Introduction to Secure Computation, part 2

Jonathan Katz

Outline: Hour 2

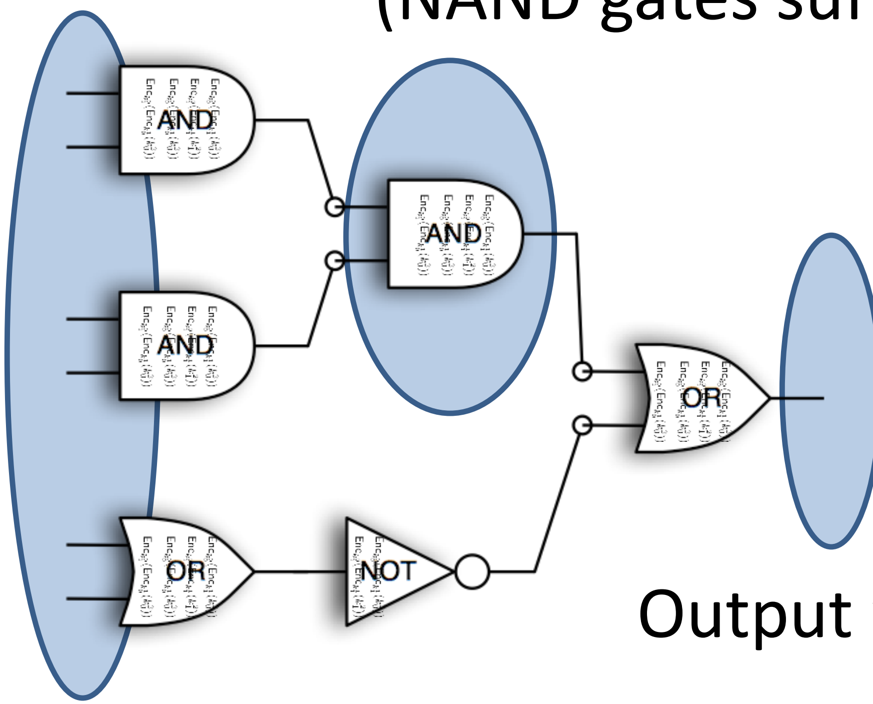
- *Semi-honest* (generic) secure computation
  - Garbled circuits and constant-round secure two-party computation
  - Secure multi-party computation
  - Constant-round secure multi-party computation

Generic secure computation

- How to prove a generic feasibility result?
- Work with *universal model of computation*
  - I.e., show protocol for any function expressed in that model
  - Here: Boolean circuits with NAND gates
- Other choices
  - Turing machines
  - RAMs
  - Arithmetic circuits

Boolean circuits

Internal gate
(NAND gates suffice)



Output wires

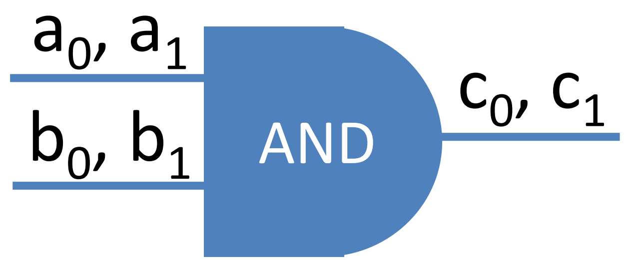
Input wires,
each “belonging” to one party

Note that out-degree can be > 1

Garbled circuits [Yao86]

- Main idea
  - Both parties agree on a Boolean circuit C
  - “Garbler” prepares a “garbled version” of C that can be evaluated “obliviously” (i.e., without knowledge of wire values)
  - “Evaluator” evaluates the garbled version of C

