Whitewash: Outsourcing Garbled Circuit Generation for Mobile Devices

Annual Computer Security Applications Conference 2014

Henry “Hank” Carter, Charles Lever, Patrick Traynor
SMC on mobile devices

- Mobile devices loaded with private and context-sensitive information and applications that use this information
- Secure Multiparty Computation (SMC) allows for computation over encrypted inputs
- Highly constrained system resources (memory, power, processing, communication)
Why don’t we have mobile SMC?

- The dominant two-party construction, garbled circuits, requires too much memory, processing power, and bandwidth
- Special purpose protocols can be optimized, but no efficient general purpose techniques
- Wish: an efficient mobile two-party SMC protocol that generalizes to any function
Head in the clouds

- Given a technique for performing SMC between servers, can we outsource expensive operations to the cloud?
- How trustworthy is the cloud?
- Secure outsourcing requires mechanisms for
  - Hiding inputs and outputs
  - Ensuring the cloud follows the protocol
Background: Garbled Circuits

- Two-party protocol for jointly evaluating a function without revealing inputs

- Important facts:
  - Represents functions as logical circuits
  - Requires interactivity (garbling vs. evaluating)
  - The basic construction is secure against semi-honest adversaries
Background: Oblivious Transfer

- Allows one party to choose a data value held by a second party without the second party knowing which item was selected (i.e., not-magic trick)

- Important facts:
  - Use specialized public-key cryptography
  - Consume bandwidth
Setting

• A limited mobile device communicating with an application server. The mobile also has access to a cloud service.

• Goal: The mobile device and application server securely compute a two-party function using garbled circuits.

• Security:
  ‣ Preserve input and output privacy from both the other party and the cloud
  ‣ Security in the malicious setting
Previous work

• Salus Framework (Kamara et al., CCS 2012)
  ‣ First garbled circuit outsourcing scheme
  ‣ Provides malicious and covert secure protocols for outsourcing

• CMTB Outsourcing (USENIX Security 2013)
  ‣ Used outsourced oblivious transfer (OOT) to deliver garbled inputs of phone to the evaluating cloud
  ‣ Phone performs some checks to ensure that the cloud doesn’t “lazily” check
Can we do better?

- **OT on the phone**
  - Reduced, but slow. Bottleneck for parallelization

- **Consistency checks**
  - Ensure cloud is behaving, but require exponentiations
  - Shown to be very slow on mobile devices

- **Restricted collusion model**

- **Can we improve on these techniques?**
Whitewash

- Consider reversing outsourced party (i.e., outsource generation instead of evaluation)
- Have the mobile device produce random seeds, Cloud generates garbled circuits
- Standard OT/evaluation between app server and Cloud to garble the server’s inputs
Whitewash

- Built on two garbled circuit advances:
  - Shelat-Shen (CCS ’13) Uses only symmetric-key operations
  - PCF (Kreuter et al. USENIX ’13) compiles smaller circuits with compact program representations
- Improved efficiency for both mobile and servers
- Improved Security for certain types of collusion
- Protocol takes place in 6 phases
Phase 1: Parameter Setup

Random Seeds
Phase 1: Parameter Setup
Phase 2: Input Commitment
Phase 2: Input Commitment
Phase 2: Input Commitment
Phase 3: Circuit Setup
Phase 4: Oblivious Transfers
Phase 5: Evaluation

Pipeline
Phase 6: output proof and release
Phase 6: output proof and release
What have we gained?

• No OT on the phone
  ‣ Mobile device generates randomness and input wire labels, so it can garble its own input

• No consistency check on the phone
  ‣ Significantly reduces the number of group algebraic ops required on the device

• Stronger collusion resistance
  ‣ Secure when mobile and cloud collude

• In exchange: randomness generation
  ‣ Can be done a priori to save time.
Collusion Assumptions

- Kamara et al. notes that an outsourcing scheme with collusion implies an SFE scheme where one party performs sub-linear work w.r.t. circuit size.

