Secure Computation: Chaining Garbled Circuits:

Alex Ledger

Reed College

April 12, 2016

Overview

1. Secure Computation
2. Garbled Circuits
3. Chaining Garbled Circuits
4. My work: SCMC and Programming

Goals and Notes

1. Understand the high level idea of secure computation
   - Maybe you'll run into a situtation where it's useful, and it can help solve an otherwise unsolvable, real world problem.
2. Understand garbled circuits - the most basic construction
3. Understand chaining - how we (my thesis) made secure computation faster

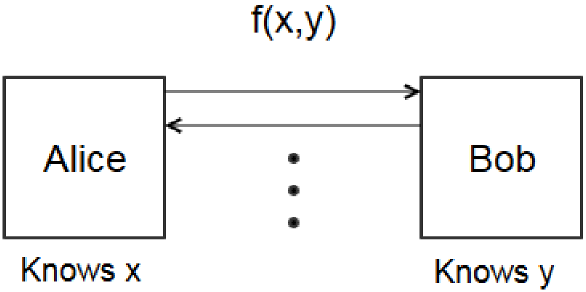
Before we start:

- I wil not emphasize security
- Lots of notation: ask me if I brush over something, or you forget what something means
- Lots of moving parts

The Millionaire Problem

- Alice and Bob want to determine who is wealthier.
- They do want to disclose their wealth to each other.
  - Alice has \$$x$, Bob has \$$y$.
  - Alice should not learn anything about y.
  - Bob should not learn anything about x.

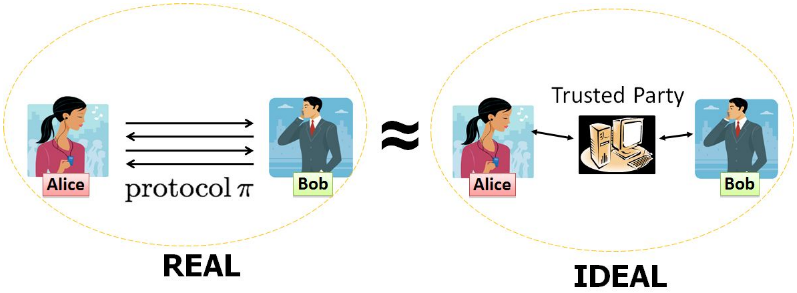
$$f(x,y) = \begin{cases} \mathit{Alice}, & y \leq x; \\ \mathit{Bob}, & y > x. \end{cases} \tag{1}$$



Security Properties

- Privacy of inputs
  - Alice and Bob do not learn anything about the other's input.
  - Except for info that is inferable from x and $f(x, y)$.
  - Bob should not learn that $1,000,000 < x \leq 2,000,000$.
  - But if $y < x$ and $y = 2,000$, then he learns $x < 2,000$.
- Correctness
  - Alice and Bob receive $f(x, y)$.
  - As opposed to some value near $f(x, y)$.
  - Or not receiving a value at all
- Semi-honest
  - We assume that each party obeys the protocol, but attemps to learn extra information from its interactions

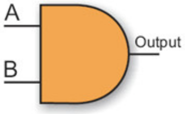
Security

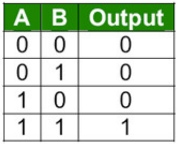


A secure computation protocol is secure if Alice and Bob learn the same information in the real world as they would in ideal world.

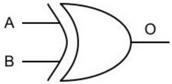
Boolean Circuits

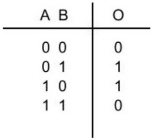
- We encode a function f into a circuit C.
- Circuit C is made of AND, XOR and NOT gates.
- Each gate has two input wires and a single output wire