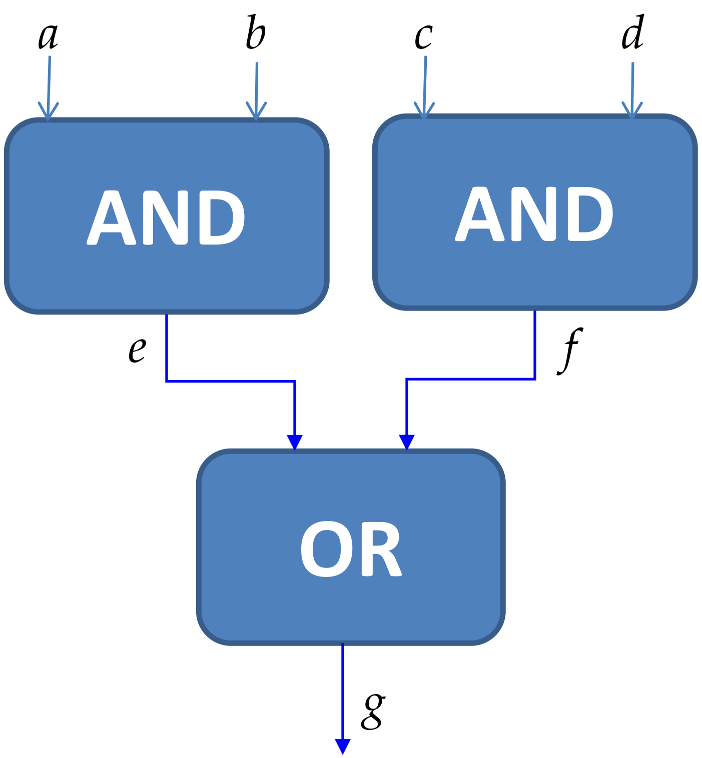
Garbled gate



1st wire	2nd wire	Output wire
a_0	b_0	c_0
a_0	b_1	c_0
a_1	b_0	c_0
a_1	b_1	c_1

AND Gate
$Enc_{a1,\ b0}(c_0)$
$Enc_{a1,\ b1}(c_1)$
$Enc_{a0,\ b1}(c_0)$
$Enc_{a0,\ b0}(c_0)$

