Algorithmic Aspects of Secure Computation and Communication

Matt Franklin
U. C. Davis
Algorithmic Aspects of ...

- Secure computation
  - Linear Algebra [KMWF 07]
    - Algorithmic idea: Linearly Recurrent Sequences
  - Longest Common Subsequence [FGM 09]
    - Algorithmic idea: Four-Russian speedup

- Secure communication
  - Reliable Message Transmission [BF 99, BM 05]
    - Algorithmic idea: graph connectivity variants
Secure Computation: Private 2-Party Setting

Compute function without leaking inputs. semi-honest adversary (passive faults)
Secure Computation: Private 2-Party Variant

Enc(x), Enc(y), ...

Enc(F(x,y, ..))

public key pk, secret key sk

Essentially equivalent to the classic setting.

ISAAC 2010
Private 2-Party Linear Algebra

encrypted inputs
(matrices, vectors, etc)

encrypted outputs
(determinant, rank, linear system solve, etc)

public key pk, secret key sk

ISAAC 2010
Private 2-Party Linear Algebra

<table>
<thead>
<tr>
<th></th>
<th>Communication complexity</th>
<th>Round complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yao86</td>
<td>(\tilde{O}(n^{2.38}))</td>
<td>(O(1))</td>
</tr>
<tr>
<td>KMWF07</td>
<td>(\tilde{O}(n^2))</td>
<td>(O(\log n))</td>
</tr>
</tbody>
</table>

Communication = # of encrypted values exchanged  
inputs = n by n matrices, etc.
# Private 2-Party Linear Algebra

<table>
<thead>
<tr>
<th></th>
<th>Communication complexity</th>
<th>Round complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yao86</td>
<td>$\tilde{O}(n^{2.38})$</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>KMWF07</td>
<td>$\tilde{O}(n^2)$</td>
<td>$O(\log n)$</td>
</tr>
<tr>
<td>Gentry 09</td>
<td>$\tilde{O}(n^2)$</td>
<td>$O(1)$</td>
</tr>
</tbody>
</table>

fully homomorphic encryption

ISAAC 2010
Linearly Recurrent Sequences for Solving Linear Systems

• Faster than Gaussian Elimination for:
  – sparse linear systems (Wiedemann 86)
    • Ay = x where A has few nonzero entries.
  – special form linear systems (Kaltofen-Sanders 91)
    • Ay = x where Av can be computed “fast” (for all v)
    • Sparse, Vandermonde, Sylvester, Toeplitz, etc.

• We apply to ordinary matrices
Linearly Recurrent Sequences of Field Elements

• The sequence of field elements \((a_i)_i\) is linearly recurrent if there exists field elements \(f_0, \ldots, f_n\) such that \(f_0 a_i + \ldots + f_n a_{i+n} = 0\) for all \(i\).
  - \(f(x) = f_n x^n + \ldots + f_0\) is a characteristic poly of \((a_i)_i\)

• minimal poly = char poly of least degree
  - \(O(n \text{ polylog } n)\) algorithm to compute min poly
  - Padé Approximation, Fast Extended GCD.

ISAAC 2010
Linearly Recurrent Sequences of Matrices

- The sequence of matrices \((m_i)_i\) is linearly recurrent if there exists field elements \(f_0, ..., f_n\) such that \(f_0 m_i + ... + f_n m_{i+n} = 0\) for all \(i\).
  - \(f(x) = f_n x^n + ... + f_0\) is a characteristic poly of \((m_i)_i\)
- Min poly of \((m_i)_i\) = char poly of least degree
- Min poly of matrix \(M = \min \text{poly of } (M^i)_i\)
  - Useful for computing rank, determinant, etc.
Additively Homomorphic Encryption

\[ \text{add-hom public key } pk, \]  
\[ \text{add-hom secret key } sk \]

\[ \text{Enc}(x), \text{Enc}(y) \]

\[ \text{Enc}(x+y) \]

Relatively mild assumption (e.g., Paillier 1999)
Yao’s Garbled Circuit Protocol [1986]

• Private 2-party computation for any function

• Communication complexity $O(G + \alpha + \beta)$:
  – $G =$ size of Boolean circuit to compute function
  – $\alpha =$ number of inputs, $\beta =$ number of outputs

• Our solutions use Yao as a sub-protocol
  – On functions with small boolean circuits

ISAAC 2010
Efficient Private Matrix Multiply

encrypted matrix $\text{Enc}(A)$, encrypted matrix $\text{Enc}(B)$, add-hom public key $pk$