A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1

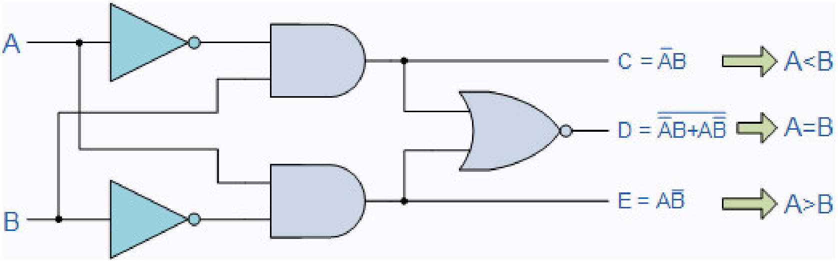




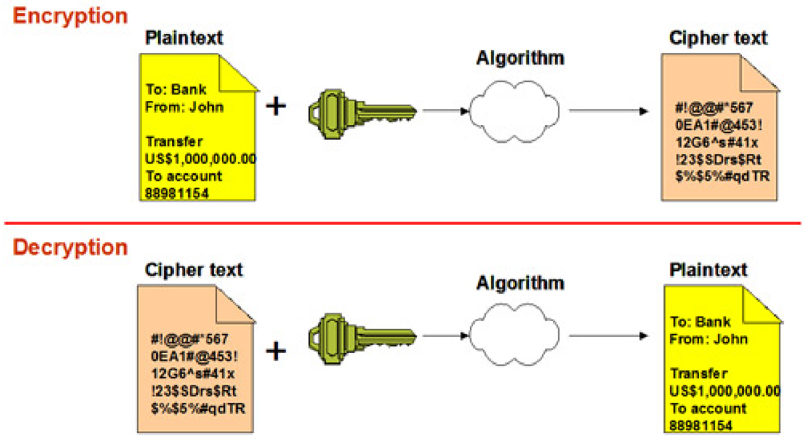
A	B	O
0	0	0
0	1	1
1	0	1
1	1	0

Boolean Circuits

- Any function can be encoded into a circuit.
- Here is the less than circuit.

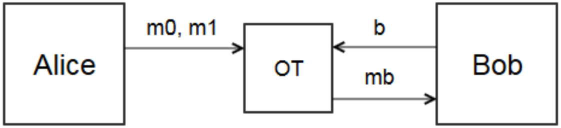


Encryption


Encryption
Plaintext
To: Bank
From: John
Transfer
US$1,000,000.00
To account
88981154
+
Algorithm
Cipher text
#!@@#*567
0EA1#@453!
12G6^s#41x
!23$$Drs$Rt
$%$5%#qd TR
Decryption
Cipher text
#!@@#*567
0EA1#@453!
12G6^s#41x
!23$$Drs$Rt
$%$5%#qd TR
+
Algorithm
Plaintext
To: Bank
From: John
Transfer
US$1,000,000.00
To account
88981154

Oblivious Transfer (OT) in brief

- Alice potentially sends either m_0 or m_1 to Bob.
- Bob, without seeing m_0 or m_1, decides that he wants m_b.
- Bob receives m_b.
- Property 1: Alice does not know which message Bob recieved.
- Property 2: Bob doesn't anything about m_{1-b}.



Hash Function in brief

- Hash function $H : \{0,1\}^* \to \{0,1\}^{128}$
- For our purposes, H maps any string to a uniform, random 128-bit string.
- A.k.a. H is a random oracle.

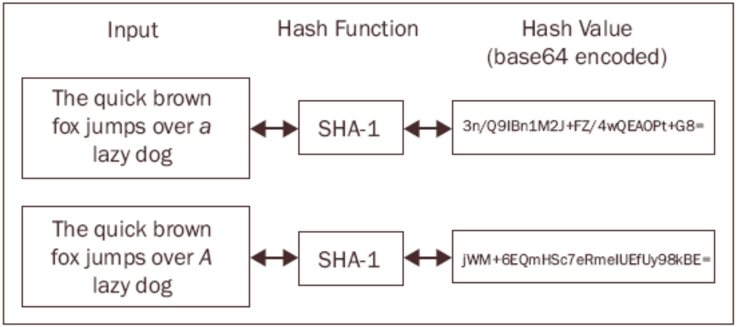
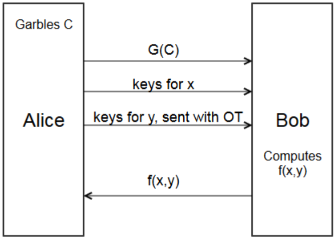

Input
Hash Function
Hash Value (base64 encoded)
The quick brown fox jumps over a lazy dog
SHA-1
3n/Q9IBn1M2J+FZ/4wQEAOPt+G8=
The quick brown fox jumps over A lazy dog
SHA-1
jWM+6EQmHSc7eRmeIUEfUy98kBE=

Figure 17: Hash Function

Roadmap of Garbled Circuits

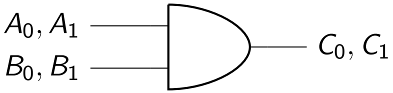
1. Alice (garbler) *garbles* the AND gate.
2. Alice sends the *garbled table* of the gate and some keys to Bob.
3. Bob (evaluator) *evaluates* the gate.



