Multi-Party Indirect Indexing and Applications

Matthew Franklin   Mark Gondree   Payman Mohassel

Department of Computer Science
University of California, Davis

December 4, 2007
Asiacrypt 2007
Kuching, Sarawak, Malaysia

Outline

Outline

Summary

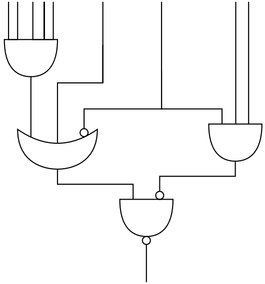
Our main result:

- an efficient multiparty generalization of Naor-Nissim circuits with look-up-tables (LUT)
- a useful (and efficient) generalization of oblivious transfer

Outline

Motivation: using RAM machines in secure MPC

Poly-log reduction of RAM machines to circuits



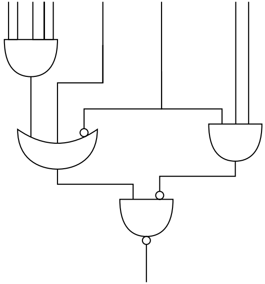
Circuit

Motivation: using RAM machines in secure MPC

Poly-log reduction of RAM machines to circuits

not known

Thus, RAM machines may be much more efficient than circuits



Circuit

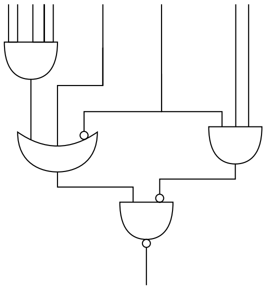
Motivation: using RAM machines in secure MPC

Poly-log reduction of RAM machines to circuits

not known

Thus, RAM machines may be much more efficient than circuits

Poly-log reduction of RAM machines to *circuits with look-up tables* (LUT)



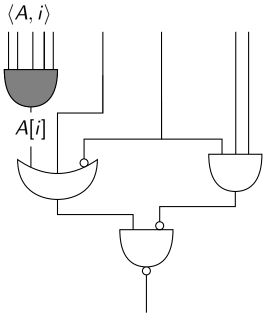
Circuit

Poly-log reduction of RAM machines to circuits

not known

Thus, RAM machines may be much more efficient than circuits

Poly-log reduction of RAM machines to *circuits with look-up tables* (LUT)



Circuit
with LUT

Motivation: using RAM machines in secure MPC

Poly-log reduction of RAM machines to circuits

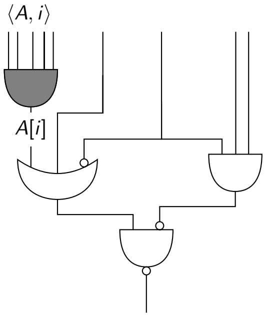
not known

Thus, RAM machines may be much more efficient than circuits

Poly-log reduction of RAM machines to *circuits with look-up tables* (LUT)

known

for any RAM machine M running in time $T(n)$ using space $S(n)$, $\exists$ series of circuits with LUT $\{C_n\}_{n\in\mathbb{N}}$ computing f_M, where C_n is size $T(n)\mathrm{polylog}(S(n))$



Circuit with LUT

Motivation: using RAM machines in secure MPC

- Given a private circuit w/ LUT construction
    - simulate a RAM machine

Motivation: using RAM machines in secure MPC

- Given a private circuit w/ LUT construction
  - simulate a RAM machine
  - what more:
    - a RAM machine where reads are oblivious

Motivation: using RAM machines in secure MPC

- Given a private circuit w/ LUT construction
    - simulate a RAM machine
    - what more:
        - a RAM machine where reads are oblivious

Here, *oblivious* means

time and location is independent of the computation's inputs/randomness

Motivation: using RAM machines in secure MPC

- Given a private circuit w/ LUT construction
  - simulate a RAM machine
  - what more:
    - a RAM machine where reads are oblivious
    - a RAM machine where writes are oblivious (via [NN01])

