Implementing Component-Based Garbled Circuits

Alex Ledger

Reed College

May 2, 2016

Security of Encryption

An encryption scheme is secure under a chosen-plaintext attack if for all probabilistic polynomial-time adversaries A, there exists a negligible function μ such that

$$Pr[E_{\mathcal{A},\Pi}(n) = 1] \leq \frac{1}{2} + \mu(n)$$

where E is the following experiment:

1. Generate key k by running $\mathrm{Gen}(1^n)$.
2. The adversary $\mathcal{A}$ is given 1^n and oracle access to Enc_k, and outputs a pair of messages m_0 and m_1 of the same length.
3. A uniform bit $b \leftharpoondown \{0, 1\}$ is sampled uniformly at random, and then ciphertext $c \leftarrow \mathrm{Enc}_k(m_b)$ is computed and given to $\mathcal{A}$.
4. Then $\mathcal{A}$ continues to have oracle access to Enc_k, and outputs a bit b'.
5. The output of the experiment is defined to be 1 if $b' = b$ and 0 otherwise. If at any point $\mathcal{A}$ encrypts m_0 or m_1 with their encryption oracle, the output is 0. 1 indicates that the adversary wins, and 0 indicates that the adversary loses.

Computational Indistinguishability

Intuition:

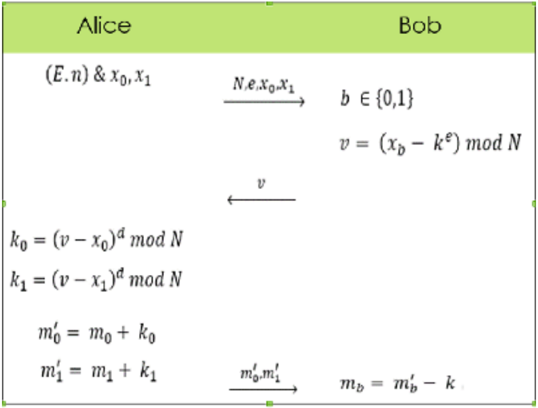
- Let $\mathcal{X}$ and $\mathcal{Y}$ be distribution ensembles.
- Give an adversary one of the distribution ensembles.
- Can the adversary sample from the distribution a polynomial number of times to determine which distribution they were given?

Formal Definition:

- Let $\mathcal{X} = \{X_n\}_{n\in\mathbb{N}}$ and $\mathcal{Y} = \{Y_n\}_{n\in\mathbb{N}}$ be distribution ensembles.
- Then $\mathcal{X}$ and $\mathcal{Y}$ are computationally indistinguishable, denoted $\mathcal{X} \approx_C \mathcal{Y}$, if for all probabilistic polynomial-time algorithms D, there exists a negligible function μ such that:

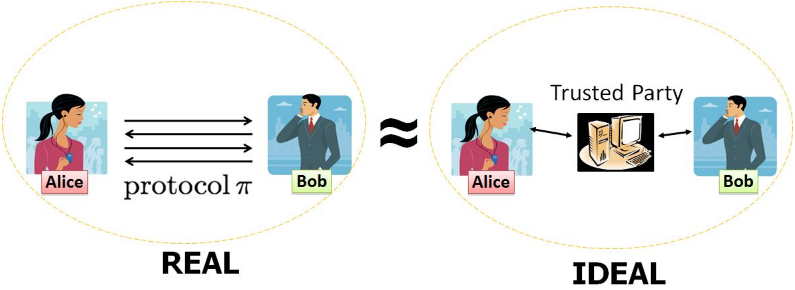
$$|Pr_{x\leftarrow X_n}[D(1^n, x) = 1] - Pr_{y\leftarrow Y_n}[D(1^n, y) = 1]| < \mu(n)$$

Oblivious Transfer 2



RSA-based oblivious transfer

Security of 2PC (Overview)



Security of 2PC (Ideal World)

Define the following:

$$\{S_A(x, f(x,y))\}_{x \in \{0,1\}^*}$$
$$\{S_B(y, f(x,y))\}_{y \in \{0,1\}^*}$$

$$\{\text{view}_A(x,y)\}_{x,y \in \{0,1\}^*}$$
$$\{\text{view}_B(x,y)\}_{x,y \in \{0,1\}^*}$$

$$output_i^{\Pi}(n, x, y)$$

Then Π is secure if:

$$\{(S_A(x, f_A(x,y)), f(x,y))\}_{x,y} \approx_C \{(\text{view}_A^{\Pi}(x,y), \text{output}^{\Pi}(x,y))\}_{x,y}$$
$$\{(S_B(x, f_B(x,y)), f(x,y))\}_{x,y} \approx_C \{(\text{view}_B^{\Pi}(x,y), \text{output}^{\Pi}(x,y))\}_{x,y}$$