Garbled circuit

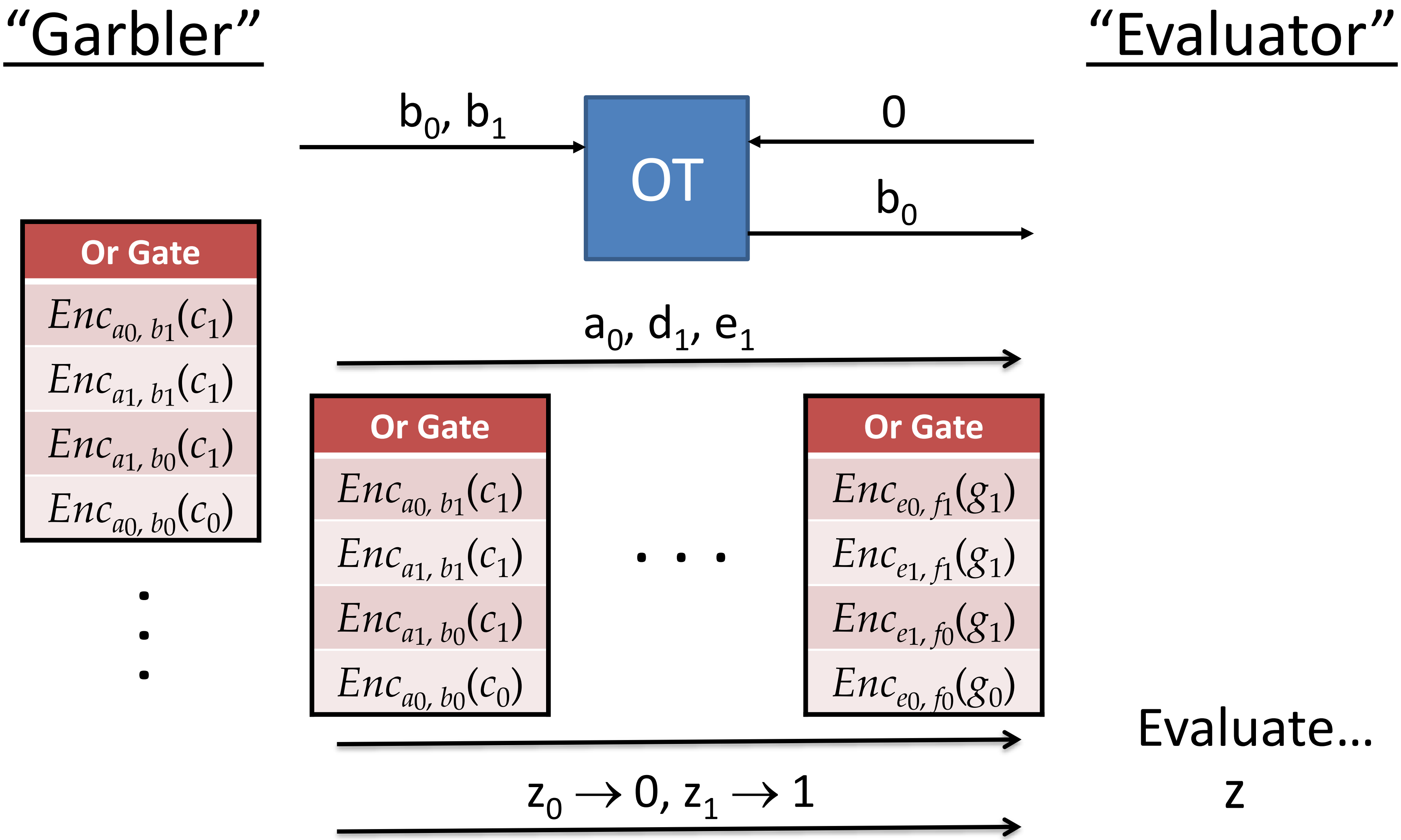


And Gate
$Enc_{a0,\ b1}(e_0)$
$Enc_{a1,\ b1}(e_1)$
$Enc_{a0,\ b0}(e_0)$
$Enc_{a1,\ b0}(e_0)$

And Gate
$Enc_{c1,\ d1}(f_1)$
$Enc_{c1,\ d0}(f_0)$
$Enc_{c0,\ d1}(f_0)$
$Enc_{c0,\ d0}(f_0)$

Or Gate
$Enc_{e0,\ f1}(g_1)$
$Enc_{e1,\ f1}(g_1)$
$Enc_{e1,\ f0}(g_1)$
$Enc_{e0,\ f0}(g_0)$

Semi-honest 2PC



Invariant

- For each wire w with value b (when evaluated on parties' inputs), evaluator knows key w_b
- Proof by induction:
  - Base case: input wires:
    - Evaluator's input wires: from OT
    - Garbler's input wires: by construction
  - Inductive step
    - By construction of garbled gate

Correctness

- At end of protocol, evaluator knows z_b
  - Can recover output b from the decoding table sent by the garbler

Proof of security?

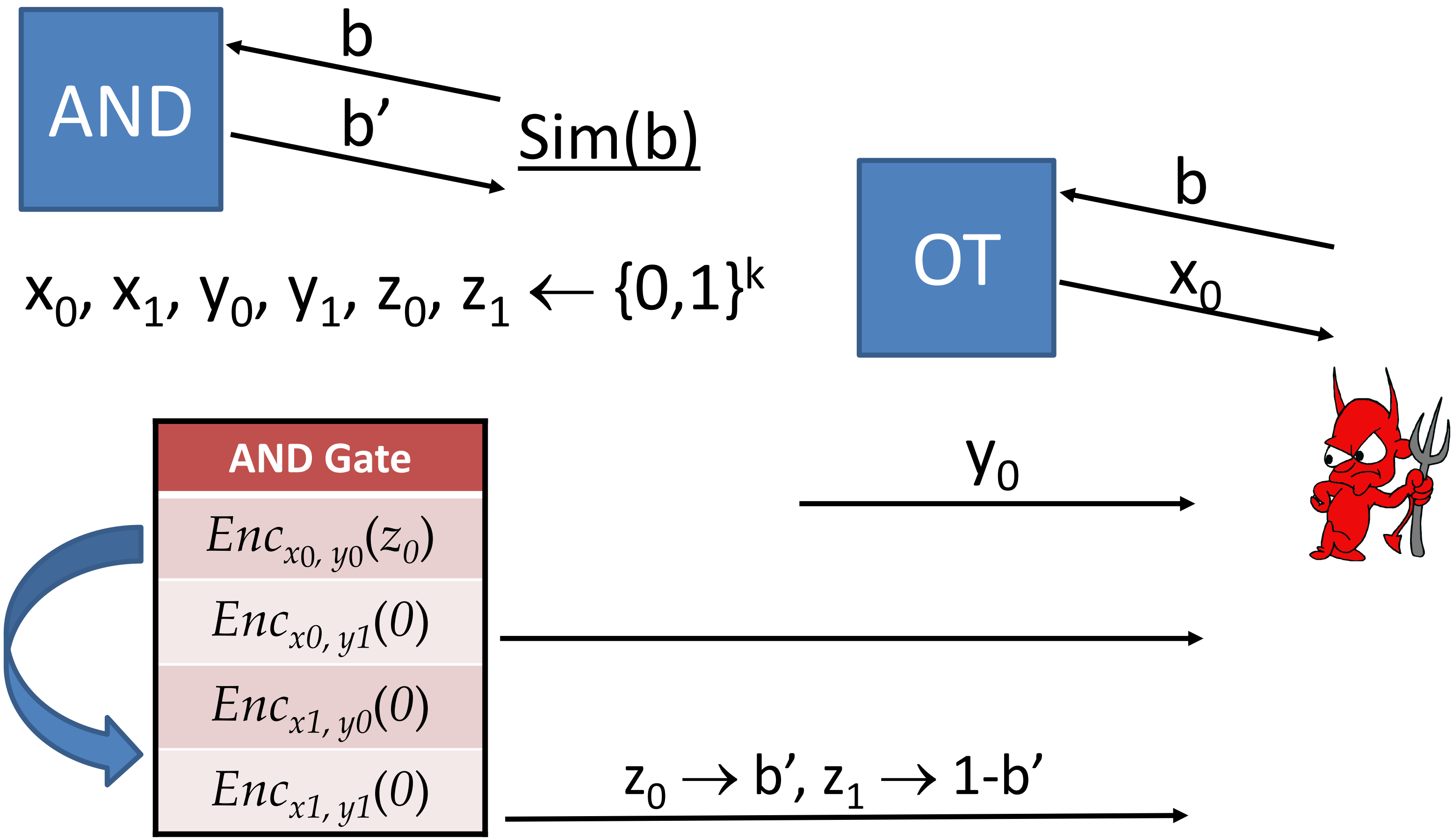
- Prove security in *OT-hybrid world*
  - Rely on composition theorem
- Prove (semi-honest) security for one AND gate
  - Proof for larger circuits tedious but not difficult

