Practical Cryptographic Secure Computation

DHOSA MURI
PIs Meeting
Berkeley, CA
28 April 2011

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

TRANSFORMATION
- SVA
- Binary translation and emulation
- Formal methods

→ e.g., Enforce properties on a malicious OS

HARDWARE
- Hardware support for isolation
- Dealing with malicious hardware

→ e.g., Prevent data exfiltration

SYSTEM ARCHITECTURES
- **Cryptographic secure computation**
- Data-centric security
- WEB-BASED ARCHITECTURES
  - Secure browser appliance
  - Secure servers

→ e.g., Enable complex distributed systems, with resilience to hostile OS's

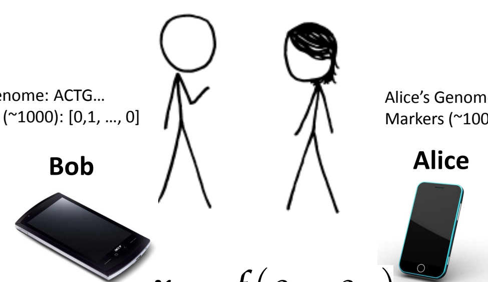
---

Secure Two-Party Computation

Bob's Genome: ACTG…
Markers (~1000): [0,1, …, 0]

Alice's Genome: ACTG…
Markers (~1000): [0, 0, …, 1]

Bob **Alice**



$$x = f(g_A, g_B)$$

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)**

Picks $a \in \{0,1\}^s$ Agree on $f(a,b) \rightarrow x$ Picks $b \in \{0,1\}^t$

Garbled Circuit Protocol

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

Andrew Yao, 1982/1986

Yao's Garbled Circuits

Inputs		Output
a	b	x
0	0	0
0	1	0
1	0	0
1	1	1

a, b → **AND** → x

Computing with Meaningless Values?

Inputs		Output
a	b	x
a_0	b_0	x_0
a_0	b_1	x_0
a_1	b_0	x_0
a_1	b_1	x_1

a_i, b_i, x_i are **random** values, chosen by the **circuit generator** but **meaningless** to the **circuit evaluator**.

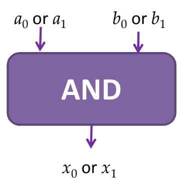
a_0 or a_1, b_0 or b_1 → **AND** → x_0 or x_1

Computing with Garbled Tables

Inputs		Output
a	b	x
a_0	b_0	$Enc_{a0,b0}(x_0)$
a_0	b_1	$Enc_{a0,b1}(x_0)$
a_1	b_0	$Enc_{a1,b0}(x_0)$
a_1	b_1	$Enc_{a1,b1}(x_1)$

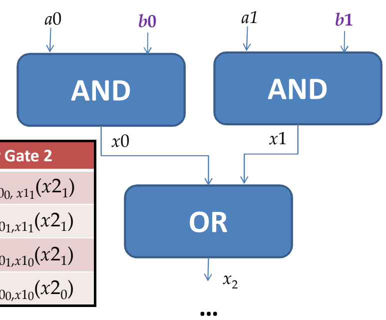
Bob can only decrypt **one** of these!

a_i, b_i, x_i are **random** values, chosen by the **circuit generator** but **meaningless** to the **circuit evaluator**.



Garbled And Gate
$Enc_{a0,\,b1}(x_0)$
$Enc_{a1,b1}(x_1)$
$Enc_{a1,b0}(x_0)$
$Enc_{a0,b0}(x_0)$

Chaining Garbled Circuits

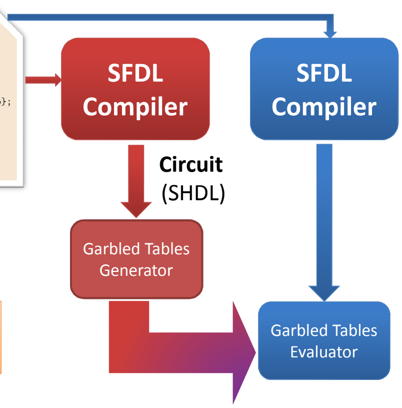


And Gate 1
$Enc_{a10,\,b11}(x1_0)$
$Enc_{a11,b11}(x1_1)$
$Enc_{a11,b10}(x1_0)$
$Enc_{a10,b10}(x1_0)$

Or Gate 2
$Enc_{x00,\,x11}(x2_1)$
$Enc_{x01,x11}(x2_1)$
$Enc_{x01,x10}(x2_1)$
$Enc_{x00,x10}(x2_0)$

We can do ***any*** computation privately this way!

Fairplay



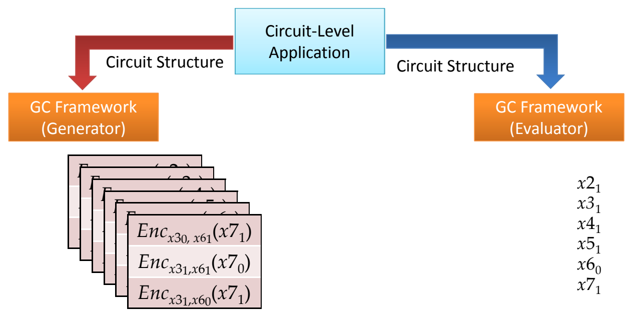

```
program Millionaires {
  type int = Int<4>;  // 4-bit integer
  type AliceInput = int;
  type BobInput = int;
  type AliceOutput = Boolean;
  type BobOutput = Boolean;
  type Output = struct {
      AliceOutput alice, BobOutput bob};
  type Input = struct {
      AliceInput alice, BobInput bob};

  function Output out(Input inp) {
    out.alice = inp.alice > inp.bob;
    out.bob = inp.bob > inp.alice;
```

