An Efficient Protocol for Oblivious DFA Evaluation and Applications

Saeed Sadeghian
University of Calgary

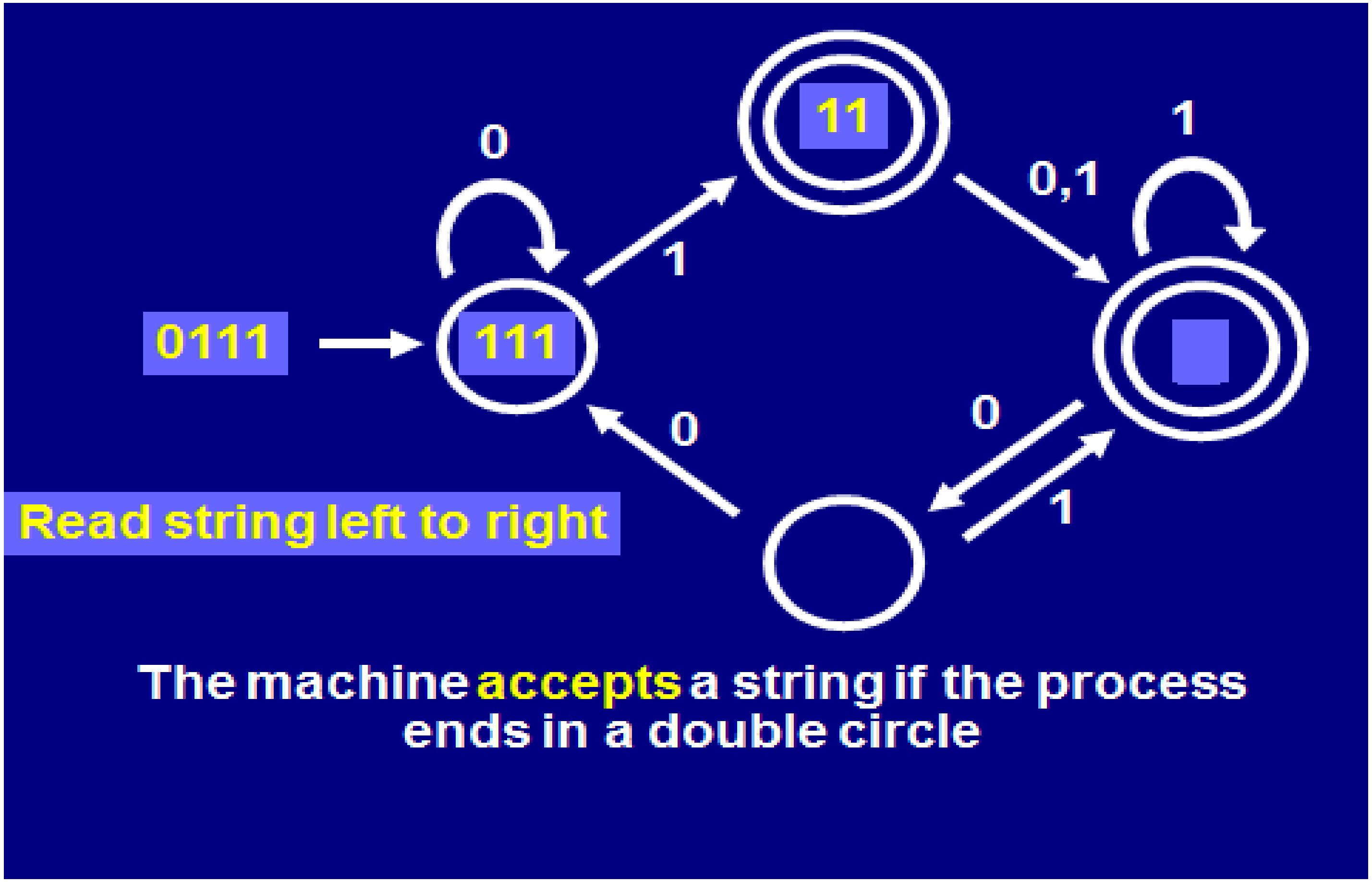
Joint work with Payman Mohassel and Salman Niksefat

Deterministic Finite Automaton

- Other names
  - Finite State Machines (FSM)
  - Finite State Automaton (FSA)
- A simple model of computation
  - Digital logic
  - Computer programs
  - Pattern matching

DFAs

[CMU, 453]


0
11
1
0,1
1
0111
111
1
0
0
Read string left to right
1
The machine accepts a string if the process ends in a double circle

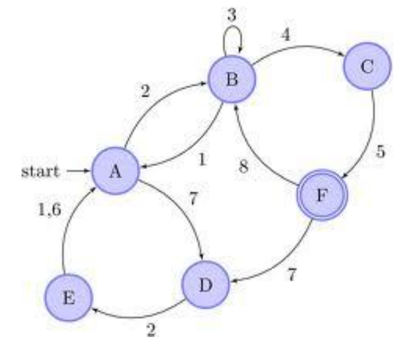
Formal DFA Representation

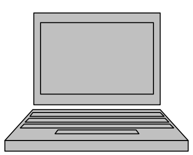
[CMU, 453]

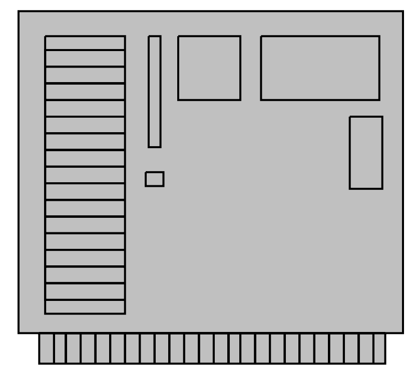
- $M = (Q, \Sigma, \delta, q_0, F)$
- Q is the set of states (finite)
- Σ is the alphabet (finite)
- $\delta: Q \times \Sigma \rightarrow Q$ is the transition function
- $q_0 \in Q$ is the start state
- $F \subseteq Q$ is the set of accept states

Oblivious DFA Evaluation

X= | a | b | e | a | h |

M = 



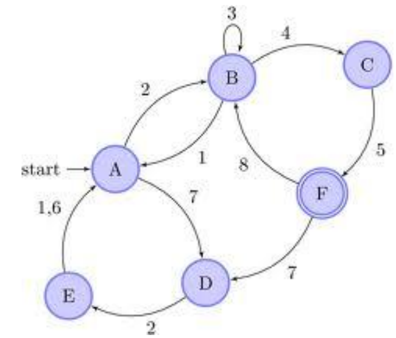


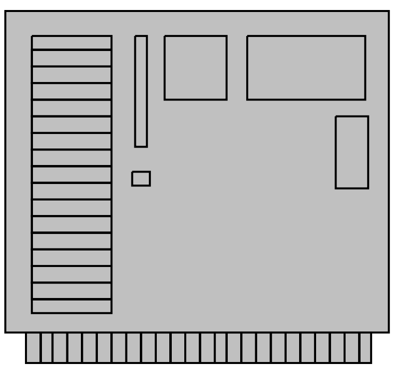
M(x)

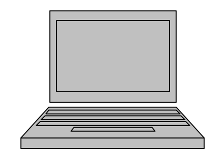
IDS application

Oblivious DFA Evaluation

X= a b e a h

M = 





M(x)

Pattern matching

Security Requirement

- Secure two-party computation
  - Hide the DFA
  - Hide the input string
  - Only reveal the output
- Malicious input holder
  - Guarantee Idea/real world simulation
- Malicious DFA holder
  - Guarantee input privacy

General-Purpose Solutions

- General two-party computation
  - Garbled circuit approach
- Drawbacks
  - Circuits get big quickly
  - Circuit creation is sometimes cumbersome
  - Not suitable for the client/server model