Here, *oblivious* means

time and location is independent of the computation's inputs/randomness

Motivation: using RAM machines in secure MPC

- Given a private circuit w/ LUT construction
  - simulate a RAM machine
  - what more: we get a simulation of an oblivious RAM machine
    - a RAM machine where reads are oblivious
    - a RAM machine where writes are oblivious (via [NN01])

Here, *oblivious* means

time and location is independent of the computation's inputs/randomness

Motivation: Applications for RAM Machines

Private computation via simulation of a RAM machine appropriate for

Motivation: Applications for RAM Machines

Private computation via simulation of a RAM machine appropriate for

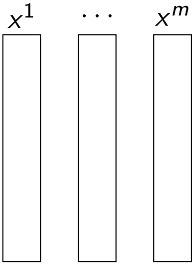
- any problem where a large array of data must be used and
  - only some of the data is ever accessed, or
  - the access pattern leaks information

Example: Private Sampling [IMSW07]

Input

- an m-ary function f
- m inputs of length n,

$$x_j^i = j\text{-th element of } x^i$$



Example: Private Sampling [IMSW07]

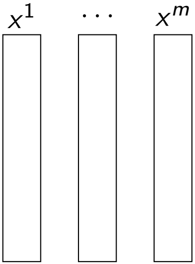
Input

- an m-ary function f
- m inputs of length n,

$$x^i_j = \ j\text{-th element of } x^i$$

Output

$f(x^1_r, \ldots, x^m_r)$, some secret, random $r \in [n]$



Example: Private Sampling [IMSW07]

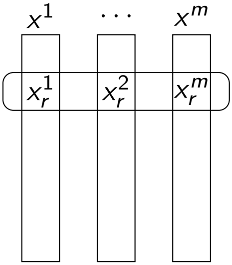
Input

- an m-ary function f
- m inputs of length n,

$$x_j^i = j\text{-th element of } x^i$$

Output

$f(x_r^1, \ldots, x_r^m)$, some secret, random $r \in [n]$



Example: Private Sampling [IMSW07]

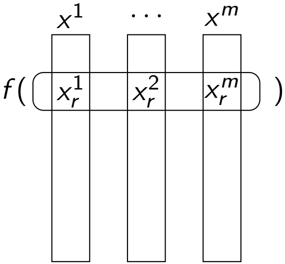
Input

- an m-ary function f
- m inputs of length n,

$$x^i_j = j\text{-th element of } x^i$$

Output

$f(x^1_r, \ldots, x^m_r)$, some secret, random $r \in [n]$



Example: Private Sampling [IMSW07]

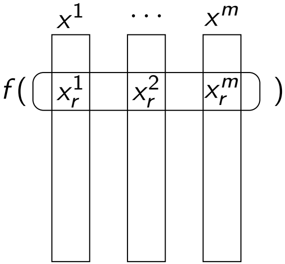
Input

- an m-ary function f
- m inputs of length n,

$$x_j^i = j\text{-th element of } x^i$$

Output

$f(x_r^1, \ldots, x_r^m)$, some secret, random $r \in [n]$



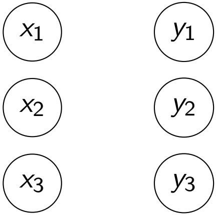
- used in:
  - private approximation (*e.g.* of the sum, of the norm)
  - private data-mining

Example: Private Bipartite Stable Matching [Gol06]

Input

- two sets (men and women) of size n
- a set of rankings

$$x_j^i = k \text{ if } x_i \text{ gives } y_j \text{ rank } k$$



Example: Private Bipartite Stable Matching [Gol06]

