(Ever More) Efficient Protocols for Secure Computation

Jonathan Katz
Associate Professor
Dept. of Computer Science
\( (y_1, y_2, ..., y_n) = f(x_1, x_2, ..., x_n) \)
(y_1, y_2, ..., y_n) = f(x_1, x_2, x_3, x_4, x_n)
Drawbacks?

• May not be a single party trusted by everyone

• Trusted party learns all inputs + outputs
  – May be problematic for legal/policy reasons
  – Trusted party may later be compromised...

• Trusted party becomes a high-value target
  – If compromised, all security lost
A protocol is **secure** (within some specified threat model) if it **emulates** the use of a trusted party.

In particular:

- The computed results are **correct**
- Parties’ inputs remain **private** (except for what is implied by the outputs)
- Parties’ inputs are chosen **independently**
Threat models

• Assumed **bound** on the number of corrupted (and colluding) parties

• **Types** of misbehavior:
  – Semi-honest
  – Malicious
Applications?

• Distributed **auctions**
  – Used to compute the market-clearing price for sugar beets in Denmark (2008)

• Financial **auditing/supervision**
  – Compute HHIs for int’l institutions to look for unsafe levels of exposure

• **Privacy-preserving** malware analysis
Applications?

• Preventing satellite collisions
  – Sharemind (cf. youtube)

• Searching on encrypted data
  – (Without fully homomorphic encryption...)

• Funded projects by DARPA, IARPA, ERC, ...
Is secure computation possible?

Protocols for secure computation of **any** function, with security against **malicious** behavior of **any number** of parties, have been known since the ‘80s.

These protocols are **generic**, and work for any function represented as a **boolean circuit**.
Is secure computation practical?

- 1987: “Hopelessly inefficient”
- 2004: Fairplay (two parties, semi-honest)
- 2008: multi-party; malicious
- 2011: smartphone app

Asymptotic improvements, tailored protocols, (faster computers)
Progress (2-party, semi-honest)

Performance

Scalability

1 billion gates at 10 μs/gate
Generic vs. tailored protocols? (E.g., PSI)
### Other functions

<table>
<thead>
<tr>
<th>Problem</th>
<th>Best Previous Result</th>
<th>Our Result</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Hamming Distance</strong> (face recognition) – 900-bit vectors</td>
<td>213s [SciFI, 2010]</td>
<td><strong>0.051s</strong></td>
<td>4000x</td>
</tr>
<tr>
<td><strong>Edit Distance</strong> (genome, text comparison) – 200-character strings, 8-bit alphabet</td>
<td>534s [Jha+ 2008]</td>
<td><strong>18.4s</strong></td>
<td>30x</td>
</tr>
<tr>
<td><strong>Smith-Waterman</strong> (genome alignment) – 60-nucleotide sequences</td>
<td>447s [Not Implementable]</td>
<td><strong>0.2s</strong></td>
<td>16x</td>
</tr>
<tr>
<td><strong>Oblivious AES Evaluation</strong></td>
<td>3.3s [H^+10]</td>
<td><strong>0.2s</strong></td>
<td><strong>16x</strong></td>
</tr>
</tbody>
</table>

An alternative approach ... would have been to apply [a] generic secure two-party protocol.... This would have required expressing the algorithm as a circuit ... and then sending and computing that circuit.... **[We] believe that the performance of our protocols is significantly better than that of applying generic protocols.**

Osadchy et al., *IEEE Security & Privacy (Oakland), 2010*
Recent work

• Secure computation with sublinear work

• More efficient protocols with security against malicious behavior
  – Relaxed security guarantees ("one-bit leakage")
  – Full malicious security
Secure 2PC over “large data”?

- Consider lookup in a (sorted) database

Person 1  
Person 2  
...  
Person n

Is passenger $X$ on the no-fly list?

- The circuit for this function has size $\Omega(n)$
  - The circuit for any (non-trivial) function has size $\Omega(n)$

- Could have computed it (insecurely) in $O(\log n)$ time
Secure 2PC over “large data”?

(At least) linear complexity is *inherent* for secure computation of any “non-trivial” $f$

- If the server never touches the $i$th record, then it learns that the $i$th record was irrelevant
Secure 2PC over “large data”

Ideas:

1. Instead of secure 2PC based on circuits, explore secure 2PC based on RAMs

2. Consider secure 2PC in a setting where $f$ is evaluated \textit{multiple} times, and we aim for good \textit{amortized} complexity
Experimental results

(512-bit entries)
Malicious adversaries?

• In principle, a malicious adversary attacking a semi-honest protocol can completely violate privacy/correctness
  – Can we guarantee any protection?

• Best known malicious protocols roughly 200x slower than semi-honest protocols*

(* In recent work we improve this by a factor of 3)
Malicious adversaries?

• What if we **relax** the security requirements?

• **Here:** 1-bit leakage
  – The *best* a malicious adversary can do is learn is **one bit** of “disallowed” information
  – Cannot affect correctness
Performance

- Semi-honest
- DualEx (dual-core)
- DualEx (single-core)

Time (seconds)

- PSI (4096)
- ED (200x200)
- AES (100)
Current work

• Bridging **PL** and **cryptography**
  – Better compilers for secure computation
  – Better understanding of what *is* leaked by (even ideally) secure computation

• **Multiparty** computation

• Bridging **game theory** and **cryptography**
Conclusions

Secure computation is already practical
(for moderate-sized circuits,
and semi-honest security)

Privacy-preserving applications can run orders of magnitude faster than previously thought

Secure computation will be deployed in <10 years
Acknowledgments

• Collaborators
  – Yan Huang, Dave Evans, Lior Malka
  – Dov Gordon, Vlad Kolesnikov, Fernando Krell, Tal Malkin, Mariana Raykova, Yevgeniy Vahlis
  – Mike Hicks, Elaine Shi

• Research supported by
  – NSF ("TC: Large: Collaborative Research: Practical Secure Computation: Techniques, Tools, and Applications")
  – DARPA ("Toward Practical Cryptographic Protocols for Secure Information Sharing")
  – ARL-ITA ("Secure Information Flow in Hybrid Coalition Networks")
Selected publications


Papers and code linked from http://www.cs.umd.edu/~jkatz/papers.html