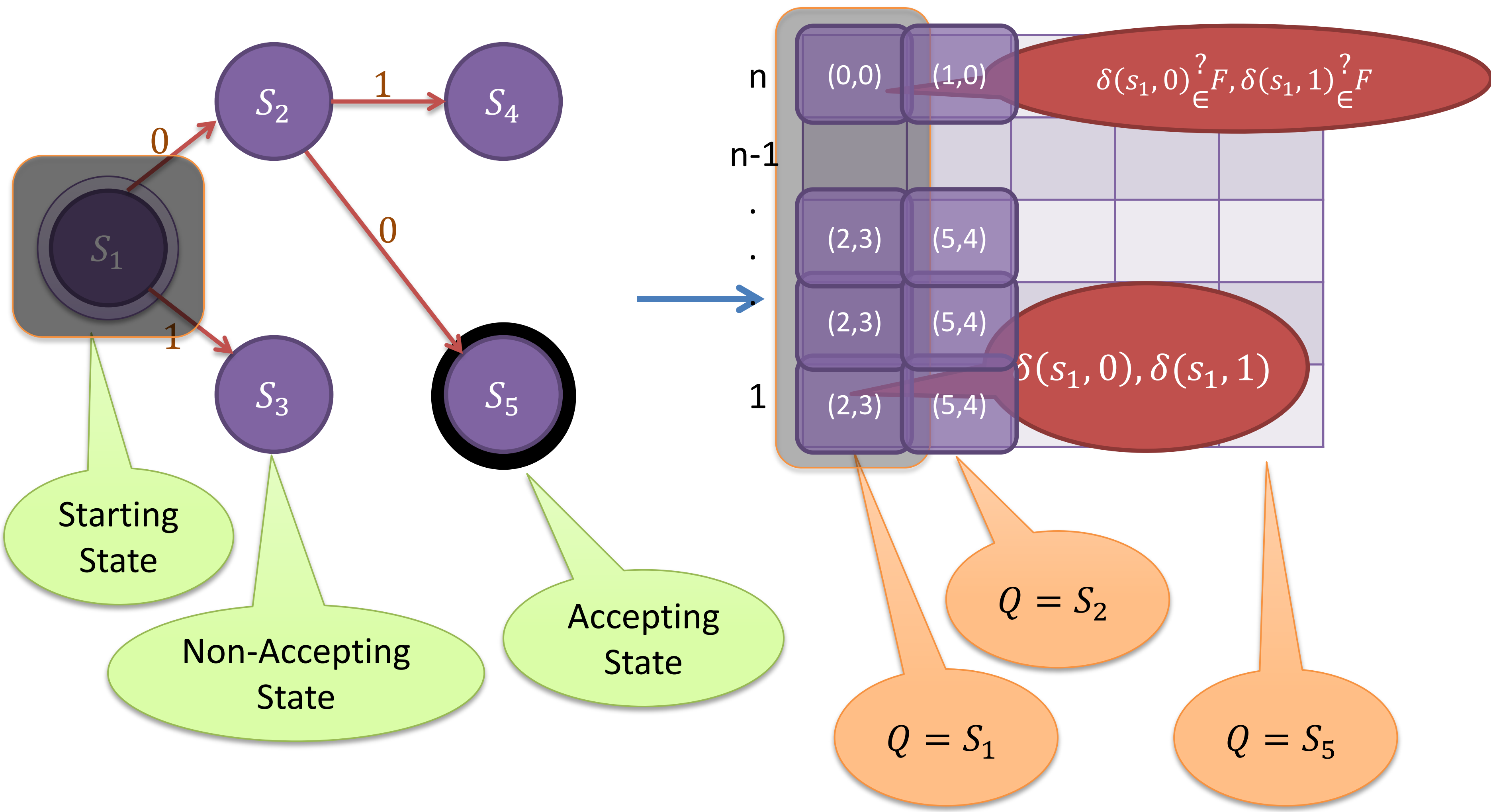
Special-Purpose Solutions

- A number of constructions
  - For oblivious DFA evaluation
  - Or oblivious branching program
- Drawbacks
  - DFA holder’s public-key ops is large
    - Proportional to DFA size

A Yao-like Approach to ODFA

- DFA holder
  - DFA → DFA matrix
  - DFA matrix → Garbled DFA matrix
    - Permute and encrypt the matrix
- Oblivious transfer
  - Receive garbled inputs
- Input holder
  - ***Evaluate/ungarble a single transit path***

(1) DFA → DFA Matrix



DFA Matrix

- DFA matrix size
  - n x $|Q|$ matrix
  - Each cell holds 2 index
  - $2n|Q|\log|Q|$ bits to represent
- DFA evaluation
  - Traverse a single transit path on the matrix

Problems

- ***We should hide the state number***
- We should make sure he has come from the correct last state
- We have to make sure he is not able to decrypt more than one cell in each row

(2) DFA Matrix → Permuted DFA Matrix

$n = 1: Per[1] = \{3,2,4,5,1\}$

$Q = 3$	$Q = 2$	…	$Q = 1$
$\delta(s_3, 0), \delta(s_3, 1)$	$\delta(s_2, 0), \delta(s_2, 1)$	…	$\delta(s_1, 0), \delta(s_1, 1)$

(1) Permute each row

$n = 0: Per[0] = \{2,1,5,4,3\}$

$Q = 2$	$Q = 1$	…	$Q = 3$
$\delta(s_2, 0), \delta(s_2, 1)$	$\delta(s_1, 0), \delta(s_1, 1)$	…	$\delta(s_3, 0), \delta(s_3, 1)$

(2,3)

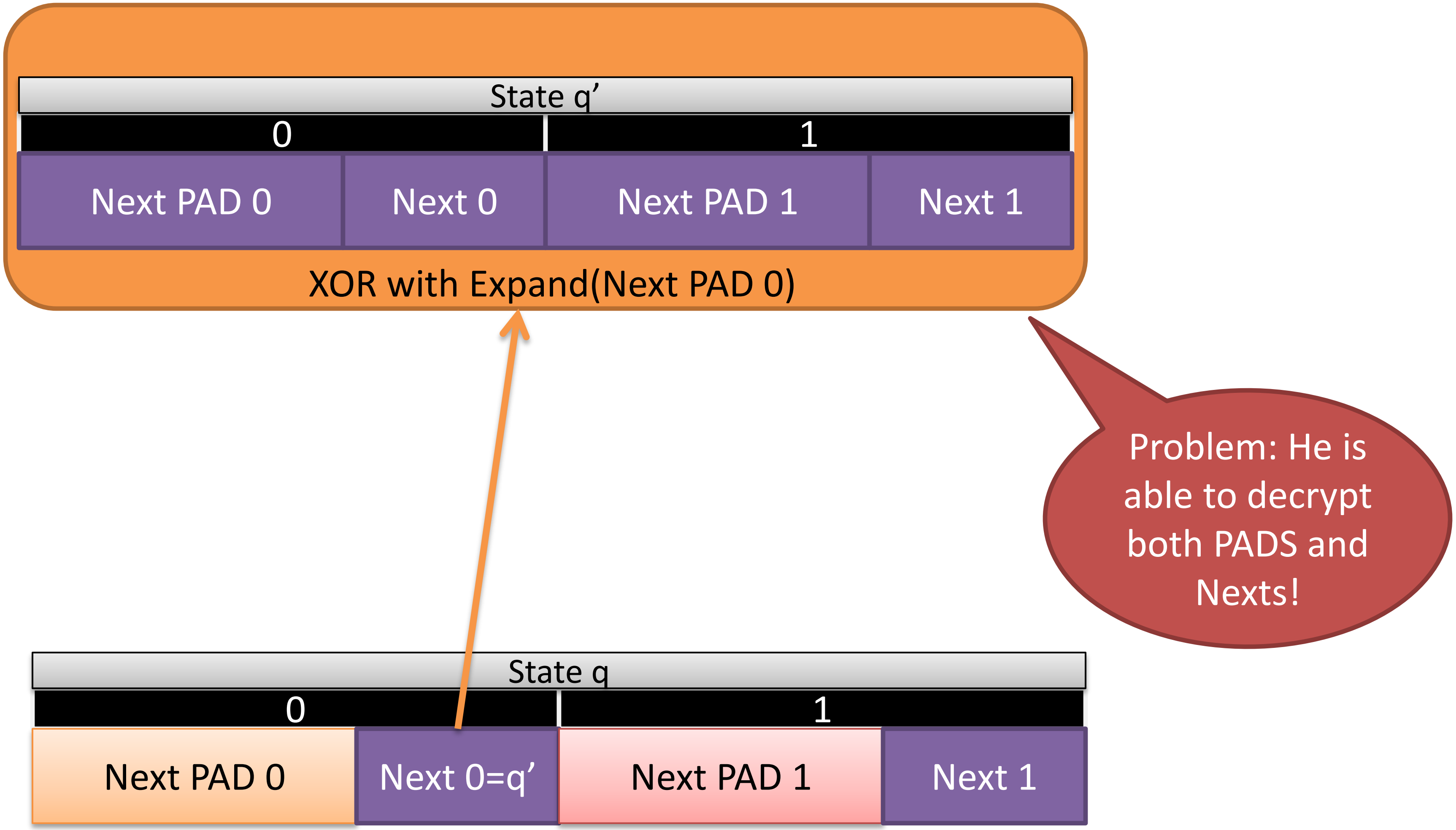
(2,1)

(2) Update the pointers

Problems

- We should hide the state number
- ***We should make sure he has come from the correct last state***
- We have to make sure he is not able to decrypt more than one cell in each row

