Secure Computation in the Real(ish) World

Carnegie Mellon
20 April 2011

David Evans
University of Virginia
http://www.cs.virginia.edu/evans
http://www.MightBeEvil.com

“Genetic Dating”

Bob

Alice

Genome Compatibility Protocol

WARNING! Don’t Reproduce

WARNING! Don’t Reproduce

Genome Sequencing

1990: Human Genome Project starts, estimate $3B to sequence one genome ($0.50/base)

2000: Human Genome Project declared complete, cost ~$300M

Cost to sequence human genome

Moore’s Law prediction (halve every 18 months)

Data from National Human Genome Research Institute: http://www.genome.gov/sequencingcosts
Human Genome Sequencing Using Unchained Base Reads on Self-Assembling DNA Nanoarrays


George Church (Personal Genome Project)

On Jim Watson’s APOE status: genetic information is hard to hide


Secure Two-Party Computation

Can Alice and Bob compute a function of their private data, without exposing anything about their data besides the result?

Secure Function Evaluation

Alice (circuit generator) Bob (circuit evaluator)

Agree on $f(a, b) \rightarrow x$

Picks $a \in \{0, 1\}^5$

Picks $b \in \{0, 1\}^3$

Outputs $x = f(a, b)$ without revealing $a$ to Bob or $b$ to Alice.

Yao’s Garbled Circuits

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>$b$</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Computing with Meaningless Values?

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>(b)</td>
</tr>
<tr>
<td>(a_0)</td>
<td>(b_0)</td>
</tr>
<tr>
<td>(a_0)</td>
<td>(b_1)</td>
</tr>
<tr>
<td>(a_1)</td>
<td>(b_0)</td>
</tr>
<tr>
<td>(a_1)</td>
<td>(b_1)</td>
</tr>
</tbody>
</table>

\(a\), \(b\), \(x\) are random values, chosen by the circuit generator but meaningless to the circuit evaluator.

Computing with Garbled Tables

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>(b)</td>
</tr>
<tr>
<td>(a_0)</td>
<td>(b_0)</td>
</tr>
<tr>
<td>(a_0)</td>
<td>(b_1)</td>
</tr>
<tr>
<td>(a_1)</td>
<td>(b_0)</td>
</tr>
<tr>
<td>(a_1)</td>
<td>(b_1)</td>
</tr>
</tbody>
</table>

\(a\), \(b\), \(x\) are random values, chosen by the circuit generator but meaningless to the circuit evaluator.

Garbled Circuit Protocol

**Alice (circuit generator)**

Creates random keys: \(a_0, a_1, b_0, b_1, x_0, x_1\)

**Bob (circuit evaluator)**

Sends \(a_i\) to Bob based on her input value


does the Bob learn his own input wires?

**Primitive: Oblivious Transfer**

**Alice**

Knows \(b_0, b_1\)

**Bob**

Picks \(i \in \{0,1\}\)

Learns nothing

Learns \(b_i\) (only)

Oblivious: Alice doesn’t learn which secret Bob obtains

Transfer: Bob learns one of Alice’s secrets

Rabin, 1981; Even, Goldreich, and Lempel, 1985; many subsequent papers

Garbled Circuits security proofs depend on this very weak model

**Threat Model**

**Semi-Honest (Honest But Curious) Adversary**

Adversary follows the protocol as specified (✓)

Curious adversary tries to learn more from protocol execution transcript

Garbled Circuits security proofs depend on this very weak model
**Building Computing Systems**

<table>
<thead>
<tr>
<th>Digital Electronic Circuits</th>
<th>Garbled Circuits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operate on known data</td>
<td>Operate on encrypted wire labels</td>
</tr>
<tr>
<td>One-bit logical operation requires moving a few electrons a few nanometers (hundreds of Billions per second)</td>
<td>One-bit logical operation requires performing (up to) 4 encryption operations (~100,000 gates per second)</td>
</tr>
<tr>
<td>Reuse is great!</td>
<td>Reuse is not allowed!</td>
</tr>
<tr>
<td>All basic operations have similar cost</td>
<td>Some logical operations “free” (XOR, NOT)</td>
</tr>
</tbody>
</table>

**Faster Garbled Circuits**

- Circuit-Level Application
- Circuit Structure
- GC Framework

Protocol 2 (generic SMC) is very fast. Protocol 1 is ideal for small strings because the entire computation is performed in one round, but the circuit size is extremely large for longer strings. Our prototype circuit compiler can compile circuits for problems of size (200, 200) but uses almost 2 GB of memory to do so. Significantly larger circuits would be constrained by available memory for constructing their garbled versions.

