Secure Multi-party Computation
Protocols from a High-Level Programming Language

Sander Siim
Supervisors: Dan Bogdanov, Sven Laur

CS Theory Days
Narva-Jõesuu, 16.05.2014
What we’ll cover

• Secure multi-party computation
  – One technique – Yao’s garbled circuits
  – Another – secret sharing
• State-of-the-art Boolean circuit compilers
• A hybrid protocol
• Implementation benchmarks
Secure multi-party computation (SMC)

- A subfield of cryptography
- The goal: *enable multiple parties to compute functions on their joint inputs while preserving the privacy of those inputs*
- Enables computations with sensitive data
- Useful in modern applications
Classic example

• Millionaire’s problem (Andrew C. Yao 1982)
• Two rich guys want to know who is richer without revealing their net worth
• Need to compute $f(x,y) = x > y$ securely
• Two-party secure computation
Yao’s garbled circuits

- A solution to two-party secure computation problem
- A seminal SMC method
- Enables secure evaluation of arbitrary Boolean functions
Yao’s garbled circuits

- Represent function as Boolean circuit

\[ z = a \geq b \]
Yao’s garbled circuits

- Two parties
- **Garbler** encrypts circuit’s truth tables with random tokens and sends them to **evaluator**
- **Evaluator** can obliviously evaluate circuit using input tokens
- Oblivious transfer for sending tokens
- Evaluator does not learn input nor intermediate results!
Yao’s garbled circuits

<table>
<thead>
<tr>
<th>$a$</th>
<th>$b$</th>
<th>$z = a \land b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Yao’s garbled circuits

<table>
<thead>
<tr>
<th>$a$</th>
<th>$b$</th>
<th>$z = a \land b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$a$</th>
<th>$b$</th>
<th>$z = a \land b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_a^0$</td>
<td>$X_b^0$</td>
<td>$X_z^0$</td>
</tr>
<tr>
<td>$X_a^0$</td>
<td>$X_b^1$</td>
<td>$X_z^0$</td>
</tr>
<tr>
<td>$X_a^1$</td>
<td>$X_b^0$</td>
<td>$X_z^0$</td>
</tr>
<tr>
<td>$X_a^1$</td>
<td>$X_b^1$</td>
<td>$X_z^1$</td>
</tr>
</tbody>
</table>
Yao’s garbled circuits

<table>
<thead>
<tr>
<th>$a$</th>
<th>$b$</th>
<th>$z = a \land b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_a^0$</td>
<td>$X_b^0$</td>
<td>$X_z^0$</td>
</tr>
<tr>
<td>$X_a^0$</td>
<td>$X_b^1$</td>
<td>$X_z^0$</td>
</tr>
<tr>
<td>$X_a^1$</td>
<td>$X_b^0$</td>
<td>$X_z^0$</td>
</tr>
<tr>
<td>$X_a^1$</td>
<td>$X_b^1$</td>
<td>$X_z^1$</td>
</tr>
</tbody>
</table>
Yao’s garbled circuits

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>$b$</td>
<td>$z = a \land b$</td>
</tr>
<tr>
<td>$X_0^a$</td>
<td>$X_0^b$</td>
<td>$X_0^z$</td>
</tr>
<tr>
<td>$X_0^a$</td>
<td>$X_1^b$</td>
<td>$X_0^z$</td>
</tr>
<tr>
<td>$X_1^a$</td>
<td>$X_0^b$</td>
<td>$X_0^z$</td>
</tr>
<tr>
<td>$X_1^a$</td>
<td>$X_1^b$</td>
<td>$X_1^z$</td>
</tr>
</tbody>
</table>

\[ a \text{ and } b \text{ are inputs, } z \text{ is the output.} \]
Yao’s garbled circuits

- Yao’s computational overhead is quite high
- Main problem – where to get good circuits?
- Size and structure of circuit is important
Secret sharing

- Cryptographic method for sharing a secret between $n$ parties
- Individual shares are random
- Secret recovered by combining the shares
- Many possible implementations
Secret sharing SMC protocol

$IP$ secret-sharing the input $x$ $x_1$ $x_n$ $CP_1$ secure distributed computations $CP_n$ $y = f(x)$ $RP$ constructing the output $y_1$ $y_n$
Secret sharing

- Additive secret sharing scheme
- Fast ring operations on $\mathbb{Z}_{2^k}$
- Cumbersome to do bit-level manipulation
- Yao’s GC is better at this
Circuit compilers

- Problem: how to build circuits for SMC?
- Two state-of-the-art compilers from C
- Designed for SMC applications
CBMC-GC

• Based on CBMC bit-precise model checker for C – produces Boolean formula
• Transforms C source directly to full circuit representation
• Loops are unrolled – requires bound for termination
• Highly optimized circuits using SAT-solvers
PCF – Portable Circuit Format

• Compiles LCC bytecode to a stack-like circuit representation
• Loops are not unrolled – subroutines, conditional jumps
• Only hold active set of wires in memory
• Black-box PCF interpreter to evaluate circuits – emits gates one-by-one
Empirical evaluation

• PCF representation is very memory-efficient
• CBMC-GC supports different bit-length variables
• CBMC-GC produces smaller circuits
• PCF compiles orders of magnitude faster
Sharemind

- A general purpose privacy-preserving computational platform
- **additive3pp** protection domain
- 3-out-of-3 additive secret sharing scheme
- Secure against passive adversary
- Implements most-used primitive operations
Hybrid protocol

- Based on Yao GC
- Both input and output are secret-shared between computational parties
- Oblivious transfer uses secret sharing protocols
- Bellare et al. garbling scheme*
- Security against computationally bounded adversary

Hybrid protocol

• Continuation of Oleg Šelajev MSc thesis
• An improved protocol implementation
  – Works seamlessly with Sharemind secret-shared types
  – Much faster C++ implementation
  – Cryptographic optimizations to circuit garbling and evaluation

Hybrid protocol

- $C_P_1$ generates $X^0, X^1$
- $C_P_2, C_P_3$ receive $x_2, x_3$
- Oblivious transfer protocol
  - $C_P_1$ garbles $C$ and learns $y'_1$
  - $C_P_2$ evaluates $C$ and learns $y'_2$
- Resharing protocol
  - $C_P_1$ receives $y_1$
  - $C_P_2$ receives $y_2$
  - $C_P_3$ receives $y_3$
Performance

- As usual for Yao, performance is linear to the number of non-XOR gates
- Our protocol evaluates a circuit with 350k non-XOR gates in 1 second
Comparing with secret sharing

- AES-128
- S. Laur, R. Talviste, J. Willemson. 2013
  - AES using only secret sharing
  - 652 ms single operation, 0.37 ms amortised cost
- Our protocol
  - 127 ms single operation

Sven Laur, Riivo Talviste, Jan Willemson: From Oblivious AES to Efficient and Secure Database Join in the Multiparty Setting. ACNS 2013: 84-101
Comparing with Yao GC

- 32-bit multiplication
- Performance of Huang et al. framework presented by CBMC-GC authors
  - Garbling and evaluation – 127 ms
- Our protocol
  - 30.2 ms for garbling and evaluation


Conclusion and future work

• Along with circuit compilers, hybrid protocol provides an easy toolchain to build robust secure computations
• Optimize implementation – parallelization
• Leverage circuit compilers
• Goal – to build secure IEEE-754 floating-point protocol suite