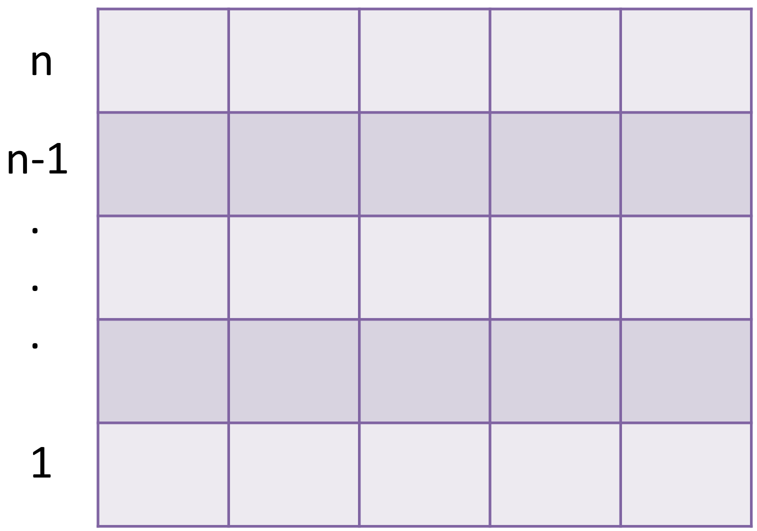
(3) Encrypt with PAD Matrix



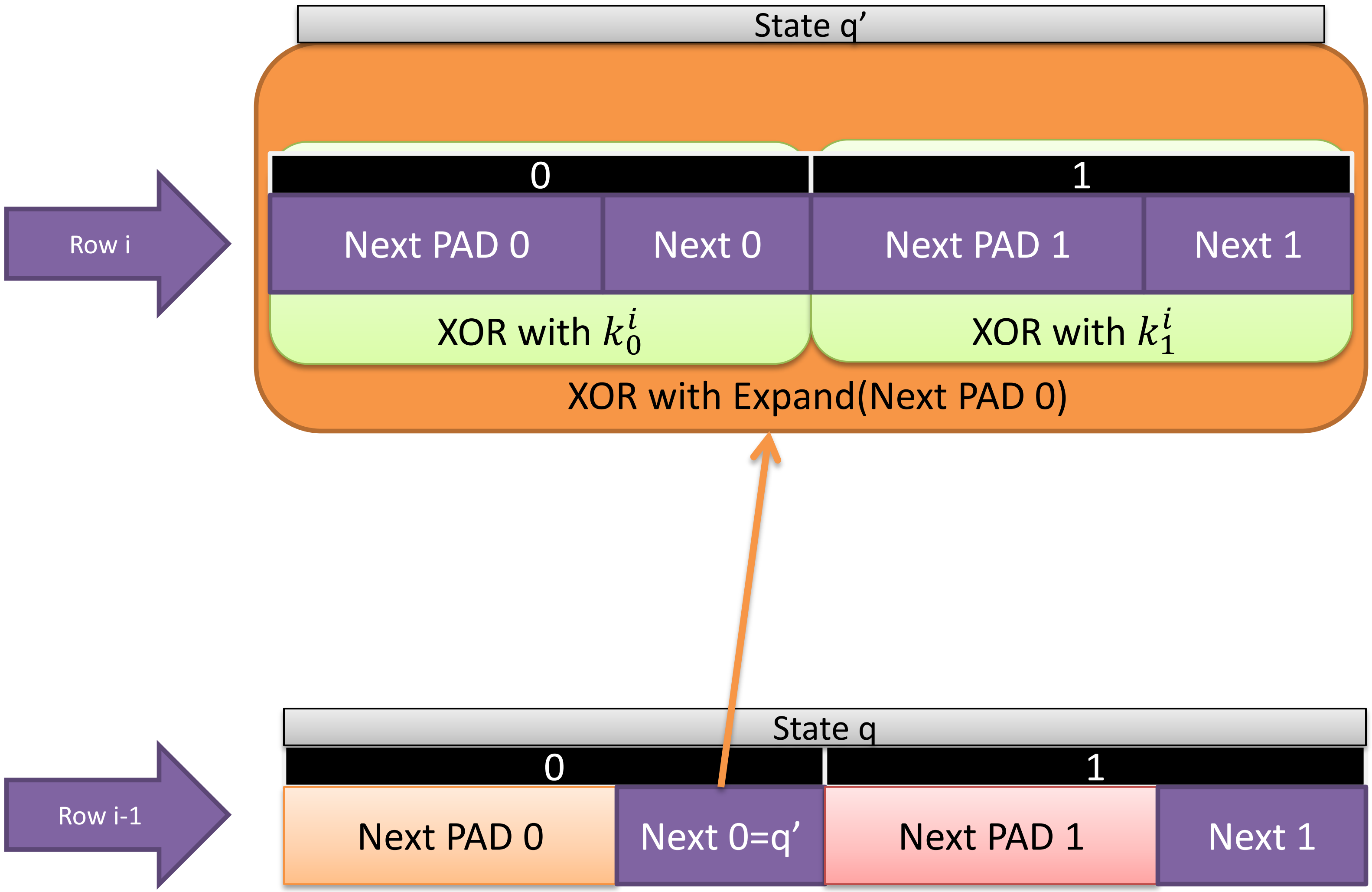
Problems

- We should hide the state number
- We should make sure he has come from the correct last state
- ***We have to make sure he is not able to decrypt more than one cell in each row***

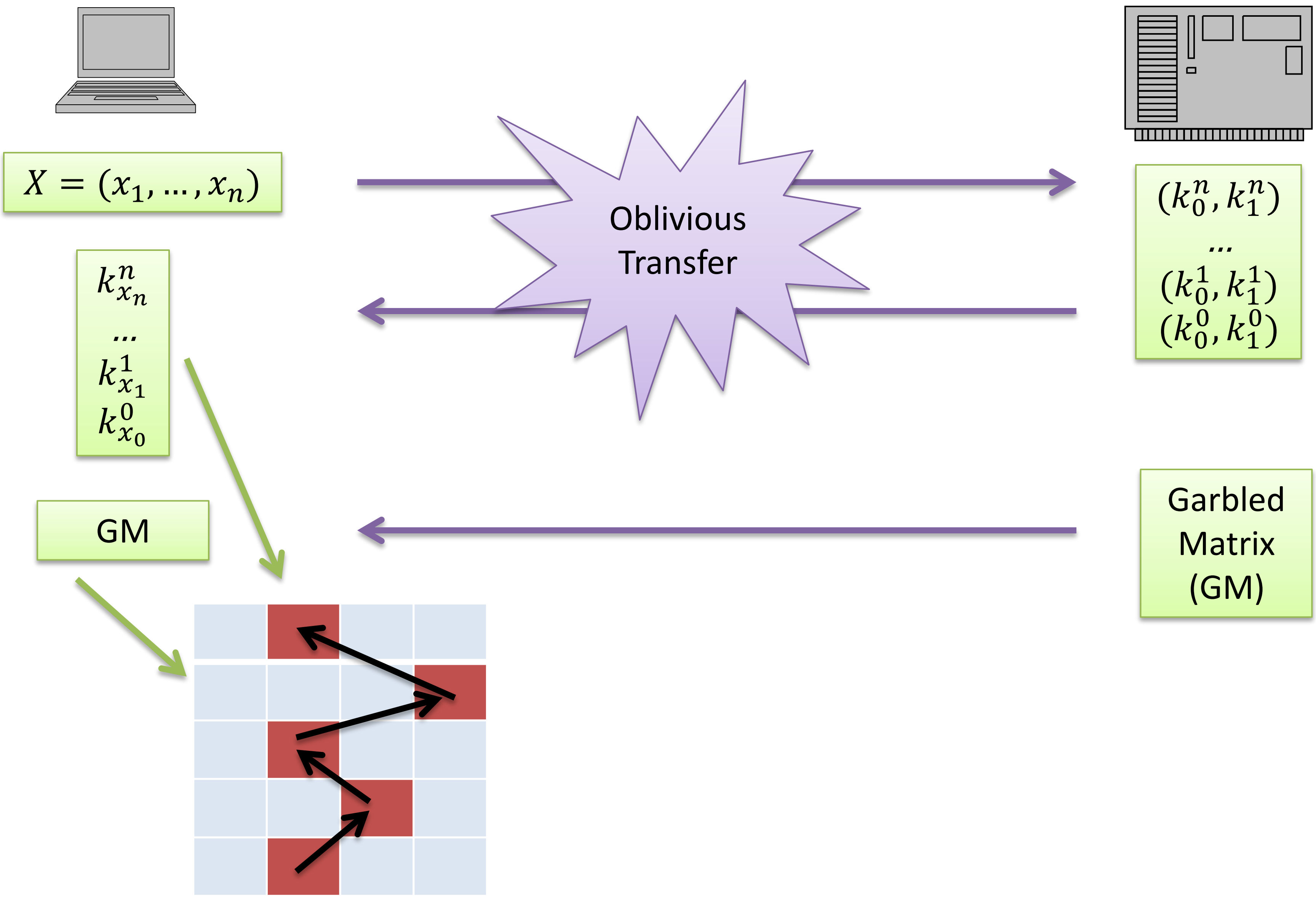
(4) Encrypt with Keys



(4) Encrypt with Keys



Protocol



$X = (x_1, \dots, x_n)$
$k^n_{x_n}$
...
$k^1_{x_1}$
$k^0_{x_0}$
GM
Oblivious Transfer
(k_0^n, k_1^n)
...
(k_0^1, k_1^1)
(k_0^0, k_1^0)
Garbled Matrix (GM)

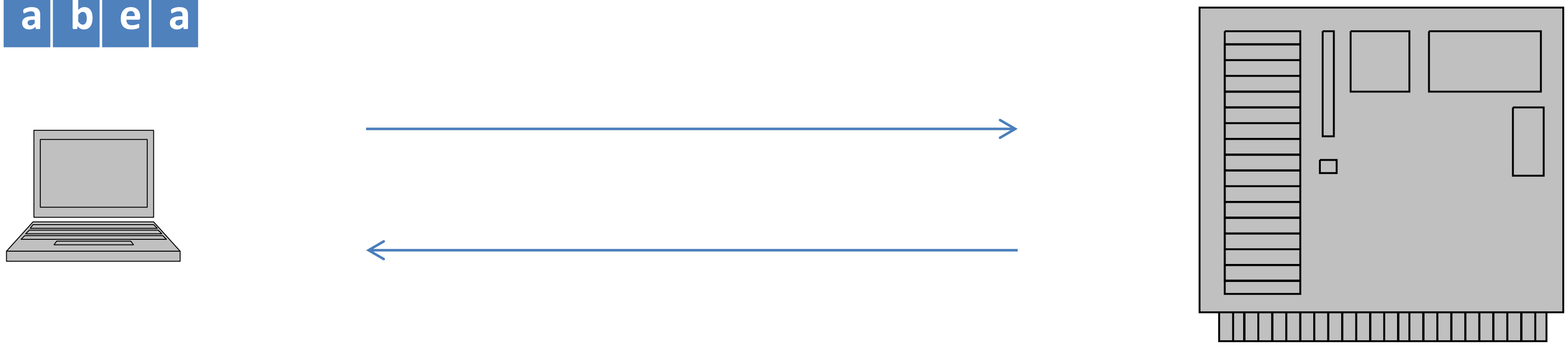
Complexity

- Public-Key ops
  - $O(n)$ for both parties
  - Can be reduced to k using OT extension
- DFA holder's symmetric-key ops
  - $n|Q|$ PRG evaluations
- Input holder's symmetric-key ops
  - n PRG evaluations
- Communication
  - $2n|Q|(\log|Q|+k)$ bits
- Round complexity
  - 1 round

