Applied MPC*

Wenting Zheng

*Some slide ideas stolen from Manoj Prabhakaran & Yuval Ishai — thanks!

Emulating trusted computation

- Goal of MPC is to emulate outsourcing computation to a trusted third party
  - Will not reveal secret inputs to other parties
  - Will not cheat in the computation
- Actual protocol: n parties emulate a trusted third party together

MPC

- $P_1 \ldots P_n$ want to securely compute $f(x_1, \ldots x_n)$
  - Up to t parties can collude
  - Corrupted parties should learn nothing but the output

Adversary’s power

- Corruption structure: honest majority or dishonest majority
- Semi-honest adversary: follows the protocol exactly
  - Tries to infer information about honest parties from what it observes
- Malicious adversary: can deviate from the protocol
  - Compute something different
  - Subbing in other party’s input as its own
  - Use inconsistent inputs

Auctions

Trading Sugar Beet Quotas - Secure Multiparty Computation in Practice

Special Theme 15 April 2008 Hits: 16109

Basic research in cryptology has shown that it is possible to combine and process information from several sources and at the same time control exactly what information is revealed. The technique is known as secure multiparty computation (SMC), and while the first solutions suggested were very inefficient, the current state of the art allows us to perform interesting computations.

In principle, the idea is simple: all necessary data are supplied and processed in encrypted form. The results made public will therefore only be those that the involved parties agree to have decrypted. Computing without actually looking at the data may seem impossible; nevertheless this is exactly the challenge that has been solved in basic research since the late 80s.

To gain an intuition into the techniques, consider the equation $g^x g^y = g^{x+y}$. Thinking of x and y as data and g^x as an 'encryption' of x, we see that multiplying encryptions adds the data implicitly without accessing it. The actual solutions use similar relations, and are based on both classical number theory and basic algebraic concepts such as polynomials over finite fields.

Auctions

- "*Several thousand Danish farmers produce sugar beets, which they sell to Danisco, the only Danish sugar producer*"
- Farmers want to trade contracts
- Double auction:
  - Farmers submit bids
  - Determine the "market clearing price", a price per unit at which all trade occurs
- "*The auction had a total of 1200 participating bidders. The actual computation took place on 14 January this year and lasted about thirty minutes. The result involved around 25,000 tons of production rights changing ownership; to our knowledge this was the first large-scale and genuinely practical application of SMC.*"

Electronic voting

- Vote tallying with potentially millions of parties
- Many issues
  - Only registered voters can vote
  - Vote only once
  - Voter cannot replace votes
  - Correct tallying of votes
  - Auditability
  - User anonymity
  - …

Real world implications

- Organizations can collaborate in spite of
  - Privacy policies
  - Being competitors
  - Fear of loss of control over data

Real world implications

- Why not always assume the stronger threat model?
  - Efficiency
  - Depends on the context (e.g., external attacker, subpoena)
- What isn’t MPC good for?
  - Hide leakage from the output
  - Enforce that a party picks the correct input

MPC systems/libraries

- Fairplay
- Sharemind
- SCAPI
- Obliv-C
- ObliVM
- JustGarble
- SPDZ
- AG-MPC

Efficient MPC?

- Typical trade off between generality and efficiency
- Many systems choose a tailored/hybrid protocol based on the specific application

Arithmetic MPC

- Secret sharing based
- Arithmetic circuit instead of boolean circuit
  - Addition and multiplication gates

Homomorphic encryption

- *Not fully HE, but partially HE*
- Example: Paillier encryption
  - $Enc(a) = g^a * r^N \bmod N^2$
  - $Enc(a + b) = Enc(a) * Enc(b)$
- Similar scheme used in Pretzel

Before we move on…

- Many real world use cases!
- Lots of real MPC libraries, with different security guarantees
- *Efficient* MPC requires a good understanding of the specific application
- Designing your own protocol is tricky! CS276 will help :)