Alin Tomescu, CSE408
Thursday, March 24nd, Lecture #16

Bit commitment, zero knowledge proofs

Bit commitment

Inspiration: Imagine we wanted to play poker over the telephone (or online). Someone could say he has 4 aces. We need a system where I can commit to my cards, and later on I can reveal my cards. The goal is that you can't figure out what my cards were from my commitments.

Model: We've got Alice and Bob. Alice wants to commit to some value v (could be a single bit could be more). She needs to compute $c = commit(v)$ (there might be no secret key involved). She sends c to Bob, and later on she could send v to Bob, and Bob can verify that $c = commit(v)$, but he cannot infer any info about v from c.

Example: To implement $commit(v)$, Alice can send $hash(v)$ or $f(v)$ to Bob where f is a OWP/F. If v is small, you can pad it to ensure Bob won't brute-force $commit(v)$.

Security goals:

1. Alice can't change her mind after committing.
    a. This means it should be difficult to find x_1 and x_2 such that $H(x_1) = H(x_2)$.
    b. This is called **strong collision resistance.**
2. Bob can't learn any info about v from $commit(v)$
    a. This means the hash can't be invertible.

Bit commitment using El Gamal

Turns out **El Gamal** is very good for solving the bit commitment problem.

Everyone has g and p.

Alice:

- has v, a secret a, and computes $g^a \bmod p$
- she picks $r \leftarrow u$ and sends $(g^a \bmod p, g^r \bmod p, g^{ar}v \bmod p)$ to Bob
    - by the Diffie-Hellman problem, you can't figure out what v is
- how will Alice reveal what v is?
    - She can just send (a, r, v) to Bob, and Bob will do the verification
        - Bob will first compute $g^a \bmod p$, and check he got the same result as Alice
        - then he will compute $g^{ar} \bmod p$
        - then he will divide what he got from Alice by $g^{ar} \bmod p$, getting v
            - $\frac{g^{ar}v \bmod p}{g^{ar} \bmod p} = v$

Can Alice change her mind? We can set this up so that there's only a single value a that maps to $g^a \bmod p$, and do the same thing for r. So Alice won't even send v and Bob can confirm that $g^a = x, g^r = y, v = z/g^{ar}$, where $z = g^{ar}v \bmod p$ (sent by Alice initially).

We can simplify this scheme by fixing g^a, sending it to Bob and having Alice just send $(g^r \bmod p, g^{ar}v \bmod \mathrm{p})$.

Alin Tomescu, CSE408
Thursday, March 24nd, Lecture #16

Zero knowledge proofs

Zero knowledge proofs: Huge area with a lot of cool facts.

Idea behind them is that you can prove some fact to a remote party without revealing any information about how you're doing the proof.

Idea: Graph coloring. Figuring out whether a graph has a k-coloring is NP-complete.

Imagine that I have some graph, and you know it too, and I have somehow discovered a k-coloring. I want to prove to you I discovered this coloring but I don't want you to know what it is.

Suppose that I discovered $c_1, c_2, \dots, c_k$. I can just shuffle the colors around and it'll still be a valid coloring.

Alice is the *prover*, **Bob** is the *verifier*.

- Alice will shuffle the colors, $shuffle(c_1, c_2, \dots, c_k)$.
- Alice sends the commitments of the colors of each vertex:
  - $\forall v \in V(G)$ she commits to $c(v)$, so she sends $Commit\big(c(v)\big)$ for each vertex in some known order.
  - **Warning:** If the commitment is the deterministic then Bob could figure out the coloring, so we'd better use something like the El Gamal scheme above.
- Bob picks a random edge in the graph and he just sends (v_i, v_j) to Alice where $(v_i, v_j) \in E(V)$
- Alice decommits $v_i \; and \; v_j$ by sending $Decommit(v_i)$, and $Decommit(v_j)$
- Bob has to check that that the coloring is valid (and that Alice hasn't changed her initial commitment)
- Suppose they repeat this $l \times |E(V)|$ (with Alice reshuffling every time)

Let's look at the chances of Alice winning this game. If Alice sends a bad coloring, there will be some adjacent vertices with identical colors. $\exists\big(v_i, v_j\big) \in E(V) \text{ s.t. } color(v_i) = color(v_j)$

$$\Pr[Alice\ is\ not\ caught] = \left(1 - \frac{1}{|E|}\right)^{l|E|} \approx e^{-l}$$

Information extractors: Save the state of Alice (as a Turing machine) after she commits, and extract the coloring of the graph.

Is it zero-knowledge?

Could Alice and Bob fool an observer that Alice has a k-coloring? What if Alice and Bob are staging the game?

Argument: Bob doesn't learn anything because the observer is not convinced, since Alice and Bob could be staging the game.

Everything that Bob learns in the protocol is just the transcript of the messages exchanged. These messages are kind of randomized. Bob can generate one of these transcripts on his own. If Bob doesn't know a coloring, then Bob can pretend to be Alice and "fake" the game. Bob can generate valid transcripts. Therefore, Bob learns nothing from the protocol.

Another zero-knowledge protocol

Suppose Alice knows a and bob knows $g^a \bmod p$. g and p are fixed and known by both.

- Alice sends $t = g^r$

Alin Tomescu, CSE408
Thursday, March 24nd, Lecture #16

- Bob sends $c = random$
- Alice sends $s = r + ac \bmod p - 1$
- Bob verifies that $g^s = t(g^a)^c$

Information extractor: Freeze Alice's state after sending t, then repeat Bob's stuff for a few times, and using linear equations you can figure out a.