Secure Pattern Matching

T = | a | b | d | r | g | e | a | b | e | a | t |

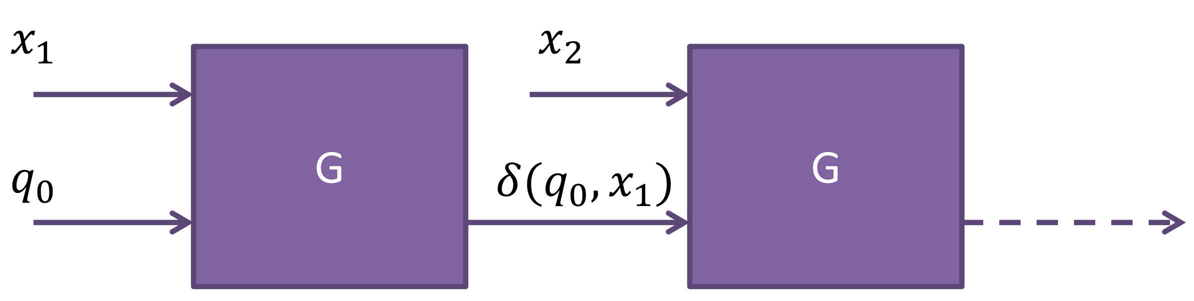
p = | a | b | e | a |



- Does a pattern p exist in text T
- Locations of occurrences of p in T
- Number of occurrences of p in T

A Different Presentation of Protocol

- Pointed out by reviewers
- Can be viewed as a generalization of Yao's Garbled Circuit Protocol
- Each gate takes non-boolean inputs and returns non-boolean outputs

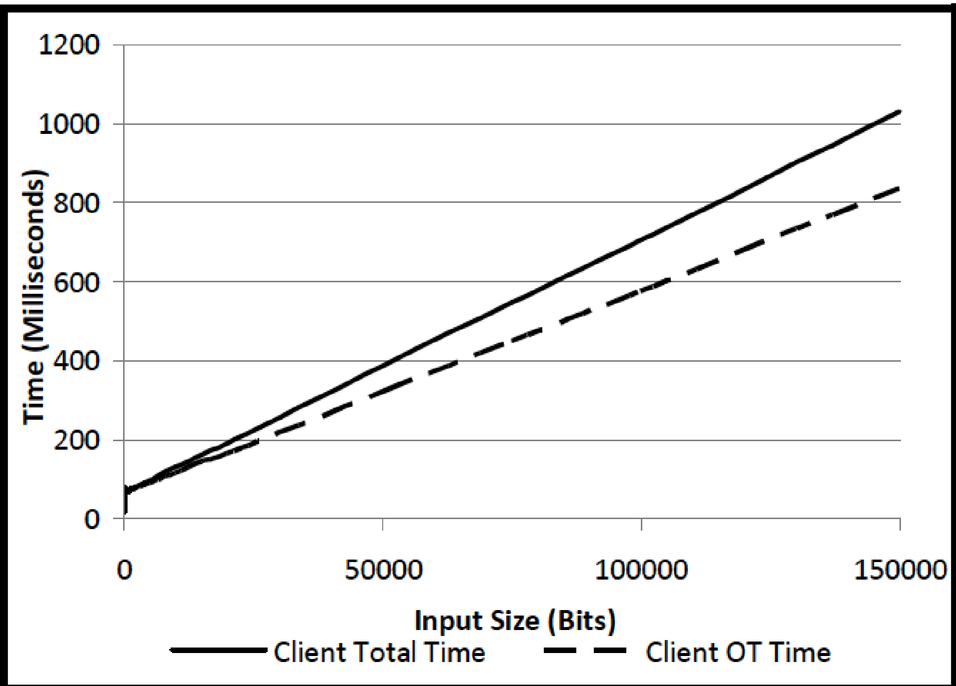


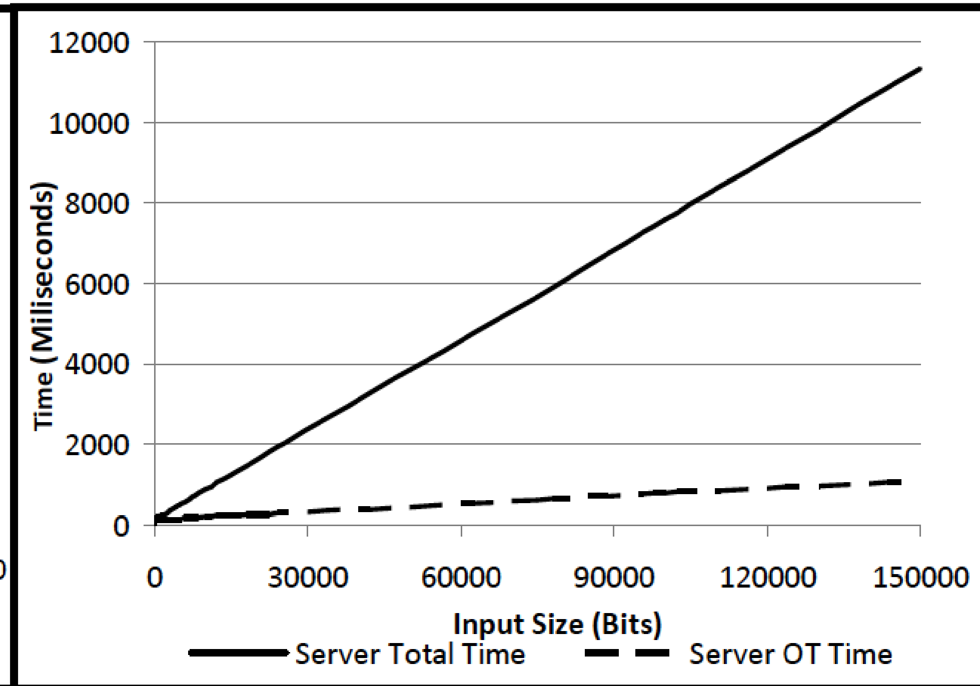
Comparison

	Round Complexity	client Computations		server Computations		Communication Complexity
		Asymmetric	Symmetric	Asymmetric	Symmetric	
Troncoso [21]	$O(n)$	$O(n\|Q\|)$	None	$O(n\|Q\|)$	$O(n\|Q\|)$	$O(n\|Q\|k)$
Frikken [3]	2	$O(n+\|Q\|)$	$O(n\|Q\|)$	$O(n+\|Q\|)$	$O(n\|Q\|)$	$O(n\|Q\|k)$
Gennaro [4]	$min(O(\|Q\|), O(n))$	$O(n\|Q\|)$	None	$O(n\|Q\|)$	None	$O(n\|Q\|k)$
Yao's protocol [22]	1	$O(n)$	$O(n\|Q\| \log \|Q\|)$	$O(n)$	$O(n\|Q\| \log \|Q\|)$	$O(n\|Q\|k)$
Ishai [12]	1	$O(n)$	None	$O(n\|Q\|)$	None	$O(kn^2)$
Protocol 1(PRG)	1	$O(n)$	$O(n)$	$O(n)$	$O(n\|Q\|)$	$O(n\|Q\|k)$
Protocol 1 (PRG+Extended OT)	1.5	$O(k)$	$O(n)$	$O(k)$	$O(n\|Q\|)$	$O(n\|Q\|k)$

Implementation

- Complete C++ implementation
- Experiments on Intel Core i7, 4GB RAM





Future Work

- IDS DFAs are not too dense
  - Can we do better?
- We will do many DFA evaluations
  - Better batch evaluations?
  - Better communication particularly
  - (reusing part of the DFA matrix?)

Secure Multi-Party Computation of Boolean Circuits with Applications to Privacy in On-Line Marketplaces

SEUNG GEOL CHOI

UNIVERSITY OF MARYLAND

Joint work with

Jonathan Katz (University of Maryland)

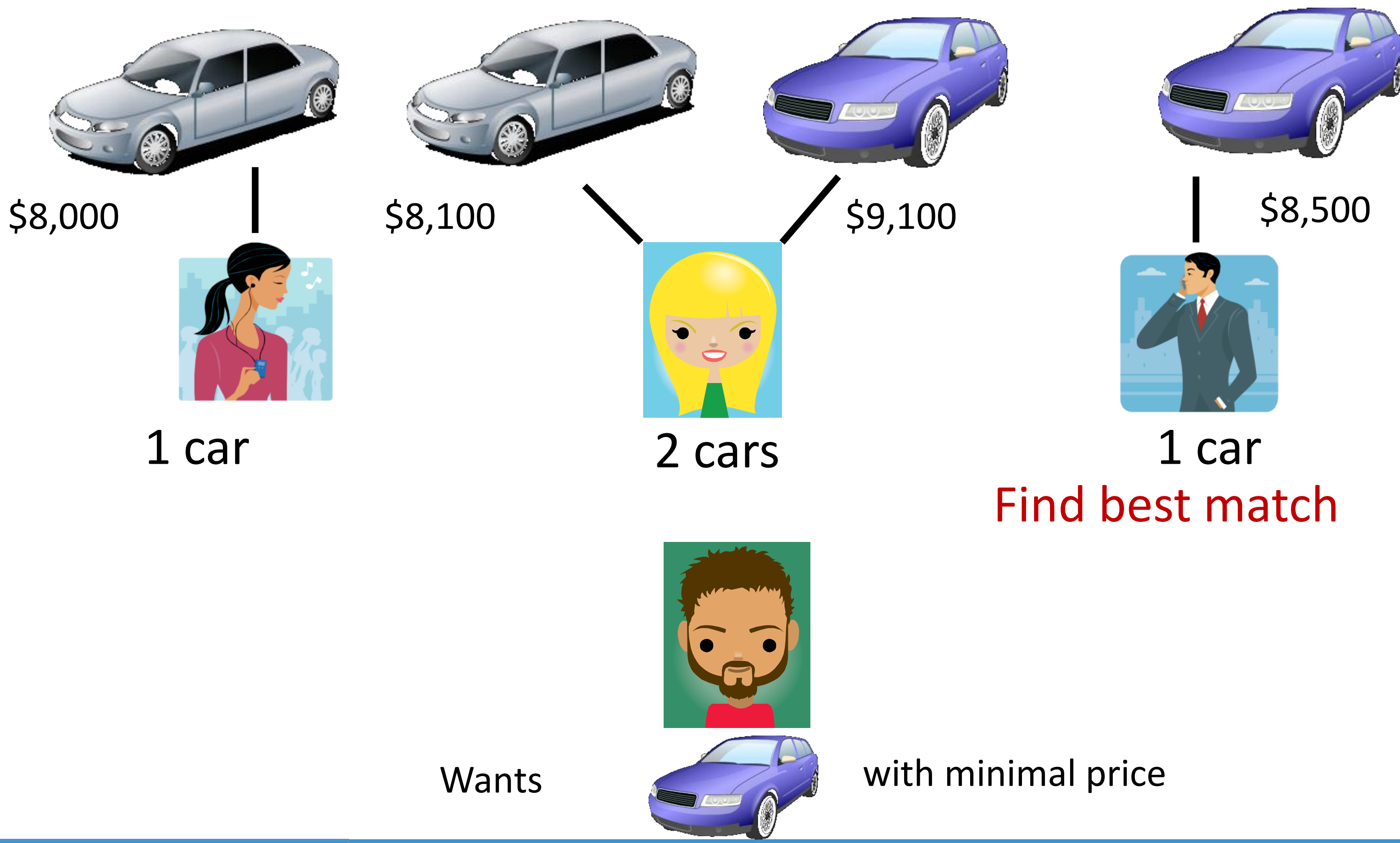
Kyung-Wook Hwang, Tal Malkin, Dan Rubenstein (Columbia University)