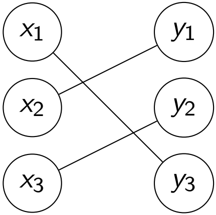
Input

- two sets (men and women) of size n
- a set of rankings

$$x_j^i = k \text{ if } x_i \text{ gives } y_j \text{ rank } k$$

Output

a *stable* bipartite matching

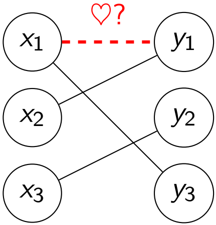


Example: Private Bipartite Stable Matching [Gol06]

Input

- two sets (men and women) of size n
- a set of rankings

$$x^i_j = k \text{ if } x_i \text{ gives } y_j \text{ rank } k$$



Output

a *stable* bipartite matching

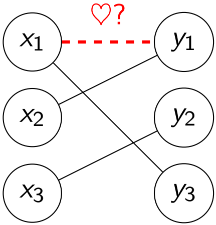
- *stability*: no unmatched individuals rank one another higher than their "spouse"

Example: Private Bipartite Stable Matching [Gol06]

Input

- two sets (men and women) of size n
- a set of rankings

$$x_j^i = k \text{ if } x_i \text{ gives } y_j \text{ rank } k$$



Output

a *stable* bipartite matching

- *stability*: no unmatched individuals rank one another higher than their "spouse"
- used in:
  - matching residents to hospitals (US, Canada, Scotland)
  - placement of students at universities (Norway, Scotland)
  - professional services (*e.g.* National Matching Services, Inc)

Outline

Introduction: Private mLUT

Input

- Database $\Delta = (\delta_0, \ldots, \delta_{n-1})$
- Party i holds $[\Delta]_i$, a share of Δ
- Party i holds $[\sigma]_i$, a share of secret index σ

Output

- Party i learns $[\delta_\sigma]_i$, a share of $\Delta[\sigma] = \delta_\sigma$

$$\mathrm{mLUT}([\Delta]_1, [\sigma]_1; [\Delta]_2, [\sigma]_2; \ldots; [\Delta]_m, [\sigma]_m) \rightarrow ([\delta_\sigma]_1; [\delta_\sigma]_2; \ldots; [\delta_\sigma]_m)$$

Definition (Private mLUT)

mLUT is t-private if no coalition of up to t parties can learn any information about σ or any of the elements in Δ.

Our Protocol: mLUT

Input

- Database $\Delta = (\delta_0, \ldots, \delta_{n-1})$
- Party i holds $[\Delta]_i$, a share of Δ, where $\bigoplus[\Delta]_i = \Delta$
- Party i holds $[\sigma]_i$, a share of secret index σ

Output

- Party i learns $[\delta_\sigma]_i$, a share of $\Delta[\sigma] = \delta_\sigma$

Protocol

- Let $[\Delta]_i = \Delta^i$
- For $i = 1$ to m:
  - Parties run $\text{g-mOT}(\Delta^i, [\sigma]_i; [\sigma]_{i+1}; \ldots; [\sigma]_{i+m}) \rightarrow ([\delta^i_\sigma]_i; [\delta^i_\sigma]_{i+1}; \ldots; [\delta^i_\sigma]_{i+m})$
- Party i (locally) computes $[\delta_\sigma]_i = \bigoplus[\delta^j_\sigma]_i$.

Outline

Our Subprotocol: g-mOT

Input

- One party holds $\Delta = (\delta_0, \ldots, \delta_{n-1})$
- Party i holds $[\sigma]_i$, a share of σ

Output

Party i holds $[\delta_\sigma]_i$, a share of $\Delta[\sigma]$

$$\text{g-mOT}(\Delta, [\sigma]_1; [\sigma]_2; \ldots; [\sigma]_m) \to ([\delta_\sigma]_1; [\delta_\sigma]_2; \ldots; [\delta_\sigma]_m)$$

Our Subprotocol: g-mOT construction (idea)

- Privately convert shares into inputs for efficient PIR

Our Subprotocol: g-mOT construction (idea)

- Privately convert shares into inputs for efficient PIR
- Use techniques to convert PIR into OT

Our Subprotocol: g-mOT construction (idea)

