X_i = (\text{addr.}, \text{data}) \)
\( k : \text{sec.par.} \)
\( m : \text{address size} \)
\( w : \text{bucket width} \)
\( \tau : \text{addr. chunk size} \)
\( D : \text{record size} \)
\( \alpha_{CT}, \alpha_{CM} : \text{circ.comp. of } C_T, C_M \) (=circuit size / input length)
Path-ORAM: from 2PC to 3PC
Access (SS-COT), Ev.1 (Yao), Ev.2 (SS-OT)

<table>
<thead>
<tr>
<th>2PC-ORAM</th>
<th>3PC-ORAM</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Acc:</strong> bndw:</td>
<td>X</td>
</tr>
<tr>
<td><strong>Ev.1:</strong> ciph:</td>
<td>C_T</td>
</tr>
<tr>
<td><strong>Ev.2:</strong> ciph:</td>
<td>C_M</td>
</tr>
</tbody>
</table>

\[
\text{bndw: } |X|k = n(m+|d|)k \approx m^2 w \cdot \alpha k
\]
\[
\text{ciph: } m^2 w \cdot (\alpha + \alpha_{CT} + \alpha \cdot \alpha_{CM})
\]

\[
\text{AC’15: 3PC with simplistic eviction: very low } \alpha_{CT}, w=O(m+k) \approx 100
\]
\[
\text{WCS’15: “Circuit-ORAM”: 2PC, greedy eviction, higher } \alpha_{CT}, w=3, \alpha_{CM}=2
\]
\[
\text{New work: 2PC with same eviction as in Circuit-ORAM, slightly higher } \alpha_{CT}
\]

\[
X = (X_1,...,X_n) : \text{tree path}
\]
\[
X_i = (\text{addr.}, \text{data})
\]
\[
k : \text{sec.par.}
\]
\[
n = m \cdot w
\]
\[
m : \text{address size}
\]
\[
w : \text{bucket width}
\]
\[
\alpha = \max(2^\tau, D/m)
\]
\[
\tau : \text{addr. chunk size}
\]
\[
D : \text{record size}
\]
\[
|d| \approx m \cdot \alpha
\]
\[
\alpha_{CT}, \alpha_{CM} : \text{circ.comp. of } C_T, C_M (=\text{circuit size / input length})
\]
Circuit-ORAM Eviction [WCS’15]
From 2PC to 3PC : Making Transition Table $T$ Uniform

Circuit ORAM Eviction:
greedy: “deepest goes first”

Making it Uniform:
1. Fill-in jumps so $T$ is a cycle
2. Reveal $(\Pi \circ T)(i)$ instead of $T(i)$ for $\Pi$
   - permute outside Garb.Circ.
   - +2 rounds for (de-)mask/permute
**Path-ORAM: 2PC vs. 3PC**

- **2PC-ORAM:**
  - $bndw = m^2w \cdot \alpha k$
  - $|circ| = m^2w \cdot (\alpha + \alpha_{CT} + \alpha \cdot \alpha_{CM})$
  - $k$: sec.par (=128)
  - $m$: address size
  - $w$: bucket width (=3)
  - $\alpha = \max(2^\tau, D/m) = 2^\tau$
  - $\tau$: addr. chunk size
  - $D$: record size (=4B)

- **3PC-ORAM:**
  - $bndw = m^2w \cdot (\alpha + k)$
  - $|circ| = m^2w \cdot \alpha_{CT}$
  - $\alpha_{CT}$ (=?) $\alpha_{CM}$ (=2): circ.comp. of $C_T, C_M$ (=circuit size / input length)

---

**Online Bandwidth**

- CORAM, $\tau=3$
- 3PORAM, $\tau=6, w=128$
- 3PCORAM, $\tau=3$
- 3PCORAM, $\tau=6$

**3PC: ~10X**

---

**CORAM:** 2PC [WCS'15]:
- higher $\alpha_{CT}$, $w=3$, $\alpha_{CM}=2$

**3PORAM:** 3PC [AC'15]:
- low $\alpha_{CT}$, $w=O(m+k) \leq 128$

**3PCORAM:** 3PC [new]:
- same $\alpha_{CT}$ (~1.2x) and $w$ as in CORAM
Path-ORAM: 2PC vs. 3PC

2PC-ORAM: $	ext{bndw} = m^2w \cdot \alpha k$

3PC-ORAM: $|\text{circ}| = m^2w \cdot (\alpha + \alpha_{CT} + \alpha \cdot \alpha_{CM})$

- $k$: sec.par (=128).
- $m$: address size
- $w$: bucket width (=3)
- $\alpha = \max(2^T, D/m) = 2^T$
- $\tau$: addr. chunk size
- $D$: record size (=4B)
- $\alpha_{CT}$ (=?) $\alpha_{CM}$ (=2): circ.comp. of $C_T, C_M$ (=circuit size / input length)

### Garbled Circuit Size

**CORAM**: 2PC [WCS'15]: higher $\alpha_{CT}$, $w=3$, $\alpha_{CM}=2$

**3PORAM**: 3PC [AC'15]: low $\alpha_{CT}$, $w=O(m+k) \leq 128$

**3PCORAM**: 3PC [new]: same $\alpha_{CT}$ (~1.2x) and $w$ as in CORAM

3PC: 5-10x
Path-ORAM: 2PC vs. 3PC

2PC-ORAM: $\text{bndw} = m^2 w \cdot \alpha k$

$|\text{circ}| = m^2 w \cdot (\alpha + \alpha_{CT} + \alpha_{CM})$

3PC-ORAM: $\text{bndw} = m^2 w \cdot (\alpha + k)$

$|\text{circ}| = m^2 w \cdot \alpha_{CT}$

$k$: sec.par ($=128$).

$m$: address size

$w$: bucket width ($=3$)

$\alpha = \max(2^T, D/m) = 2^T$

$\tau$: addr. chunk size

$D$: record size ($=4B$)

$\alpha_{CT} (=?)$ $\alpha_{CM} (=2)$: circ.comp. of $C_T, C_M$ (=circuit size / input length)

**Online End-to-end Wallclock Time**

3PC:
- Larger $\tau$ → 2x
  - 1.5x for CPU
  - 2-3x in rounds
- Pipelining → 2x

3PCORAM: 2PC [WCS’15]: higher $\alpha_{CT}$, $w=3$, $\alpha_{CM}=2$

3PORAM: 3PC [AC’15]: low $\alpha_{CT}$, $w=O(m+k) \leq 128$

3PCORAM: 3PC [new]: same $\alpha_{CT}$ (~1.2x) and $w$ as in CORAM

$m$: address size
Questions, Directions

Examples:

- pipelining, batched access with postponed eviction, parallel access
- MPC for other data-structures
- general (t,n): the “P₁/P₂ permute & P₃ gets outputs” idea doesn’t scale...
- malicious security? covert security?
- secure-computation-friendly multi-server ORAM ([LO’14]: client uses PRF)