Session ID: CRYP-403

Session Classification: Advanced

RSACONFERENCE2012

Motivation: Online Marketplace



Online Marketplace

- Participants
  - Providers: have resources and associated metric
  - Customer: has preference

- Desired result
  - For each resource r, compute its score according to the input of the customer and the providers
  - Output the resource with best score

Example: P2P Content Distribution

- Resources = Peers
- Providers: ISPs
  - Know bandwidth info of peers
- Customer
  - Knows which peer has the desired file
  - Wants to find a suitable peer with highest bandwidth
- Result
  - Score for a peer: if the peer has the file, output its bandwidth; otherwise, output 0
  - Output: the peer with the highest score

Other Examples

- Cloud computing
  - Find the best-quality cloud service within price limit
  - Find the cheapest cloud service of desired quality

- Mobile social network
  - Find the closet user within enough matching interests
  - Find the user with most matching interests within a certain distance.

Privacy in Online Marketplaces

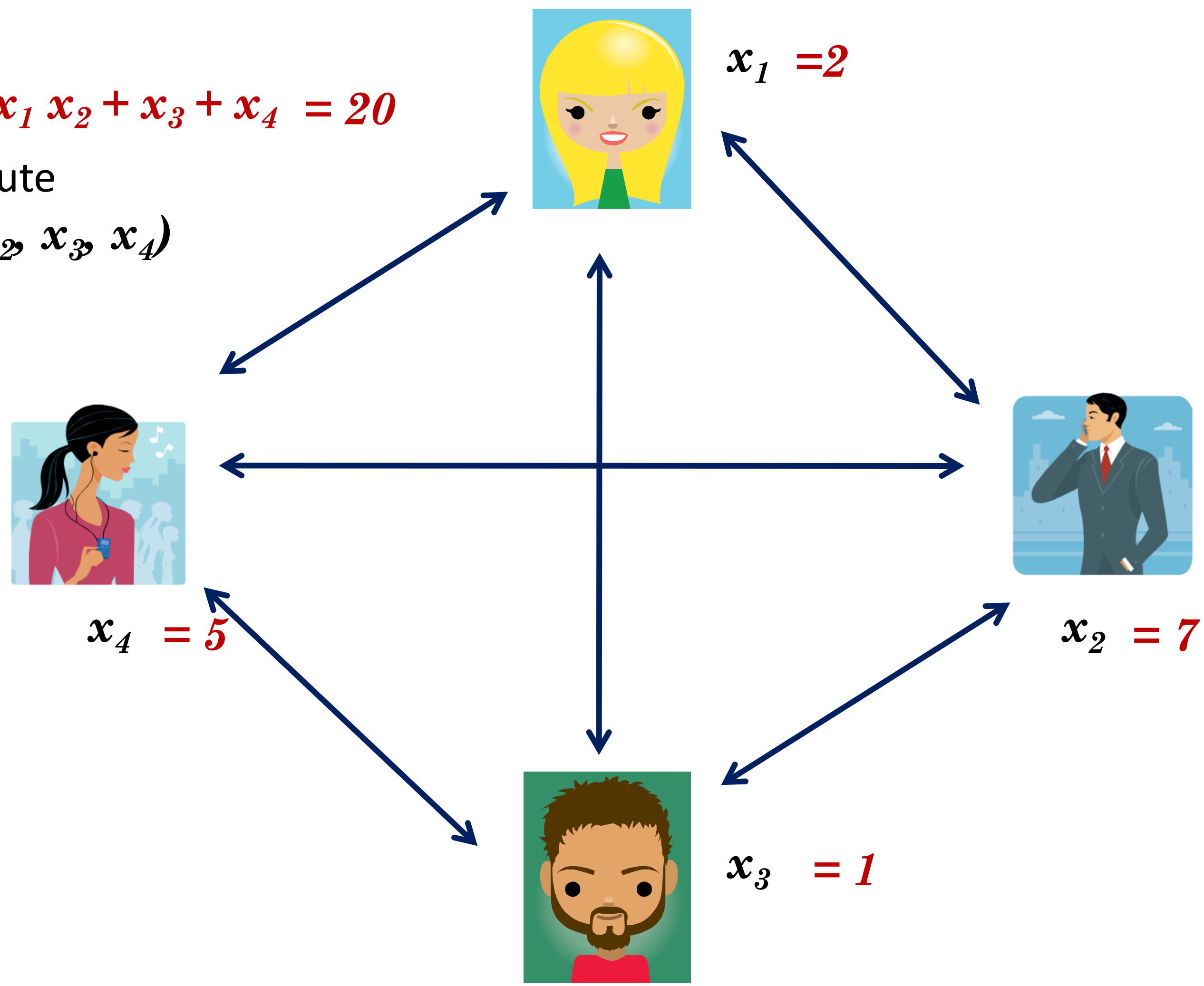
- Privacy
  - The providers and the customer should learn nothing about anyone's inputs (beyond the output)
  - Semi-honest security: corrupted parties follow the protocol honestly, but try to infer secret information from the protocol transcript.
- Protocol?
  - One could attempt to construct a specific protocol…
  - How well would a *generic* secure multi-party computation (MPC) protocol work?

Secure Multi-Party Computation

E.g., $f = x_1\, x_2 + x_3 + x_4 \;= 20$

Compute

$f(x_1, x_2, x_3, x_4)$



Generic Solutions?

- “Generic” = a protocol for any function, specified as a boolean/arithmetic circuit

- Good news: generic solutions exist

- Bad news: relatively inefficient (?)

However, in the Past Few Years

- Growing interest in research community
  - Optimizing efficiency of protocols
- Increased capability of modern computers

Several generic solutions have been implemented

Two Types of Generic MPC Solutions

- Boolean circuits
  - A circuit with boolean gates e.g., XOR and AND.
  - Input of each party: represented as bits

- Arithmetic circuits
  - A circuit with addition/multiplication gates in some field, e.g., GF(p) or GF(2^n)
  - Input of each party: an element in the given field

Boolean or Arithmetic?

Function	Boolean (Bit)	Arithmetic (Field)
Statistics (e.g., average)	Large circuit	Small circuit
Comparison (e.g., less than)	Small circuit	Large circuit

Boolean circuits better suited for addressing the private marketplaces problem

Previous Work on MPC Solutions

	Circuit	Language	Circuit Scalability	# Corrupted parties
FairplayMP [BNP08]	Boolean	Java	No (~4000 gates)	< n/2
VIFF [DGKN09]	Arithmetic	Python	Yes	< n/2
SEPIA [BSMD10]	Arithmetic	Java	Yes	< n/2
Ours	Boolean	C++	Yes	< n

Not satisfactory for our purpose

Our Contributions

- We provide the first scalable implementation of multi-party computation for *boolean* circuits, with *optimal* resilience

- We apply our implementation to the problem of online marketplaces
  - Performance better than what is obtained using previous solutions (VIFF, SEPIA)

- Another indication that generic secure MPC can be useful in solving practical problems