- Privately convert shares into inputs for efficient PIR
- Use techniques to convert PIR into OT
- PIR based on LFAH still efficient when using (traditionally inefficient) special representations of Δ

Our Subprotocol: g-mOT construction (idea)

- Privately convert shares into inputs for efficient PIR
- Use techniques to convert PIR into OT
- PIR based on LFAH still efficient when using (traditionally inefficient) special representations of Δ
  - Ex: database as $\log n$-dimensional $2 \times \ldots \times 2$ cube

Our Subprotocol: g-mOT construction (idea)

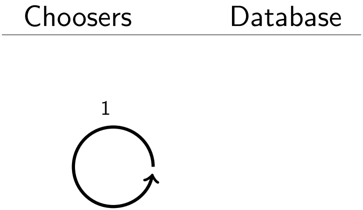
- Privately convert shares into inputs for efficient PIR
- Use techniques to convert PIR into OT
- PIR based on LFAH still efficient when using (traditionally inefficient) special representations of Δ
  - Ex: database as $\log n$-dimensional $2 \times \ldots \times 2$ cube
  - index used by PIR based on binary rep. of index

Our Subprotocol: g-mOT construction (idea)

- Privately convert shares into inputs for efficient PIR
- Use techniques to convert PIR into OT
- PIR based on LFAH still efficient when using (traditionally inefficient) special representations of Δ
  - Ex: database as $\log n$-dimensional $2 \times \ldots \times 2$ cube
  - index used by PIR based on binary rep. of index
- (Constant round) protocols exist to convert shares to this form

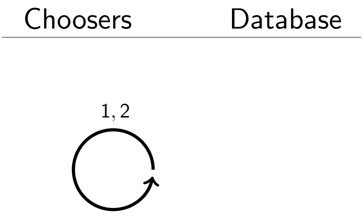
Our Subprotocol: g-mOT construction (sketch)

1. Choosers create a threshold LFAH encryption system



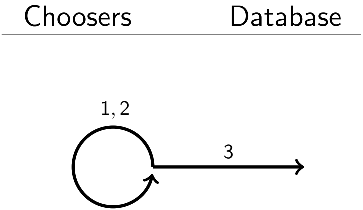
Our Subprotocol: g-mOT construction (sketch)

1. Choosers create a threshold LFAH encryption system
2. Choosers compute first PIR message using their shares of σ



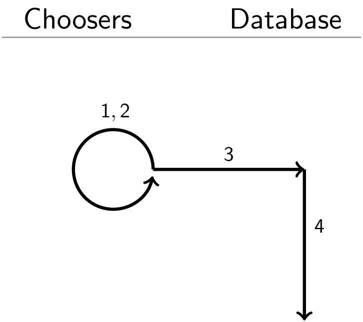
Our Subprotocol: g-mOT construction (sketch)

1. Choosers create a threshold LFAH encryption system
2. Choosers compute first PIR message using their shares of σ
3. Choosers send the above, with $E(\sigma)$, to Database



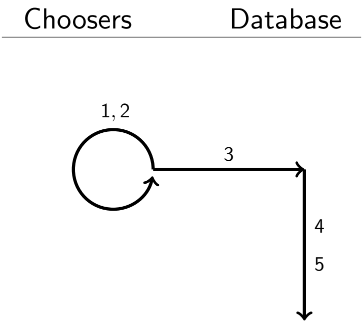
Our Subprotocol: g-mOT construction (sketch)

1. Choosers create a threshold LFAH encryption system
2. Choosers compute first PIR message using their shares of σ
3. Choosers send the above, with $E(\sigma)$, to Database
4. Database uses $E(\sigma)$ to blind Δ



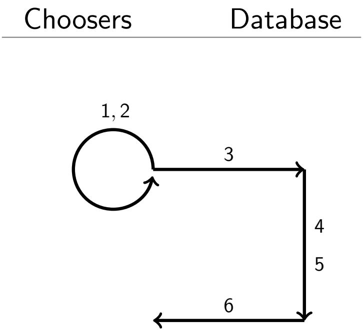
Our Subprotocol: g-mOT construction (sketch)