Choose random matrices $R, S$

encrypted matrix $\text{Enc}(AB)$

add-hom public key $pk$, add-hom secret key $sk$

$\text{Enc}(A + R)$, $\text{Enc}(B + S)$

$\text{Enc}((A + R)(B + S))$
Efficient Private Matrix Exponentiation

encrypted matrix Enc(A), add-hom public key pk

encrypted matrices Enc(A^2), Enc(A^4), Enc(A^8), ..., Enc(A^n)

add-hom public key pk, add-hom secret key sk

private matrix mult protocol \( \sim \log n \) times

ISAAC 2010
Efficient Private Sequence Mult

encrypted matrix $Enc(A)$, unencrypted vector $u, v$
add-hom public key $pk$

add-hom public key $pk$, add-hom secret key $sk$

encrypted field elements $Enc(u^TA^iv), 0 \leq i \leq 2n-1$

private matrix exponentiation
$\sim \log n$ private matrix mults (carefully chosen)
Private MINPOLY Protocol

Input: $\text{enc}(A) = \text{encrypted } n \times n \text{ matrix}$
Output: encrypted min poly of $A$

1. Pick random vectors $u, v$.
2. Compute $\text{enc}(u^T A^i v)$ for $i = 0, \ldots, 2n-1$ with Efficient Private Sequence Mult Protocol
3. Compute encrypted min poly of encrypted sequence of field elements from step (2), with Yao’s protocol on “small” boolean circuit.
Private MINPOLY Protocol Analysis

- \( \min \text{poly } m_A \text{ of matrix } A = \min \text{poly of } (u^T A^i v)_i \)
  with \( \text{prob} \geq 1 - (2 \deg(m_A) / |F|) \geq 1 - 2n/|F| \)
  when \( A \) is an \( n \) by \( n \) matrix.

- Private MINPOLY protocol computes \( \text{Enc}(m_A) \)
  from \( \text{Enc}(A) \) with \( \text{prob} \geq 1 - 2n/|F| \), using
  - \( O(n^2 \log n \log |F|) \) communication
  - \( O(\log n) \) rounds.

ISAAC 2010
Private 2-Party Linear Algebra: Other Functions

- Determinant, rank, linear system solution, etc
  - Same efficiency as Private MINPOLY
  - $O(n^2 \log n \log |F|)$ comm, $O(\log n)$ rounds
  - Similar ideas and techniques
- See paper for details [KMWF07].
- Open: $O(n^2 \text{polylog}(n, |F|))$ comm, $O(1)$ rounds
  - without using “big hammer”
Private 2-Party
Longest Common Subsequence

Both parties learn LCS(A,B), while hiding inputs otherwise.
Today’s talk: Both parties learn size of LCS.
## Private 2-Party LCS

<table>
<thead>
<tr>
<th></th>
<th>Comm complexity</th>
<th>Alice’s Work</th>
<th>Bob’s Work</th>
<th>Round complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Yao86</strong></td>
<td>$O(mn)$</td>
<td>poly($m,n$)</td>
<td>poly($m,n$)</td>
<td>$O(1)$</td>
</tr>
<tr>
<td><strong>FGM09</strong></td>
<td>$O(mn/t)$</td>
<td>$O(2^t)$</td>
<td>$O(mn/t)$</td>
<td>$O(m/t + n/t)$</td>
</tr>
</tbody>
</table>

For strings of length $m$ and $n$ over constant-size alphabet. For any $t$, $1 \leq t \leq \min(m,n)$.  

ISAAC 2010
## Private 2-Party LCS

<table>
<thead>
<tr>
<th></th>
<th>Comm complexity</th>
<th>Alice’s Work</th>
<th>Bob’s Work</th>
<th>Round complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yao86</td>
<td>$O(mn)$</td>
<td>$\text{poly}(m,n)$</td>
<td>$\text{poly}(m,n)$</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>FGM09</td>
<td>$O(mn/t)$</td>
<td>$O(2^t)$</td>
<td>$O(mn/t)$</td>
<td>$O(m/t + n/t)$</td>
</tr>
<tr>
<td>Gentry 09</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
<td>$\text{poly}(m,n)$</td>
<td>$O(1)$</td>
</tr>
</tbody>
</table>

fully homomorphic encryption

ISAAC 2010
### LCS Dynamic Programming
(Needleman-Wunsch, Smith-Waterman)

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>f</th>
<th>b</th>
<th>c</th>
<th>a</th>
<th>a</th>
<th>a</th>
<th>a</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>f</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

If $A[i] = B[j]$ then

$$L[i, j] = L[i-1, j-1] + 1$$

If $A[i] \neq B[j]$ then

$$L[i, j] = \max(L[i-1, j], L[i, j-1])$$

**LCS-length(abfc, afbcaa)**

**LCS-length(A, B)**

ISAAC 2010
Four-Russians Speedup
(Masek-Paterson)

Overlapping t by t blocks

Each block determined by:
- top row of the block,
- left column of the block,
- length-t substring of A,
- length-t substring of B.
Four-Russians Speedup

Offset vector = increments in row or column of block

Shift value = maximum increase within block (redundant but useful)

F(top row offset, left column offset, substrings of A and B) = bottom row offset, right column offset, shift value