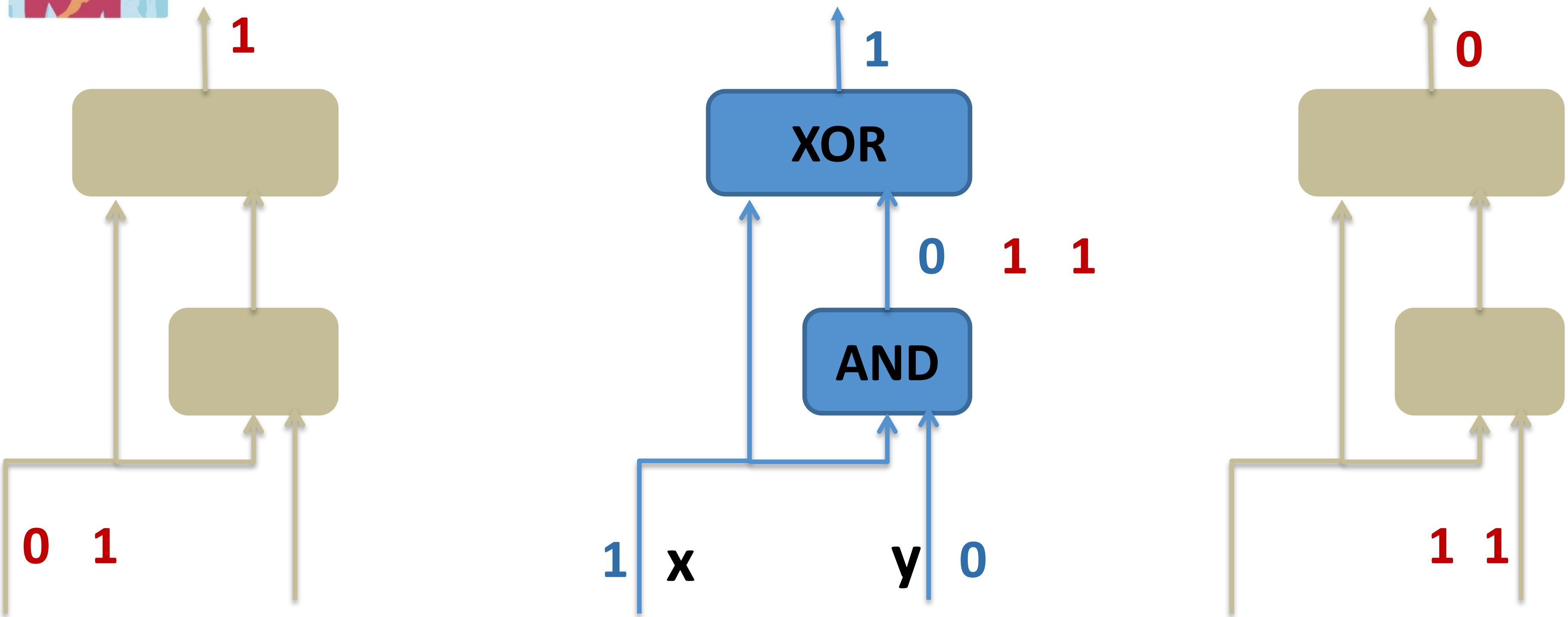
Our Generic MPC Solution

Overview of the Protocol

- We implement the [GMW87] protocol
- The function is given as a boolean circuit
  - With XOR and AND gates
- Evaluate the circuit in a gate-by-gate manner
  - Invariant: the actual value of each wire is secret-shared.

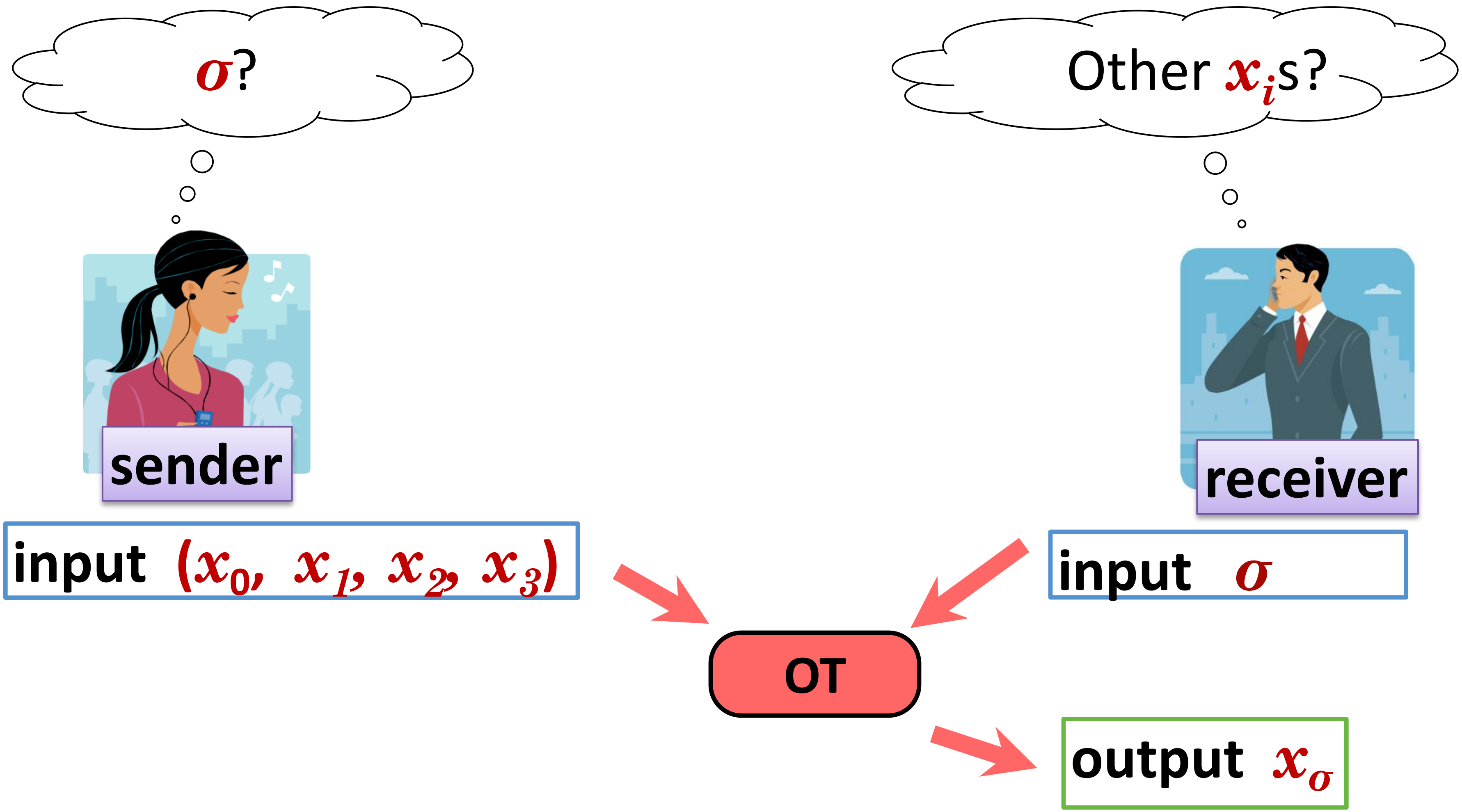
Overview of the GMW Protocol

$$x > y = x(1 \oplus y) = x \oplus xy$$



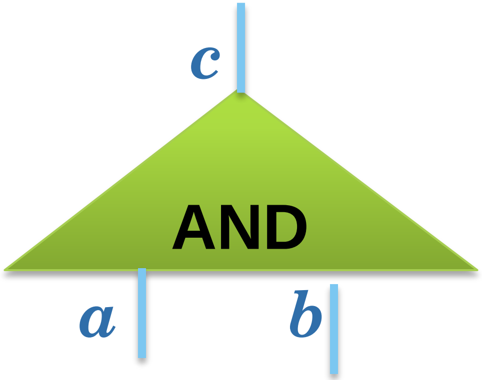
Evaluate the circuit gate-by-gate .
The value of each wire is secret-shared.

1-out-of-4 Oblivious Transfer (OT)



GMW - Evaluating AND gates

a_1, b_1

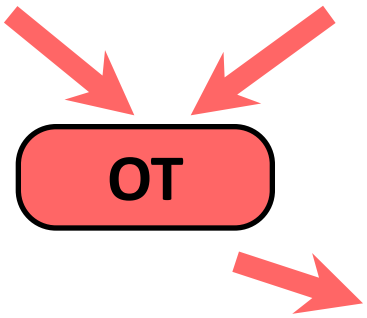


a_2, b_2

Run oblivious transfer

$$x_0 = (a_1+0)(b_1+0) + r$$
$$x_1 = (a_1+0)(b_1+1) + r$$
$$x_2 = (a_1+1)(b_1+0) + r$$
$$x_3 = (a_1+1)(b_1+1) + r$$

$$\sigma = 2a_2 + b_2$$



x_σ

$c_1 = r$

$c_2 = x_\sigma$

GMW Protocol: Multi-Party Setting

- Input wires: XOR of all shares are the actual value.
- XOR gate: same as before (i.e., $c_i = a_i + b_i$)
- AND gate: use OT between all pairs of parties
  - Details omitted