- Previous work assumes NO collusion with the cloud

- Whitewash reduces to shelat-Shen ’13 when Bob and Cloud collude
  - Loss of fair release
  - Remains malicious secure

- Realistic scenario: cloud service may collude with the customer
Evaluation

• Server setup
  ‣ 64 core, 1 TB memory
  ‣ 802.11g wireless connection
  ‣ Samsung Nexus One

• Test circuits
  ‣ Hamming Distance
  ‣ Matrix-Multiplication
  ‣ RSA

• Comparison against KsS, CMTB
Improvement: execution time

Hamming Dist

![Graph showing time (sec) vs. Circuit for Hamming 1600 and Hamming 16384](image)
Improvement: execution time

Hamming Dist

Time (sec)

Circuit

Hamming 1600
Hamming 16384

96%

WW
CMTB
KSS

Georgia Tech Information Security Center (GTISC)
Improvement: execution time

Matrix-Mult

![Graph showing execution time for different matrix sizes (3x3, 5x5, 8x8, 16x16) with two methods: WW and CMTB. The x-axis represents the circuit size, and the y-axis represents time (sec).]
Improvement: execution time

Matrix-Mult

- 3x3: WW 73% CMTB 98%
- 5x5: WW 98% CMTB 98%
- 8x8: WW 98% CMTB 98%
- 16x16: WW 98% CMTB 98%
Improvement: execution time

Matrix-Mult 3x3

- CHKS
- MOBI
- EVL
- OT

Time (sec)

WW

CMTB
## Improvement: Bandwidth

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Bandwidth (MB)</th>
<th>Reduction Over</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>WW</td>
<td>CMTB</td>
<td>KSS</td>
<td>CMTB</td>
</tr>
<tr>
<td>Hamming (1600)</td>
<td>23.56</td>
<td>41.05</td>
<td>240.33</td>
<td>42.62%</td>
</tr>
<tr>
<td>Hamming (16384)</td>
<td>241.02</td>
<td>374.03</td>
<td>x</td>
<td>35.56%</td>
</tr>
<tr>
<td>Matrix (3x3)</td>
<td>4.26</td>
<td>11.50</td>
<td>x</td>
<td>62.97%</td>
</tr>
<tr>
<td>Matrix (5x5)</td>
<td>11.79</td>
<td>23.04</td>
<td>x</td>
<td>48.82%</td>
</tr>
<tr>
<td>Matrix (8x8)</td>
<td>30.15</td>
<td>51.14</td>
<td>x</td>
<td>41.05%</td>
</tr>
<tr>
<td>Matrix (16x16)</td>
<td>120.52</td>
<td>189.52</td>
<td>x</td>
<td>36.41%</td>
</tr>
<tr>
<td>RSA-256</td>
<td>3.97</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>
# Improvement: Bandwidth

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Bandwidth (MB)</th>
<th>Reduction Over</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>WW</td>
<td>CMTB</td>
</tr>
<tr>
<td>Hamming (1600)</td>
<td>23.56</td>
<td>41.05</td>
</tr>
<tr>
<td>Hamming (16384)</td>
<td>241.02</td>
<td>374.03</td>
</tr>
<tr>
<td>Matrix (3x3)</td>
<td>4.26</td>
<td>11.50</td>
</tr>
<tr>
<td>Matrix (5x5)</td>
<td>11.79</td>
<td>23.04</td>
</tr>
<tr>
<td>Matrix (8x8)</td>
<td>30.15</td>
<td>51.14</td>
</tr>
<tr>
<td>Matrix (16x16)</td>
<td>120.52</td>
<td>189.52</td>
</tr>
<tr>
<td>RSA-256</td>
<td>3.97</td>
<td>x</td>
</tr>
</tbody>
</table>
Summary

• New protocol for outsourcing garbled circuit generation
• Removes OT and public key operations performed on the mobile device
• Remains secure under certain collusion scenarios
• Performance evaluations show up to 98% improvement in evaluation time and 63% improvement in bandwidth usage
Questions?

Thanks for your attention!

carterh@gatech.edu