Garbling a gate 1

Step 1. Alice assigns *wire labels* to each wire.

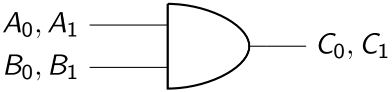
- For each wire in the circuit, assign two random labels to each wire
- Wire A has two wire labels A_0 and A_1.
- We say A_0 *semantically represents* 0, and A_1 *semantically represents* 1.
- And A_0 and A_1 are sampled uniform randomly from $\{0,1\}^n$.
  - Generally use AES-128 for encryption, so $n = 128$.



Garbling a gate 2

Step 2. Alice constructs garbled table.

- Encrypt wire labels for C, C_0 and C_1, using wire labels of A and B.
- Randomly permute table



A	B	Encryption
A_0	B_0	$\mathsf{Enc}_{A_0,B_0}(C_0)$
A_1	B_0	$\mathsf{Enc}_{A_1,B_0}(C_0)$
A_0	B_1	$\mathsf{Enc}_{A_0,B_1}(C_0)$
A_1	B_1	$\mathsf{Enc}_{A_1,B_1}(C_1)$

Garbling a gate 3

Step 3. Send garbled table to Bob.

$$\boxed{\begin{array}{c}\mathsf{Enc}_{A_0,B_0}(C_0)\\ \mathsf{Enc}_{A_1,B_0}(C_0)\\ \mathsf{Enc}_{A_0,B_1}(C_0)\\ \mathsf{Enc}_{A_1,B_1}(C_1)\end{array}}$$

Garbling a gate 4

Step 4. Alice sends input wire labels to Bob.

- Suppose $x \in \{0, 1\}$ is Alice's input, and $y \in \{0, 1\}$ is Bob's input.
- Alice sends A_x to Bob.
- Alice sends B_y to Bob via Oblivious Transfer
  - The wire labels corresponding to her inputs.
- Bob has:

Garbled Table
$\mathsf{Enc}_{A_0,B_0}(C_0)$
$\mathsf{Enc}_{A_1,B_0}(C_0)$
$\mathsf{Enc}_{A_0,B_1}(C_0)$
$\mathsf{Enc}_{A_1,B_1}(C_1)$

Input Labels
A_*
B_y

Garbling a gate 5

Step 5. Bob evaluates the circuit

- Bob has the garbled table, A_x and B_y.
- Bob trial decrypts each row of the garbled table, until an encrytion succeeds.
- Bob acquires $C_{x \wedge y}$.
- Bob has:

Garbled Table
$\mathrm{Enc}_{A_0,B_0}(C_0)$
$\mathrm{Enc}_{A_1,B_0}(C_0)$
$\mathrm{Enc}_{A_0,B_1}(C_0)$
$\mathrm{Enc}_{A_1,B_1}(C_1)$

Input Labels
A_*
B_y

Output Label
$C_{x \wedge y}$

Garbling a gate 6

Step 6. Bob gets a final answer.

- Alice sends $\mathrm{Enc}_{C_0}(0)$ and $\mathrm{Enc}_{C_1}(1)$ to Bob.
- Bob trial decrypts these with $C_{x \wedge y}$.
- One will succeed, and Bob will acquire $z = x \wedge y$.
- So Bob knows z, but not x!

Garbled Table
$\mathrm{Enc}_{A_0,B_0}(C_0)$
$\mathrm{Enc}_{A_1,B_0}(C_0)$
$\mathrm{Enc}_{A_0,B_1}(C_0)$
$\mathrm{Enc}_{A_1,B_1}(C_1)$