Implementation of the GMW Protocol

- Critically depends on efficiency of OT protocol

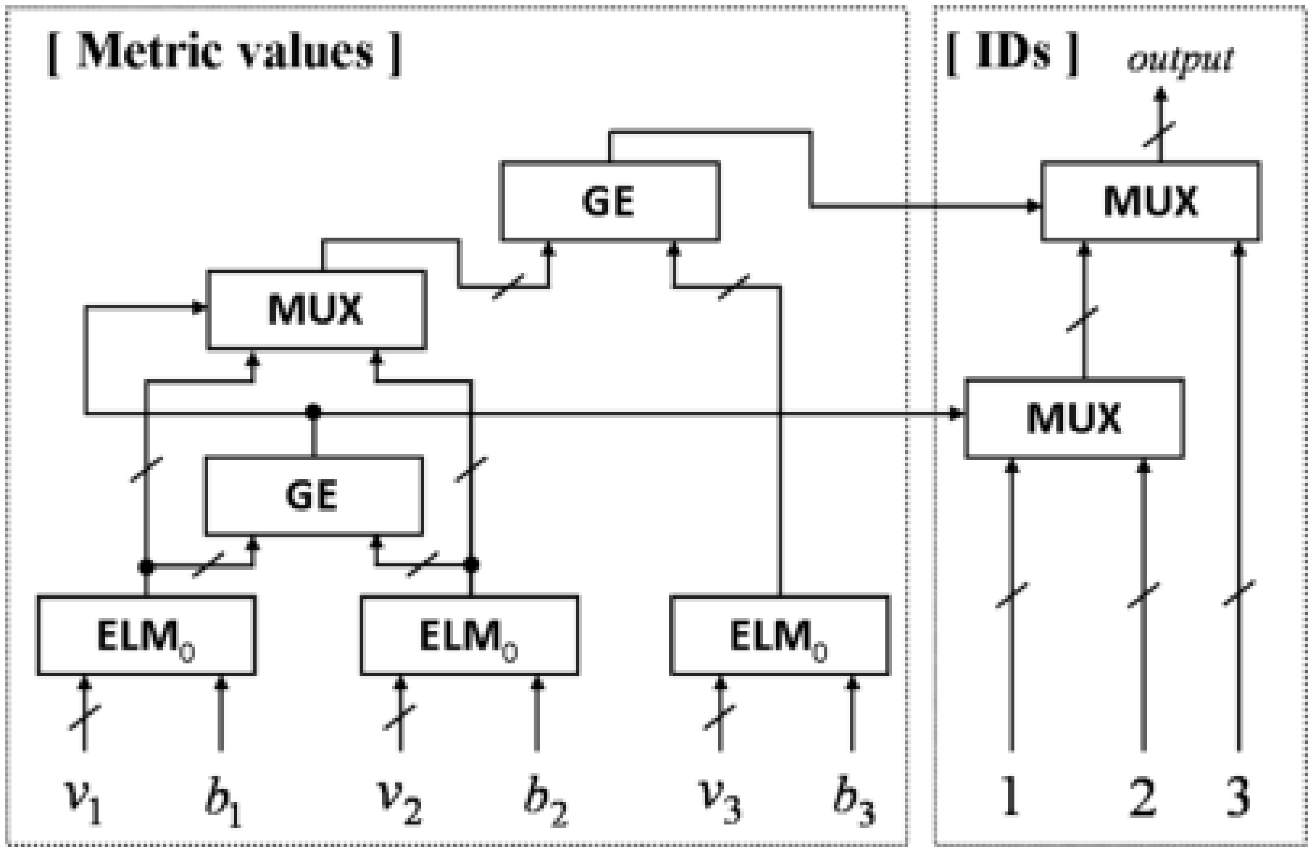
- Basic OT [NP01]
  - Multi-threading: two-threads for each pair-wise OT
  - Number-theory package: NTL http://shoup.net (modified for MT)

- OT extension [IKNP03]
  - Several (e.g., 80) basic OTs with long inputs → many bit OTs
  - Small overhead: four hash functions per OT
  - Use SHA-1 implementation from PolarSSL

- OT preprocessing [Bea95]
  - Preprocess OTs on random input
  - Use them for OTs on actual input: tiny overhead (a few bits)

Application to Online Marketplaces

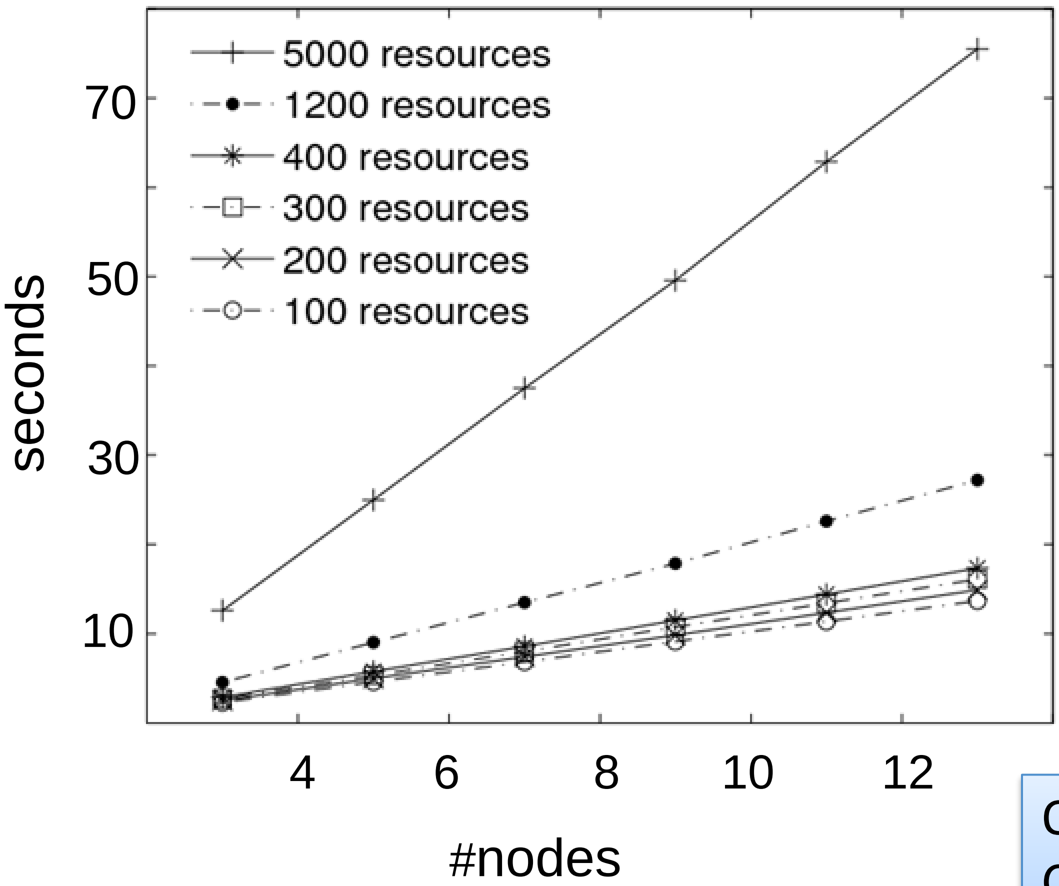
Circuit: P2P Content Distribution



$$\sharp \text{ AND gates} \approx 3r\ell + r\log r$$

$$(r = \sharp \text{ resources}, \quad \ell = \text{bit-length of an integer})$$

Experiments in LAN: P2P Content Distribution

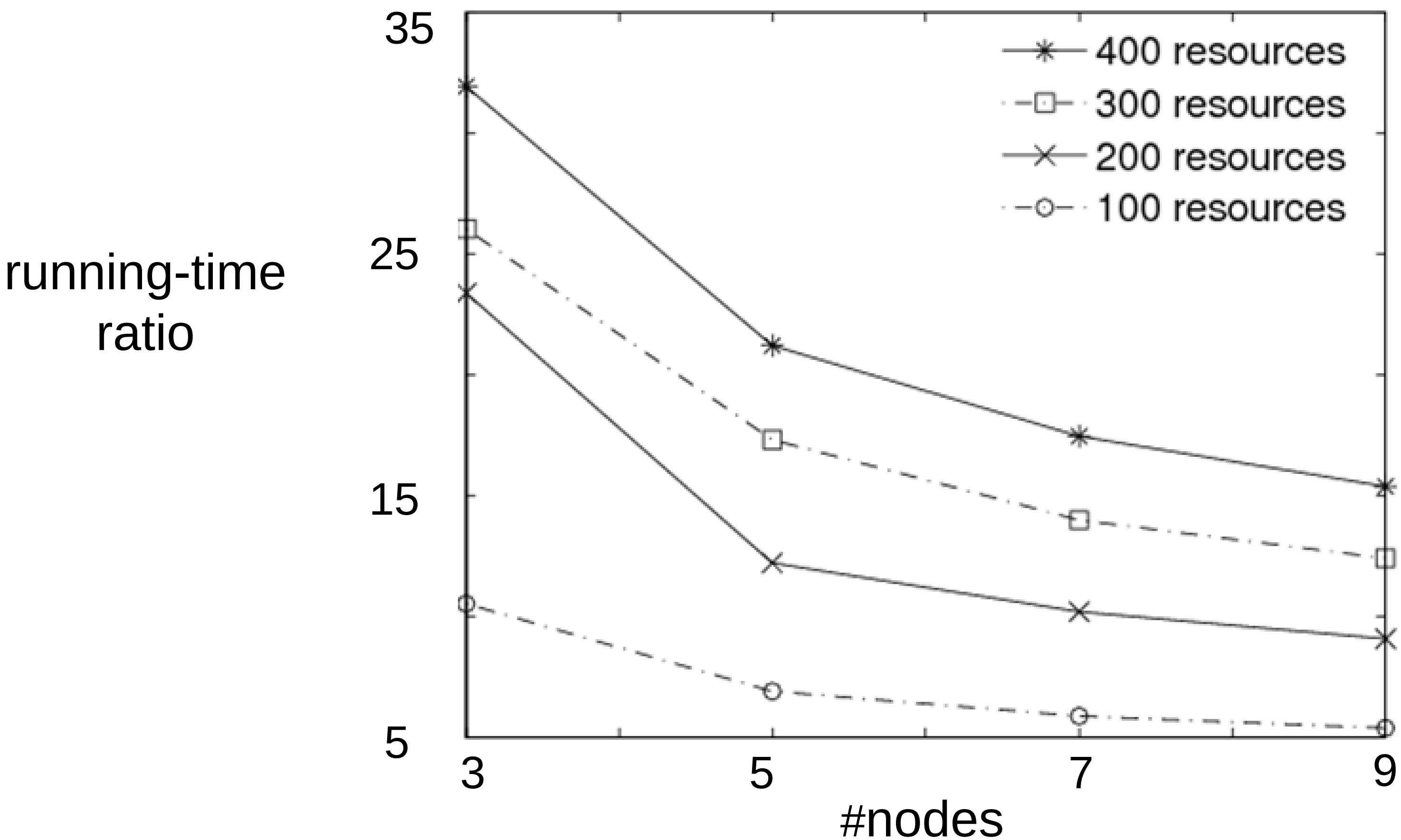


Running Time $(\ell = 16)$

- Running time linear in # nodes and (almost) in # resources
- Marginal time per AND gate:
  - 50 μs (3 nodes)
  - 340 μs (13 nodes)