1. Choosers create a threshold LFAH encryption system
2. Choosers compute first PIR message using their shares of σ
3. Choosers send the above, with $E(\sigma)$, to Database
4. Database uses $E(\sigma)$ to blind Δ
5. Database runs the PIR protocol



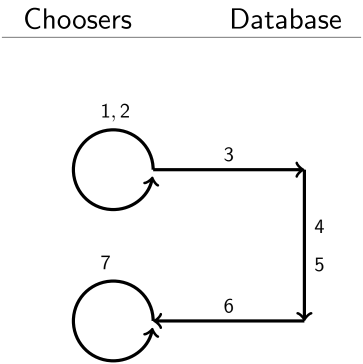
Our Subprotocol: g-mOT construction (sketch)

1. Choosers create a threshold LFAH encryption system
2. Choosers compute first PIR message using their shares of σ
3. Choosers send the above, with $E(\sigma)$, to Database
4. Database uses $E(\sigma)$ to blind Δ
5. Database runs the PIR protocol
6. Database sends response to Choosers



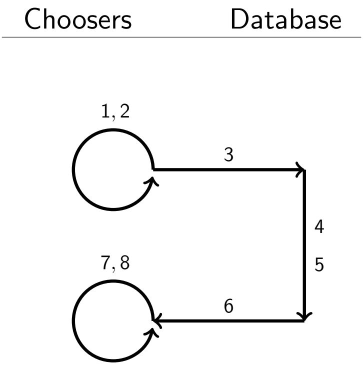
Our Subprotocol: g-mOT construction (sketch)

1. Choosers create a threshold LFAH encryption system
2. Choosers compute first PIR message using their shares of σ
3. Choosers send the above, with $E(\sigma)$, to Database
4. Database uses $E(\sigma)$ to blind Δ
5. Database runs the PIR protocol
6. Database sends response to Choosers
7. Choosers collaborate to decrypt response ($\alpha - 1$ times)



Our Subprotocol: g-mOT construction (sketch)

1. Choosers create a threshold LFAH encryption system
2. Choosers compute first PIR message using their shares of σ
3. Choosers send the above, with $E(\sigma)$, to Database
4. Database uses $E(\sigma)$ to blind Δ
5. Database runs the PIR protocol
6. Database sends response to Choosers
7. Choosers collaborate to decrypt response ($\alpha - 1$ times)
8. Choosers split remaining ciphertext into shares



Our subprotocol: g-mOT Cost Analysis

- When
  - Database elements of length ℓ
  - Security parameter k
- Total:

Our subprotocol: g-mOT Cost Analysis

- When
  - Database elements of length ℓ
  - Security parameter k
- Comm. Complexity:
  - conversion protocol $=$ $b \log b$ multiplications [DFK$^+$06]
  - here, $b = |\sigma| = \log n$
  - multiparty multiplication protocol $= O(m)$
  - thus, total for conversion protocol $= O(m \log n \log \log n)$
  - PIR $= O(k \log^2 n + \ell \log n)$ [Lip03]
- Total:
  - Comm: $O(mk \log^2 n + m\ell \log n)$

Our subprotocol: g-mOT Cost Analysis

- When
  - Database elements of length ℓ
  - Security parameter k
- Comm. Complexity:
  - conversion protocol $=$ $b \log b$ multiplications [DFK+06]
  - here, $b = |\sigma| = \log n$
  - multiparty multiplication protocol $= O(m)$
  - thus, total for conversion protocol $= O(m \log n \log \log n)$
  - PIR $= O(k \log^2 n + \ell \log n)$ [Lip03]
- Round:
  - Output share conversion protocols require $O(\log n)$ rounds
  - All other parts of the protocol are constant-round
- Total:
  - Comm: $O(mk \log^2 n + m\ell \log n)$
  - Round: $O(\log n)$

Our subprotocol: Security Claim

Claim:

Our protocol is t-private when the Damgård-Jurik LFAH system is IND-CPA secure.