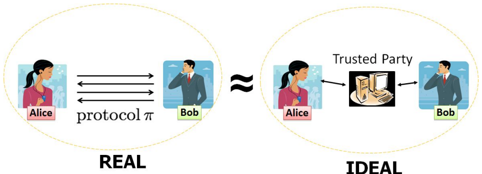
Input Labels
A_*
B_y

Output Label
$C_{x \wedge y}$

Output Map
$\mathrm{Enc}_{C_0}(0)$
$\mathrm{Enc}_{C_1}(1)$

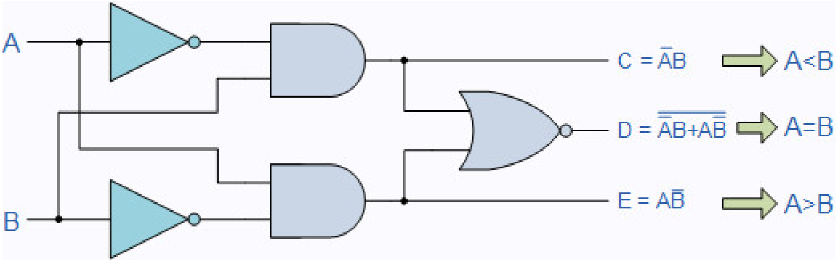
Security Considerations

- Think about what Alice acquires:
  - Well Bob had no input in this case, so ...
  - In general, Alice generates objects and sends them to Bob
  - So she doesn't have many opportunities to learn about Bob's input.
- Think about what Bob acquires:
  1. Garbled table
  2. Input wire labels: A_a and B_b
  3. Encryptions of output: $\text{Enc}_{C_0}(0)$ and $\text{Enc}_{C_1}(1)$
- Can Bob learn anything about a or b?
  - If he could decrypt another row of the garbled table, then we would learn something.
    - But he can't, because he doesn't have the keys.
  - Not really, because everything is encrypted.

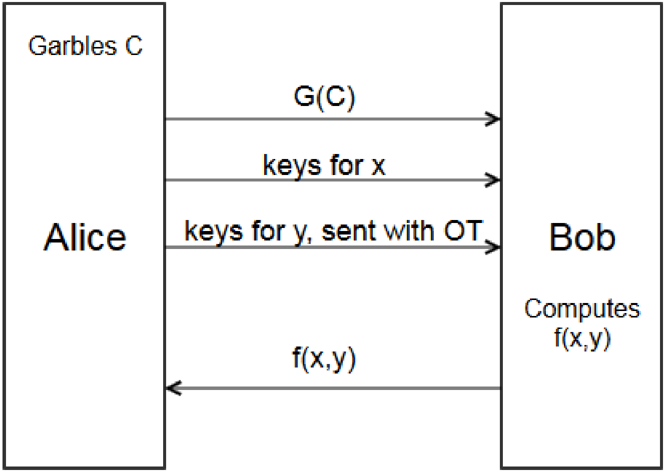


Extending a garbled gate into a garbled circuit

- To operate on a more complex function, the operation is recursed.
- The output wires of the first gates are used as inputs to subsequent gates.
- Alice only sends output maps for the final gates.



The Garbled Circuit Protocol



The cost of garbled circuits

- Alice sends 4 ciphertexts per gate, since the garbled table has 4 rows, to Bob.
- This is the bandwidth, or the amount of the communication required.
- Based on empirical work, bandwidth is the biggest bottleneck in garbled circuits.
- So reducing the size of the garbled table is of utmost priority.

Reducing the size of the garbled table with Free XOR

- Let $\Delta \leftarrow \{0,1\}^n$ be fixed globally in a circuit.
- For each input wire A, sample a single ciphertext; call it A.
  - The *zeroith* wire label is A.
  - The *first* wire label is $A \oplus \Delta$.
- Set output wire C of a gate to be $A \oplus B$.
  - The xor of its input wires.
- Bob *evaluates* the XOR gate by XORing the two input labels.

$$(A \oplus a\Delta) \oplus (B \oplus b\Delta) = A \oplus B \oplus (a \oplus b)\Delta$$

- So XOR gates do not require a garbled table, aka they're Free.