Proof of security I

- Simulator for corrupted garbler:
  - Trivial!
  - All communication is *from* the garbler (in OT-hybrid world)
    - No messages to simulate!

Proof of security II

- Simulator for corrupted evaluator:



Indistinguishability (sketch)

- In real world, garbled table is prepared properly
- In ideal world, all rows except one are encryptions of “garbage”
- Since evaluator is missing all-but-one set of keys, these are indistinguishable!

Optimization

- As described, the evaluator tries to decrypt all 4 rows of a garbled gate
- Can do better by having the garbler assign a random label to each key

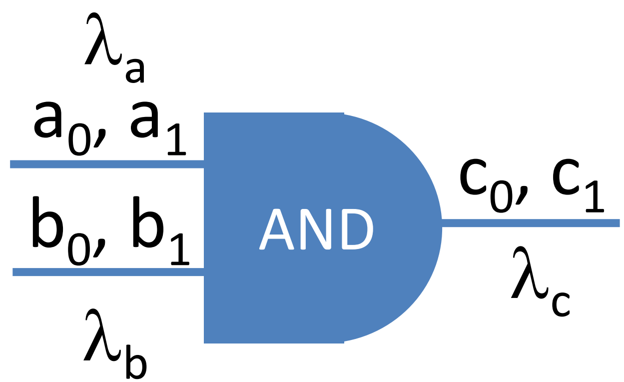
Garbled gate

$a_0 \rightarrow 0$
$a_1 \rightarrow 1$

$a_0 \rightarrow \lambda_a$
$a_1 \rightarrow 1 \oplus \lambda_a$

$b_0 \rightarrow 0$
$b_1 \rightarrow 1$

$b_0 \rightarrow \lambda_b$
$b_1 \rightarrow 1 \oplus \lambda_b$



$c_0 \rightarrow 0$
$c_1 \rightarrow 1$

$c_0 \rightarrow \lambda_c$
$c_1 \rightarrow 1 \oplus \lambda_c$

$a_0, b_0 \Rightarrow c_{\lambda_{00}}$, where
$\lambda_{00} = \lambda_c \oplus (\lambda_a \wedge \lambda_b)$

AND Gate
$Enc_{a0,\, b0}(\lambda_{00}, c_{\lambda_{00}})$
$Enc_{a0,\, b1}(\lambda_{01}, c_{\lambda_{01}})$
$Enc_{a1,\, b0}(\lambda_{10}, c_{\lambda_{10}})$
$Enc_{a1,\, b1}(\lambda_{11}, c_{\lambda_{11}})$

Properties of the protocol

- Protocol runs in *constant rounds*, regardless of the circuit
- #OTs linear in the input lengths
- Symmetric-key operations/communication linear in the circuit size
- In OT-hybrid world:
  - Perfect security for corrupted generator
  - Computational security for corrupted evaluator

Main drawback

- How to extend to *multi-party* case?

GMW protocol [GMW87]

- Main idea:
  - Parties *interactively* evaluate the circuit on their inputs, in an oblivious fashion
  - Using a “mini” secure computation at each step!