- in standard model, under Paillier and composite DDH assumptions [DJ03]

Our subprotocol: Security Claim

Claim:

Our protocol is t-private when the Damgård-Jurik LFAH system is IND-CPA secure.

- in standard model, under Paillier and composite DDH assumptions [DJ03]
- can reduce assumptions: use generic AH system (not LFAH)
- w/ generic AH, round complexity increases by polylog factor
  - less convenient database representation
  - thus more complex share-conversion operations

Outline

Improvements to Private Sampling Applications

Summary

Protocol	Work	Comm.	Round
[IMSW07]	$O(m^2 n)$	$O(m^2 \log n(k \log n + \ell + mk))$	$O(m \log n)$
Ours	$O(m)$	$O(m^2 \log n(k \log n + \ell))$	$O(\log n)$

- for comparison purposes, above ignores costs associated with f
- above, work $=$ number of invocations of the PIR routine by database
- under general AH assumptions, ours remains efficient

Summary

Protocol	Work	Comm.	Round
CT-RSA [FGM07]	$O(n^4\sqrt{\log n})$	$O(mn^3)$	$\widetilde{O}(n^2)$
Ours	$O(n^4)$	$O(mn^2)$	$\widetilde{O}(n^2)$

- in the setting of Golle's private matching algorithm

Thank you.

Thank You.

Ivan Damgård, Matthias Fitzi, Eike Kiltz, Jesper Buus Nielsen, and Tomas Toft.
Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation.
In *Proceedings of the Theory of Cryptography Conference*, pages 285–304, 2006.

Ivan Damgård and Mads Jurik.
A length-flexible threshold cryptosystem with applications.
In *Information Security and Privacy*, pages 350–364, 2003.

Matthew Franklin, Mark Gondree, and Payman Mohassel.
Improved efficiency for private stable matching.
In *The Cryptographer's Track at the RSA Conference (CT-RSA)*, 2007.

Philippe Golle.
A private stable matching algorithm.
In *Financial Crypto (FC '06)*, 2006.

Yuval Ishai, Tal Malkin, Martin J. Strauss, and Rebecca N. Wright.
Private multiparty sampling and approximation of vector combinations.
In *Proceedings of the 34th International Colloquium on Automata, Languages and Programming (ICALP)*, 2007.

Helger Lipmaa.
Verifiable homomorphic oblivious transfer and private equality test.
In *Advances in Cryptology – ASIACRYPT '03*, pages 416–433, 2003.

Moni Naor and Kobbi Nissim.
Communication preserving protocols for secure function evaluation.
In *STOC '01: Proceedings of the 33rd annual ACM Symposium on Theory of Computing*, pages 590–599, 2001.

Backup: Further Work

- Develop constant-round mOT protocols
- Find natural problems where shared inputs are already in binary representations, for which this work is very efficient (re: round complexity)
- Develop more efficient techniques for oblivious writes
- Find efficient black-box reduction of mOT to 2-party OT
- Consider active adversaries

Backup: LFAH

Message Expansion for α-times encryption

Additive Homo.	$(s+j)k \rightarrow \eta^{\alpha}(s+j)k$
Length-Flexible Additive Homo.	$(s+j\xi)k \rightarrow (s+(j+\alpha)\xi)k$

For [DJ03], $\xi = 1$

Comm. for LFAH-PIR [Lip03]

$$\frac{k\xi}{2}\log^2 n + \frac{3k\xi}{2}\log n + \ell\log n + \ell = \Theta(k\log^2 n + \ell\log n)$$

- database-side comm costs $= (\xi\log n + s)k$
- chooser-side comm costs $= (\frac{\xi}{2}\log^2 n + (s + \frac{\xi}{2})\log n)k$
- where $s = \frac{\ell}{k}$