Offline/Online

- Imagine two banks use secure computation during their daily operations
  - E.g. ATM operations
- At night - the offline phase - they exchange garbled circuits (the garbled tables)
  - So garbled circuits are ATM transactions
- During the day - the online phase - they exchange input wire labels and use the pre-exchanged garbled circuits.
- The computation is fast!
- Problems
  - Functions are decided at night - no room for flexibility
  - Input size is fixed at night

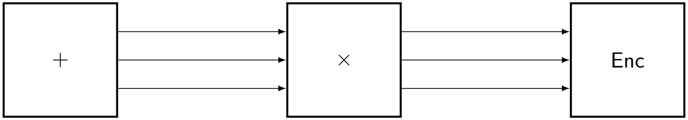
ATM Transaction

Insurance Operation

Stastical Operation

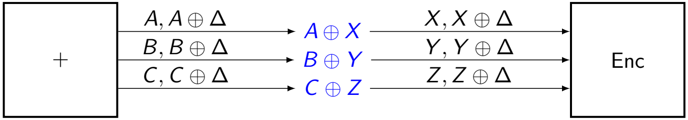
Chaining Garbled Circuits

- Goal: improve offline/online garbled circuits by adding flexibility
- Key observation: many useful functions in the real world are composed of small, standard components.
  - E.g. addition, subtraction, matrix operations
  - Leveshtein distance algorithm - a dynamic algorithm
  - Encryption is a common component (banks)
- Idea: chain garbled circuits together
  - Take the output of one garbled circuit and plug it into another garbled circuit



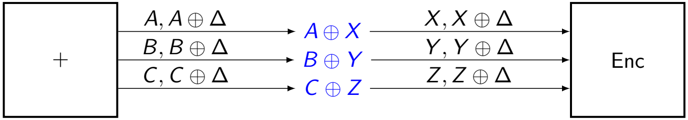
How to chain garbled circuits

- Suppose that we are chaining a garbled circuit with output wire A to garbled circuit with input wire X.
- We want $A \to X$ and $X \oplus \Delta \to X \oplus \Delta$.
- Straightforward:
  - Alice sends Bob $L_{AX} = A \oplus X$
  - Bob sets $X_* \leftarrow A_* \oplus L_{AX}$



How to chain garbled circuits

- Suppose that we are chaining a garbled circuit with output wire A to garbled circuit with input wire X.
- We want $A \rightarrow X$ and $X \oplus \Delta \rightarrow X \oplus \Delta$.
- Straightforward:
  - Alice sends Bob $L_{AX} = A \oplus X$
  - Bob sets $X_* \leftarrow A_* \oplus L_{AX}$



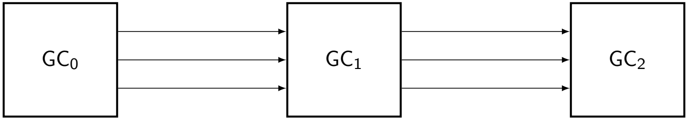
- Is this secure?

Efficiency of chaining garbled circuits

- In online phase, chaining requires communicating a ciphertext per chain
- This can be a lot of communication
  - Suppose 100 by 100 matrix with entries of max value 2^8
  - Then requires $8,000$ ciphertexts
  - That's 256 kBs.
- I came up with a method to reduce this to a single ciphertext per data object

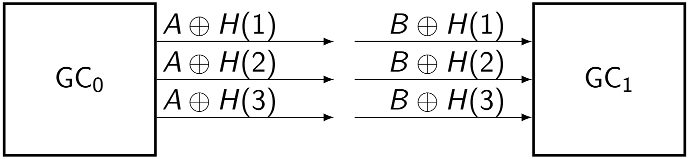
Single Communication Multiple Connections SCMC

- Problem: number of ciphertexts communicated scales linearly with the number of chains
- Key observation:
  - the chaining is predictable, consecutive wires are chained to consecutive wires
  - Pieces of data move around in a function, like matrices or numbers or text
- Solution: give consecutive wire labels a predictable pattern



Single Communcation Multiple Connections SCMC

- For each piece of data, sample a label A.
- Set ith wire label of data to $A \oplus H(i)$.
- Where H is a hash function



- Technically, we set the links to $A \oplus H(T \oplus (i||b))$

My thesis work

- I implemented chaining and SCMC in a program called CompGC
- Written in C
- It does the entire garbled circuits protocol
- Takes circuits and inputs and securely computed the output
- It's fast
    - Leventshein-60 with SCMC: $\approx$ 750 ms
    - Without chaining: $\approx$ 10 s

Conclusion

We talked about:

- Secure computation
- Garbled circuits
- Chaining garbled circuits
- SCMC
- CompGC

Thank you:

- Adam Groce
- Alex Malozemoff and Arkady Yerukhimovich
- Catdog and friends