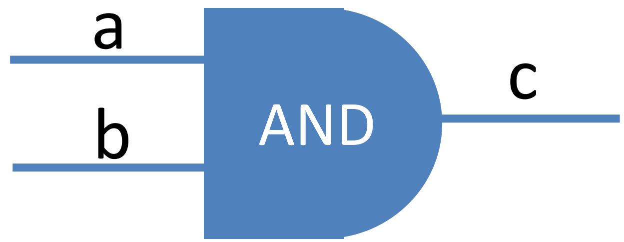
- Look at two-party case first, then extend to multi-party case

Invariant

- For each wire w with value b (when evaluated on parties' inputs), parties will hold a random *secret sharing* of b

- Note: easy to establish invariant at input wires

Evaluating a gate



$\underline{P_1}$ $\underline{P_2}$

a_1, b_1 a_2, b_2

$c_1 \leftarrow \{0,1\}$

For $x, y \in \{0,1\}$:

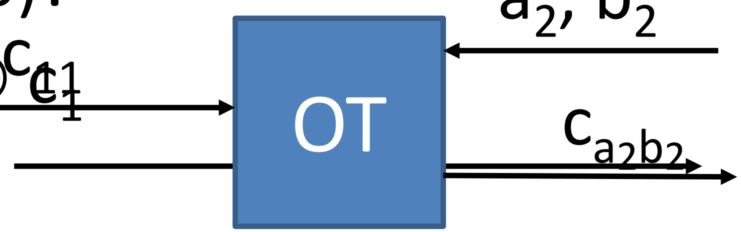
$c_{xy} = (a_1 \oplus x)\cdot(b_1 \oplus y) \oplus c_1$

What if $(a_2, b_2) = (0,0)$?

$c_2 = (a_1 \oplus 0)\cdot(b_1 \oplus 0) \oplus c_1$

$c_{00}, c_{01}, c_{10}, c_{11}$

a_2, b_2



$c_{a_2 b_2}$

$c_2 = c_{a_2 b_2}$

Output determination

- For output wires, parties simply reveal their shares to each other

Proof of security?

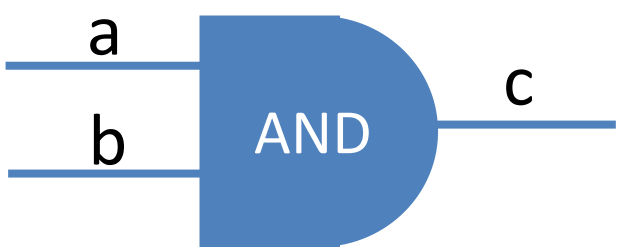
- Proof is symmetric for each party
- Prove security in OT-hybrid world
- Simulator
  - Generate random share for each input wire
  - Simulate computation of each gate by giving random OT output
  - For each output wire, send share that results in (known) output value
- *Perfectly secure* in OT-hybrid world!

Properties of the protocol

- Protocol has round complexity *linear* in the depth of the circuit
  - (Computation of gates at same depth can be parallelized)
- #OTs/communication linear in the circuit size
- In OT-hybrid world, perfect security for both parties

GMW multi-party case

- Use n-out-of-n secret sharing
  - So $a = a_1 \oplus \cdots \oplus a_n$



- To evaluate an AND gate, use the fact that

$$c = a \cdot b = (a_1 \oplus \cdots \oplus a_n) \cdot (b_1 \oplus \cdots \oplus b_n)$$
$$= \Sigma_{i,j}\, a_i b_j$$
$$= \Sigma_i\, a_i b_i + \Sigma_{i \neq j}\, a_i b_j$$

Compute locally | Compute as before

GMW protocol

- Each party shares their inputs as before
- To evaluate an AND gate with input wires shared as $\{a_i\}$ and $\{b_i\}$ do:
  - For all $i \neq j$, parties P_i and P_j compute random c_{ij}, c_{ji}, resp., such that $c_{ij} \oplus c_{ji} = a_i b_j$, and random d_{ij}, d_{ji}, resp., such that $d_{ij} \oplus d_{ji} = a_j b_i$
    - This is done using OT, as previously
  - Each P_i locally computes $c_i = a_i b_i \oplus \Sigma_j\, c_{ij} \oplus \Sigma_j\, d_{ij}$
    - Note $\oplus_i\, c_i = c = (a_1 \oplus \cdots \oplus a_n) \cdot (b_1 \oplus \cdots \oplus b_n)$
- Output wires handled as before

Security?