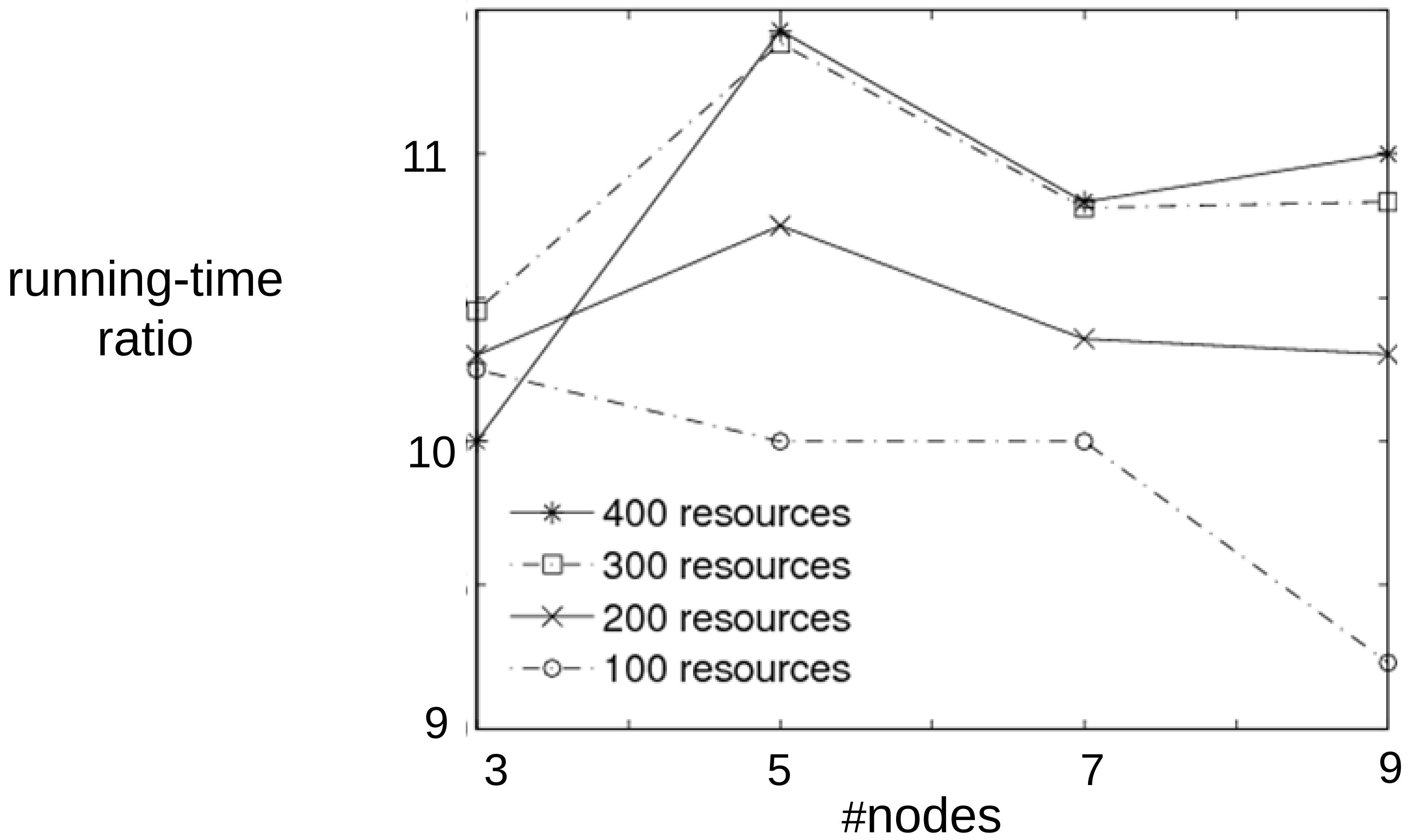
OS: Linux
CPU: Intel Xeon 2.80 GHz (dual-core)
RAM: 4GB

Running-Time Ratio: VIFF/Ours



Our implementation is 10-30x faster

Running-Time Ratio: SEPIA/ours



Our implementation is ~10x faster

Experiments in PlanetLab

- Similar results
  - With somewhat bigger deviation

- Details are in the paper

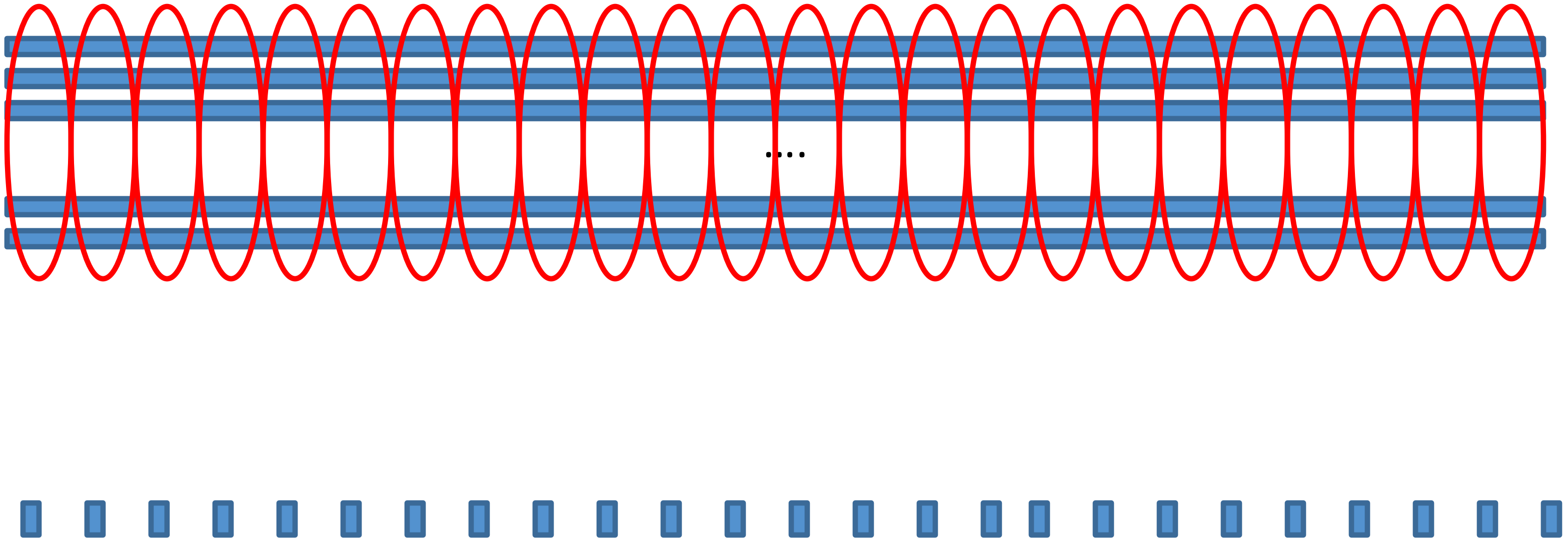
Summary

- Generic MPC implementation
  - Boolean circuit representation, optimal corruption thereshold
  - Source code: http://www.ee.columbia.edu/~kwhwang/projects/gmw.html
- Application to privacy in online marketplaces
- Generic MPC can be practical
  - Explore generic solutions before designing new protocols

Thank you

OT Extension [IKNP03,LXX05]

- Several long string OTs → many bit OTs



Very efficient: four additional hashes per OT

OT Preprocessing [Bea95]

- bit OTs on random input → bit OTs on actual input

(r_0, r_1, r_2, r_3) → OT ← 2

OT → r_2

input (x_0, x_1, x_2, x_3)

input 1

3

$(r_3+x_0,\ r_2+x_1,\ r_1+x_2,\ r_0+x_3)$

Two-Party Computation? (Not in This Talk)

- Initial work
  - Fairplay [MNPS04]
  - Rather slow and not scalable
- Subsequent work
  - Improves performance and scalability
  - [LPS08,PSSW09,HEKM11, M11]
- With semi-honest security
  - Corrupted parties follow the protocol honestly, but try to infer secret information from the protocol transcript.

GMW - Evaluating AND gates

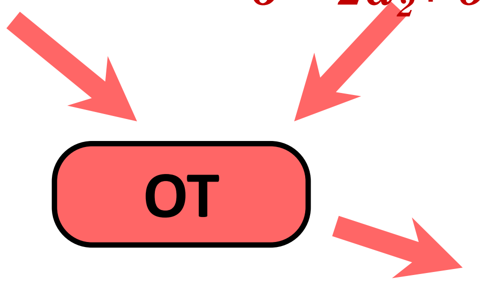
$x_0 = (a_1+0)(b_1+0) + r$

$x_1 = (a_1+0)(b_1+1) + r$

$x_2 = (a_1+1)(b_1+0) + r$

$x_3 = (a_1+1)(b_1+1) + r$

$\sigma = 2a_2 + b_2$



x_σ

$$c_1 + c_2 = r + x_\sigma = \boxed{(a_1+a_2)(b_1+b_2) = ab = c}$$

Check $x_\sigma = (a_1+a_2)(b_1+b_2) + r$

$a_2 = 0,\ b_2 = 0{:}\quad \sigma = 0,\quad x_0 = (a_1+a_2)(b_1+b_2) + r$

$a_2 = 0,\ b_2 = 1{:}\quad \sigma = 1,\quad x_1 = (a_1+a_2)(b_1+b_2) + r$

$a_2 = 1,\ b_2 = 0{:}\quad \sigma = 2,\quad x_2 = (a_1+a_2)(b_1+b_2) + r$

$a_2 = 1,\ b_2 = 1{:}\quad \sigma = 3,\quad x_3 = (a_1+a_2)(b_1+b_2) + r$