ISAAC 2010
Four-Russians Speedup

• Pre-compute all possible $t$ by $t$ blocks
  – $2^{2^t}$ possible offset vectors for top row, left column
  – $|\Sigma|^{2^t}$ possible length-$t$ substrings for $A$ and $B$
  – (bottom row offset vector, right column offset vector, shift value) stored

• Fill in the block boundaries of the LCS table:
  – Look up the appropriate pre-computed block
  – Add offset vectors and shift value to LCS table.
  – Repeat $(m/t) \times (n/t)$ times
### Four-Russians Table Look-Up

<table>
<thead>
<tr>
<th>index</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(i_1, i_2, i_3, i_4)$</td>
<td>(offsets, shift value)</td>
</tr>
<tr>
<td>$(j_1, j_2, j_3, j_4)$</td>
<td>(offsets, shift value)</td>
</tr>
<tr>
<td>$(k_1, k_2, k_3, k_4)$</td>
<td>(offsets, shift value)</td>
</tr>
</tbody>
</table>

$O(2^t)$ entries of size $O(t)$ for alphabet of size $O(1)$
Private Block Retrieval

Database DB = vector of bitstrings

Index hidden from database owner.
Nothing hidden from index holder.

 ISAAC 2010
Private Indirect Indexing

2-out-of-2 share of index $i$

$\text{Index hidden from database owner.}$
$\text{Database hidden from index holder.}$

$\text{DB = vector of bitstrings,}$
$\text{and 2-out-of-2 share of index } i$

$\text{2-out-of-2 share of DB}[i]$
$\text{= ith bitstring}$
Efficient Private Indirect Indexing

• Private Block Retrieval (Gentry-Ramzan):
  – $O(k+d)$ comm for $d$-bit blocks (sec param $k$)
  – Hardness related to $\phi$-Hiding Assumption
    • RSA modulus $n$ “hides” small factors of $\phi(n) = (p-1)(q-1)$

• Private Indirect Indexing from any PBR:
  – General transform (Naor-Pinkas, Naor-Nissim)
  – No asymptotic increase in communication
  – Requires pseudorandom functions
Our Two-Party LCS Protocol

• Alice pre-computes Four-Russians table.
  – Indexed by top row (offset), left column (offset),
    length-t substrings of A and B.
  – Entries are bottom row offset vector, right column
    offset vector, shift value.
• Alice and Bob iterate Private Indirect Indexing:
  – compute shares of offset vectors, shift values
• L(m,n) recovered from shift value shares
Total Cost

\[ \frac{mn}{t^2} \] Private Indirect Indexings,
\[ \frac{m}{t} + \frac{n}{t} \] rounds (parallelizing)
See paper for details [FGM09]

<table>
<thead>
<tr>
<th>Comm complexity</th>
<th>Alice’s Work</th>
<th>Bob’s Work</th>
<th>Round complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>( O(\frac{mn}{t}) )</td>
<td>( O(2^t) )</td>
<td>( O(\frac{mn}{t}) )</td>
<td>( O(\frac{m}{t} + \frac{n}{t}) )</td>
</tr>
</tbody>
</table>

Open: Reduce comm and rounds with poly work (without using “big hammers”).
Secure Communication:
Reliable Message Transmission

protocol over network of reliable channels

maliciously faulty network node
Secure Communication: Reliable Message Transmission

Message sent on 3 paths

Message corrupted on 1 path

Message accepted if received on 2 paths

ISAAC 2010
Reliable Message Transmission

• $t$ malicious faults $\Leftrightarrow (2t+1)$-connected [DDWY]
  – Simple efficient protocol (“majority vote”)
  – Perfect reliability (failure probability 0)

• Can MAC’s help?
  – MAC = symmetric key analog of digital signature
  – MAC($message$, key) = hard-to-forge “tag”
Reliable Message Transmission when Sender and Receiver Have MAC Key

MAC key \( u \) (message, tag) sent on 2 paths

MAC key \( v \) (message, tag) corrupted on 1 path

Any message accepted with valid tag

ISAAC 2010
Reliable Message Transmission with Arbitrary Distribution of MAC Keys

Can A send reliably to B if u, v, w or x is malicious fault?
(t,\varepsilon)-Reliable Message Transmission with Arbitrary Distribution of MAC Keys

- Send message across network (with \varepsilon error)
  - Synchronous network of reliable channels
  - Pre-distributed MAC keys (pair-wise)
  - Adversary controls t malicious faulty parties

- Characterization [Beimel-Franklin 1999]
  - Complex recursive condition on comm/auth networks
  - Highly inefficient: \((n/t)^{O(t)}\) rounds [Beimel-Malka 2005]
  - Open: Find an efficient protocol (or counterexample)
Conclusion

- Rich algorithmic issues in crypto protocols
- Open Problems:
  - find algorithms that map to efficient protocols
  - Exploit relatively efficient “generic” methods
    - Yao: Boolean circuits
    - Naor-Nissim01: Bool circuits with look-up tables
    - BGN05, Gentry09, etc: somewhat-hom encryption
  - Consider protocols with small privacy leakage
- Thank you!