Garbled Circuits

1. Alice assigns wire labels
2. Alice make garbled tables
    - Alice permutes rows
3. Alice sends garbled tables to Bob
4. Alice sends input wire labels to Bob
    - Some are sent via OT
    - Others are simply sent
5. Bob trial decrypts each row of garbled table with input wire labels
6. Bob acquires output wire label
7. Bob uses output wire label in subsequent gates
8. Bob eventually acquires output wire labels

Improvements to Garbled Circuits

Garbled Circuit Improvement	Table Size (xλ)		Garble Cost		Eval Cost	
	XOR	AND	XOR	AND	XOR	AND
Classical	4	4	4	4	4	4
Point and Permute	4	4	4	4	1	1
GRR3	3	3	4	4	1	1
Free XOR	0	3	0	4	0	1
GRR2	2	2	4	4	1	1
FleXOR	{0,1,2}	2	{0,2,4}	4	{0,1,2}	1
Half Gates	2	0	2	0	0	2

Table: Table size is number of ciphertexts. Garble cost is number of encryptions the garbler performs. Eval cost is number of decryptions the evaluator performs.

Point and Permute

All examples use an AND gate with input wires W_i and W_j and output wire W_k.

Select Bit	Wire Label
0	W_i^0
1	W_i^1
1	W_j^0
0	W_j^1

Select Bits	Encryption
(0,0)	$\mathrm{Enc}_{W_i^0, W_j^1}(W_k^0)$
(0,1)	$\mathrm{Enc}_{W_i^0, W_j^0}(W_k^0)$
(1,0)	$\mathrm{Enc}_{W_i^1, W_j^1}(W_k^1)$
(1,1)	$\mathrm{Enc}_{W_i^1, W_j^0}(W_k^0)$

Table: Garbled AND gate for Point and Permute

Garbled Row Reduction 3

Select Bit	Wire Label
0	W_i^0
1	W_i^1
1	W_j^0
0	W_j^1

Select Bits	Encryption
(0,1)	$\mathsf{Enc}_{W_i^0, W_j^0}(W_k^0)$
(1,0)	$\mathsf{Enc}_{W_i^1, W_j^1}(W_k^1)$
(1,1)	$\mathsf{Enc}_{W_i^1, W_j^0}(W_k^0)$

$$\boxed{\begin{array}{c} W_k^0 \leftarrow \mathsf{Enc}^{-1}_{W_i^0, W_j^1}(0^n) \\ W_k^1 \leftharpoondown \{0,1\}^n \end{array}}$$

Free XOR

- Let $\Delta \leftarrow \{0,1\}^n$ be fixed globally in a circuit.
- For each input wire:
  - Sample W_i^0 randomly.
  - Set $W_i^1 \leftarrow W_i^0 \oplus \Delta$.
- For each output wire of a gate W_k
  - Set $W_k^0 \leftarrow W_i^0 \oplus W_j^0$.
  - Set $W_k^1 \leftarrow W_k^0 \oplus \Delta$.
- 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.

OT-preprocessing

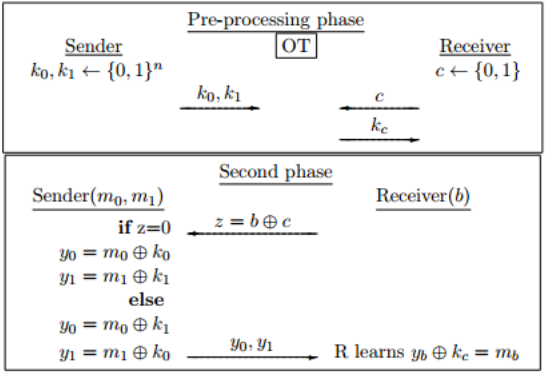


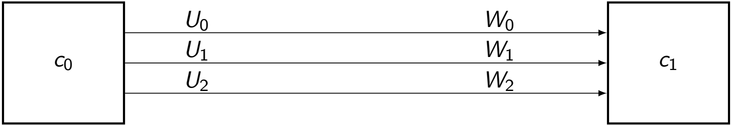
Figure 1: Protocol for Pre-processing OT