SFDL Program

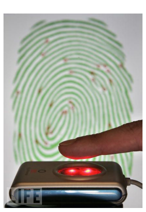
Dahlia Malkhi, Noam Nisan, Benny Pinkas and Yaron Sella [USENIX Sec 2004]

Our Approach: Faster Garbled Circuits



Gates can be evaluated as they are generated: **pipelining**
Gates can be evaluated in any topological sort order: **parallelizing**
Garbled evaluation can be **combined with normal execution**

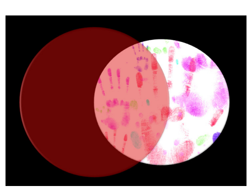
Applications



Privacy-Preserving Biometric Matching

Private Personal Genomics

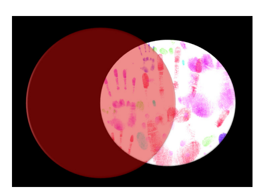
Private AES Encryption



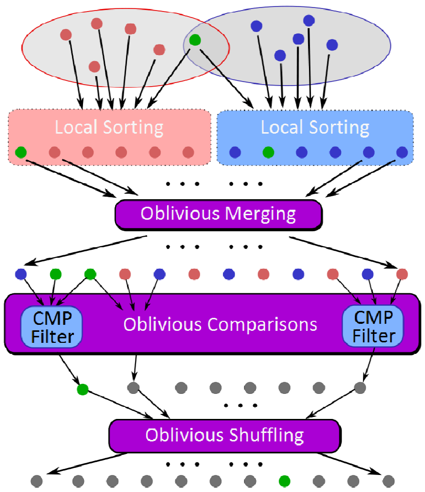
Private Set Intersection

Private Set Intersection

- Do Alice and Bob have any contacts in common?
- Two countries want to compare their miscreant lists
- Identify common medical records across hospitals
- Two companies want to do joint marketing to common customers



Sort-Compare-Shuffle

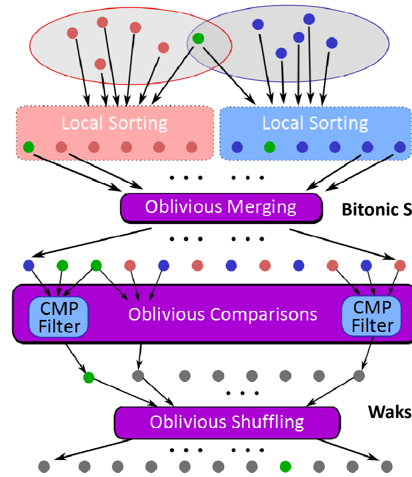


Sort: Take advantage of total order of elements

Compare adjacent elements

Shuffle to hide positions

Private Set Intersection Protocol



Gates to generate and evaluate

Free

Bitonic Sorting Circuit

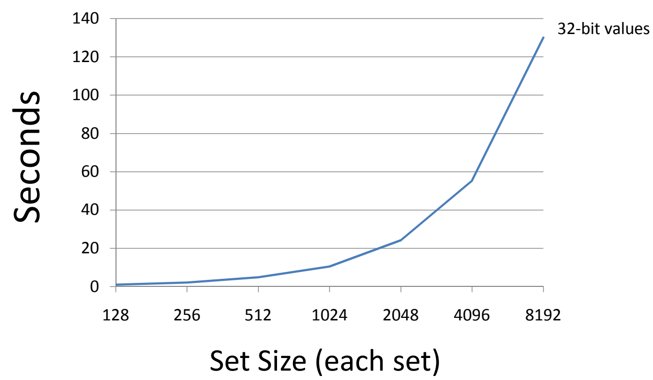
$$n\log(2n)\times 2\sigma$$

$$(3\sigma-1)(n-1)+(2\sigma-1)$$

Waksman Permutation Network

$$n\log n - n + 1$$

Private Set Intersection Results



Some Other Results

	Problem	Best Previous Result	Our Result	Speedup
USENIX Security 2011	**Hamming Distance** (Face Recognition) – 900-bit vectors	213s [SCiFI, 2010]	**0.051s**	**4176**
USENIX Security 2011	**Levenshtein Distance** (genome, text comparison) – two 200-character inputs	534s [Jha+, 2008]	**18.4s**	**29**
USENIX Security 2011	**Smith-Waterman** (genome alignment) – two 60-nucleotide sequences	[Not Implementable]	**447s**	-
USENIX Security 2011	**AES Encryption**	3.3s [Henecka, 2010]	**0.2s**	**16.5**
NDSS 2011	**Fingerprint Matching** (1024-entry database, 640x8bit vectors)	~83s [Barni, 2010]	**18s**	**4.6**

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

Collaborators

Yan Huang (UVa PhD Student),
Yikan Chen (UVa PhD Student),
Samee Zahur (UVa MS Student),
Peter Chapman (UVa BACS Student)
Jonathan Katz (University of Maryland)
Aaron Mackey (UVa Public Health Genomics)

David Evans
evans@cs.virginia.edu
http://www.cs.virginia.edu/evans