- As before, possible to prove perfect security for any $t < n$ corrupted parties in the OT-hybrid world

Properties of the protocol

- Protocol has round complexity *linear* in the depth of the circuit
  - (Computation of gates at same depth can be parallelized)
- #OTs/communication = $O(n^2 \cdot |C|)$
- In OT-hybrid world, perfect security for any number of corrupted parties

The BMR protocol

- A *constant-round* multi-party protocol
- Main idea:
  1. Parties run a linear-round MPC protocol to compute a garbled circuit GC
  2. All parties evaluate the garbled circuit
- Key insight: if the circuit for computing GC has constant depth, then the protocol runs in constant rounds

Computing a garbled gate

- Say all keys are secret shared
  - E.g., $a_0 = a_{01} \oplus a_{02} \oplus \cdots \oplus a_{0n}$
- Consider the function that maps these shares to a garbled gate
  - Problem: unclear what is the depth of the circuit computing Enc
  - Solution: need to choose the right encryption scheme…
    - And be slightly clever about how it is implemented

AND Gate
$Enc_{a1,\ b0}(c_0)$
$Enc_{a1,\ b1}(c_1)$
$Enc_{a0,\ b1}(c_0)$
$Enc_{a0,\ b0}(c_0)$

Computing a garbled gate

- Each wire key will now be a *vector* of keys, one held by each party
  - So $a_0 = (a_{01}, \ldots, a_{0n})$
- Define $\mathrm{Enc}_{a,\, b}(\mathrm{tag}; m) = m \oplus_i F_{a_i}(\mathrm{tag}) \oplus_i F_{b_i}(\mathrm{tag})$, where tag does not repeat, F is a block cipher
  - Here, tag = (gate number, row index)
  - Note: missing any key $\Rightarrow$ decryption impossible

Computing a garbled gate

- Each party P_i does:
  - For each wire w, choose w_{0i}, w_{1i}, and λ_{wi}
- Consider the following functionality (for garbling AND gate with wires a, b, c):
  - Input of P_i: $(a_{0i}, a_{1i}, \lambda_{ai}), (b_{0i}, b_{1i}, \lambda_{bi}), (c_{0i}, c_{1i}, \lambda_{ci})$
  - Set $a_0 = (a_{01}, \ldots, a_{0n})$, etc.
  - Set $\lambda_a = \oplus_i \lambda_{ai}$, etc.
  - Output garbled gate
  - Important that no party knows $\lambda_a, \lambda_b, \lambda_c$!

AND Gate
$Enc_{a0,\, b0}(00; \lambda_{00}, c_{\lambda_{00}})$
$Enc_{a0,\, b1}(01; \lambda_{01}, c_{\lambda_{01}})$
$Enc_{a1,\, b0}(10; \lambda_{10}, c_{\lambda_{10}})$
$Enc_{a1,\, b1}(11; \lambda_{11}, c_{\lambda_{11}})$

Computing a garbled gate

- Problem: unclear what is the depth of the circuit computing F!
- Solution: Each P_i also provides $F_{a_{0i}}(00)$, $F_{a_{0i}}(01)$, etc. as input to the gate-garbling functionality
  - Important here that there is no dependence on what is being encrypted

Computing a garbled circuit

- Input wires handled differently
  - If a is an input wire belonging to P_i, then only P_i chooses a_0, a_1, λ_a
  - If input on that wire is b, then P_i broadcasts $b \oplus \lambda_a$ and $a_{b \oplus \lambda_a}$ after garbling
- Note that all gates can be garbled *in parallel*
  - So the circuit for computing a garbled circuit has constant depth

BMR protocol

- Parties collectively generate a garbled circuit, and broadcast the keys for input wires
- Each party locally evaluates the garbled circuit
- For each output wire w, each P_i reveals λ_{wi}

Security?

- Prove security in “GMW-hybrid world”
- Secure for any $t < n$ corrupted parties

Properties of the protocol

- Runs in *constant* rounds
- O(|C|) “small” secure computations

Summary

- Three protocols for semi-honest secure computation
  - Garbled circuits
    - Constant-round, two-party
  - GMW
    - Linear-round, multi-party
  - BMR
    - Constant-round, multi-party

References

- Lindell-Pinkas, “A Proof of Yao’s Protocol for Secure Two-Party Computation,” https://eprint.iacr.org/2004/175
- Goldreich, “Foundations of Cryptography, vol. 2”
- Rogaway, “The Round Complexity of Secure Protocols,” PhD thesis

Thank you!