**Heterozygous Recessive Risk**

Alice: {5283423, 1425236, 839523, ...}
Bob: {5823527, 839523, 169325, ...}

**Goal:** Find the intersection of A and B
**Bit Vector Intersection**

Alice's Recessive genes:
\[ \{ 5283423, 1425236, 839523, \ldots \} \]

Bob's Recessive genes:
\[ \{ 5823527, 839523, 169325, \ldots \} \]

\[ [0, 0, 1, 0, 0, 1, 0, 1, 1, 0] \]

\[ [0, 0, 1, 0, 0, 0, 0, 1, 0, 0] \]

Bitwise AND

Apple

**Scaling**

What if there are millions of possible diseases?

- Length of bit vector:
  - number of possible values
  - \((2^L\text{ where } L \text{ is number of bits for each value})\)

**Pairwise Comparison**

- randomly permute A
- randomly permute B
- for i in range(0, n-1):
  - for j in range(0, n-1):
    - if \(A[i] = B[j]\) output \(A[i]\)

\(n^2\) comparisons

**Short-Circuit Pairwise Comparison**

for i in range(0, n-1):
  - mask[i] = false
for i in range(0, n-1):
  - for j in range(0, n-1):
    - if not mask[i] and \(A[i] = B[j]\):
      - reveal \(A[i]\) to both
      - mask[i] = true
      - break

**Short-Circuit Analysis**

\(N_{Equal} \approx \frac{(2n - \hat{n})^2 - \hat{n}}{4}\)

\(\%\) of elements joint:
- save 43% of effort

**Scaling**

This is still \(O(n^2)\). Is there an \(O(n \log n)\) solution?
Sort-Compare-Shuffle

Sort: Take advantage of total order of elements

Compare: adjacent elements

Shuffle to hide positions

Bitonic Sorting

Sort 2n bitonic inputs with n log(2n) CompareSwap circuits.

Can’t reveal results yet! Position leaks information.
Oblivious Shuffling

Homomorphic Encryption Shuffling Protocol
Add random mask, permute, exchange and reveal
Expensive
Sort
Simple… but expensive
Random Permutation

)!-

I do not imagine that many of the Turing lecturers who will follow me will be people who were acquainted with Alan Turing. ... Although a mathematician, Turing took quite an interest in the engineering side of computer design... Turing's contribution to this discussion was to advocate the use of gin, which he said contained alcohol and water in just the right proportions...

Sir Maurice Wilkes (1913-29 Nov 2010), Computers Then and Now (1967 Turing Award Lecture)

Waksman Network

Same circuit can generate any permutation: select a random permutation, and pick swaps

\[ n \log n - n + 1 \] gates

Private Set Intersection Protocol
Gates to generate and evaluate

Free

\[ n \log(2n) \times 2\sigma \]

\[ (3\sigma - 1)(n - 1) + (2\sigma - 1) \]

\[ n \log n - n + 1 \]
Some Other Results

<table>
<thead>
<tr>
<th>Problem</th>
<th>Best Previous Result</th>
<th>Our Result</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hamming Distance (Face Recognition, Genetic Dating) – two 900-bit vectors</td>
<td>213s [SCiFI, 2010]</td>
<td>0.051s</td>
<td>4176</td>
</tr>
<tr>
<td>Levenshtein Distance (genome, text comparison) – two 200-character inputs</td>
<td>534s [Jha+, 2008]</td>
<td>18.4s</td>
<td>29</td>
</tr>
<tr>
<td>Smith-Waterman (genome alignment) – two 60-nucleotide sequences</td>
<td>[Not Implementable]</td>
<td>447s</td>
<td>-</td>
</tr>
<tr>
<td>AES Encryption</td>
<td>3.3s [Menezes, 2010]</td>
<td>0.2s</td>
<td>16.5</td>
</tr>
<tr>
<td>Fingerprint Matching (1024-entry database, 640x8bit vectors)</td>
<td>~83s [Barni, 2010]</td>
<td>18s</td>
<td>4.6</td>
</tr>
</tbody>
</table>

Scalable: 1 Billion gates evaluated at ~100,000 gates/second on laptop

Demo!
Private Set Intersection on Android Devices
http://MightBeEvil.com/mobile/
Peter Chapman and Yan Huang

Much of the early engineering development of digital computers was done in universities. A few years ago, the view was commonly expressed that universities had played their part in computer design, and that the matter could now safely be left to industry. I do not think that it is necessary that work on computer design should go on in all universities, but I am glad that some have remained active in the field. Apart from the obvious functions of universities in spreading knowledge, and keeping in the public domain material that might otherwise be hidden, universities can make a special contribution by reason of their freedom from commercial considerations, including freedom from the need to follow the fashion.

Sir Maurice Wilkes (June 1913-Nov 2010), 1967 Turing Award Lecture

Funding: NSF, MURI (AFOSR)
Android toys: Google

Shameless Plug
www.computingbook.org