Source: Katz lecture notes

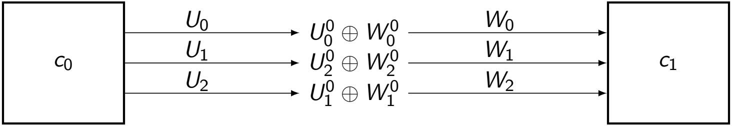
Motivating Component-Based GCs

- Split protocols into offline/online
  - Garble and send circuit in offline phase.
  - Determine inputs during online phase, evaluate.
- Problem: function chosen ahead of time - no flexiblity

Details of Component-Based Garbled Circuits (1)

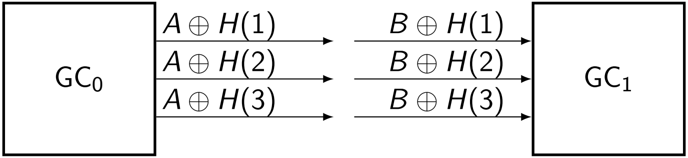


Details of Component-Based GCs (2)



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 $A \oplus H(T \oplus (i||b))$

Results

	Time (localhost)		Time (simulated network)		Communication	
	Naive	CompGC	Naive	CompGC	Naive	CompGC
AES	4.4 $\pm$ 0.0 ms	3.0 $\pm$ 0.2 ms	542.6 $\pm$ 0.7 ms	68.5 $\pm$ 0.2 ms	24 Mb	254 Kb
CBC, 10 blocks	45.8 $\pm$ 4.0 ms	22.7 $\pm$ 1.4 ms	4.8 $\pm$ 0.0 s	216.7 $\pm$ 0.2 ms	235 Mb	2.6 Mb
Leven, 30 symbols	28.9 $\pm$ 6.6 ms	24.3 $\pm$ 1.2 ms	2.2 $\pm$ 0.0 s	315.9 $\pm$ 0.5 ms	108 Mb	6.3 Mb
Leven, 60 symbols	109.8 $\pm$ 7.0 ms	62.2 $\pm$ 0.7 ms	10.6 $\pm$ 0.0 s	742.5 $\pm$ 2.0 ms	524 Mb	25 Mb

Table: Experimental results. Time is online computation time, not including the time to preprocess OTs, but including the time to load data from disk. All timings are of the evaluator's running time, and are the average of 100 runs, with the value after the $\pm$ denoting the 95% confidence interval. The communication reported is the number of bits received by the evaluator.

Results Without Loading Time

	Time (localhost)		Time (simulated network)	
	Naive	CompGC	Naive	CompGC
AES	4.4 ± 0.0 ms	1.3 ± 0.1 ms	542.6 ± 0.7 ms	66.9 ± 0.1 ms
CBC mode, 10 blocks	45.8 ± 4.0 ms	8.8 ± 0.5 ms	4.8 ± 0.0 s	204.3 ± 0.2 ms
Levenshtein, 30 symbols	28.9 ± 6.6 ms	14.1 ± 0.4 ms	2.2 ± 0.0 s	305.6 ± 0.2 ms
Levenshtein, 60 symbols	109.8 ± 7.0 ms	27.1 ± 0.4 ms	10.6 ± 0.0 s	703.4 ± 1.5 ms

Table: Experimental results without counting the evaluator time to load data from disk.

Comparing Naive to SCMC

	Time (simulated network)		**Communication**	
	Standard	SCMC	Standard	SCMC
AES	134.4 $\pm$ 0.1 ms	68.5 $\pm$ 0.2 ms	656 Kb	254 Kb
CBC mode, 10 blocks	321.5 $\pm$ 0.9 ms	216.7 $\pm$ 0.2 ms	7.4 Mb	2.6 Mb
Levenshtein, 30 symbols	371.0 $\pm$ 0.9 ms	315.9 $\pm$ 0.5 ms	10.0 Mb	6.3 Mb
Levenshtein, 60 symbols	1119.6 $\pm$ 2.1 ms	742.5 $\pm$ 2.0 ms	44 Mb	25 Mb

Table: Comparison of the two approaches for component-based garbled circuits: the standard approach and the SCMC approach. The experiments are run on the simulated network.