Advanced crypto:

Zero-knowledge proofs and Secure Multi Party computations

Further Advanced Crypto techniques

- Zero-knowledge proofs
- Secure Multi-Party Computations

Zero-Knowledge Proofs

- Can we convince somebody that we know something (solution to the problem, a secret password, etc) without actually revealing it?
- Yes, we can! (starting from Goldwasser, Micali, Rackoff, 1980s)

Interactive zero-knowledge proof of 3-colorability

http://web.mit.edu/~ezyang/Public/graph/svg.html

Zero-knowledge proof for 3-colorabiity

- Problem:
  - How to colour vertices of a graph with no neighbouring vertices having the same colour?
  - In general it is a difficult NP-hard problem.
- For a given graph, can the Prover (Peggy) convince the Verifier (Victor) she knows a 3-coloring of the graph without revealing any information?

  Yes! In short:

1. Peggy hides her solution and allows Victor to open the colours of any pair of vertices
2. Then, Peggy reshuffles actual colours preserving 3-colorability, hides modified solution, and allows Victor to open the colours of any pair of vertices again;
3. Repeat step 2 until Victor convinced.

See detailed popular explanation at

https://blog.cryptographyengineering.com/2014/11/27/zero-knowledge-proofs-illustrated-primer

ZKP: Prover vs Verifier

Prover needs to convince verifier that he has a solution (information required)

Requirements for ZKP

Completeness: Prover should be able to convince Verifier that he has true solution (with high-probability)

Soundness: It should be able to convince Verifier only in true solutions

Zero-knowledge(ness): Verifier should not be able to learn anything but the fact Prover has a solution

Applications:

Zero-knowledge authentication; zero-knowledge information exchange contracts, verifiable computing, etc

Zero-Knowledge: interactive vs non-interactive

zk-SNARK:

- Zero-Knowledge Succinct *Non-Interactive* Argument of Knowledge

- “Prover sends a short message (evidence) to a Verifier to prove that he knows something (without revealing any additional information
- For example: prover may convince verifier that he has an input (message, number) which produces a given hash!
- Question: can you think about an application of such a technique?

zk-SNARK

- Small proofs and cheap verification
- But expensive prover computations (both time & space)
- Statement to be proved (F(x,w)=true) has to be presented by a circuit:
  - 10-20 millions gates max (2017) and to produce a proof it takes 1ms per gate

DIZK

- DIZK: A Distributed Zero Knowledge Proof System (H. Wu et al, UC Berkley, 2018)
- Distributes proofs computations across a computer cluster (Java & Apache Spark)
- Can generate proofs for the statements up to billions of circuits taking 0.01ms per gate Available at https://github.com/scipr-lab/dizk
- Examples of applications: authenticity of edited photo, integrity of machine learning models

Secure Multi Party Computations

- Several parties would like to compute something useful with their secret information, but would not like to share information.
  - Yao millionaire problem (1982): Alice and Bob would like to know who is richer without revealing their actual wealth.

Introductory explanation of a solution of a bit simpler problem

  - Candy problem: Alice and Bob would like to know whether they have the same number of chocolates without revealing to each other how many they have.

is at

https://hackernoon.com/eli5-zero-knowledge-proof-78a276db9eff

Short History of Secure MPC

- Yao, 1982: the idea of MPC, in following few years the first Garbled Circuits Protocol for generic MPC was introduced
- Malkhi et al, 2004: Fairplay implementation of generic SMPC (median of two sorted arrays of ten integers in 7s)
- 2004-now, speed of SMPC increased in 5 orders of magnitude

(D Evans et al, A Pragmatic Introduction to Secure Multi-Party Computation, 2018)

Applications of SMPC

- Danish sugar beets auction (2009);
- Estonian Student Study (2015);
- Boston Wage Equity study (2017) ...

Secure Auctions, Voting, Secure Machine Learning, Privacy-Preserving Genomics, etc

How does it work? Simplified Yao's Protocol

- We would like to compute F(x,y) where P1 holds $x \in X$ and P2 holds $y \in Y$ as their private values;
- P1 calculates and tabulates all values of F(x,y) $x \in X$ and $y \in Y$ in the table $T = (T_{x,y})$
- P1 encrypts each $T_{x,y}$ with two random keys k_x and k_y, randomly permutes the table T and sends it to P2
- Now, in order for P2 to compute F(x,y), P1 sends k_x and k_y to P2, who after that can complete computation

How does it work? Simplified Yao's Protocol

- We would like to compute F(x,y) where P_1 holds $x \in X$ and P_2 holds $y \in Y$ as their private values;
- P_1 calculates and tabulates all values of F(x,y) $x \in X$ and $y \in Y$ in the table $T = (T_{x,y})$
- P_1 encrypts each $T_{x,y}$ with two random keys k_x and k_y, randomly permutes the table T and sends it to P_2
- Now, in order for P_2 to compute F(x,y), P_1 sends k_x and k_y to P_2, who after that can complete computation
- **Question:** How does P_1 know which k_y to send to P_2? P_1 knows x, but not y!
- **Answer:** It is done using *oblivious transfer*!

Oblivious Transfer (OT)

- **Given:** Sender S has input secrets x_0 and x_1 (binary strings) and Receiver R has a selection bit $b \in \{0,1\}$
- **After executing OT:** Receiver gets x_b and Sender does not learn anything about b
- Can be implemented using public-key encryption *(can you propose how to do it?)*
- The above version is 1-out-of-2 OT, for simplified Yao's protocol we need 1-out-of-Y OT, which can be implemented similarly.

From Theory to Applications

- JIFF library developed in Boston University (multiparty.org)

- *JIFF is a JavaScript library for building applications that rely on secure multi-party computation. JIFF is built to be highly flexible with a focus on usability, with the ability to be run in the browser, on mobile phones, or via Node.js. JIFF is designed so that developers need not be familiar with MPC techniques or know the details of cryptographic protocols in order to build secure applications.*

  (from https://github.com/multiparty/jiff|)

Advanced Crypto to Applications

- All these technologies:
  - Fully Homomorphic Encryption (FHE)
  - CryptDB-like solutions
  - Zero-knowledge proofs (ZKP)
  - Secure Multi-Party Computations (SMPC)
  - Verifiable Computing

  are at the beginning of exciting applications.