SPCS Cryptography Class Lecture 11

July 6, 2015

- So far we have focused on two-people scenarios, where Alice and Bob want to send a message to each other securely.
- Some other situations that can arise:
  - Coca Cola and its secret formula.
    - President would have access to the secret formula.
    - In urgent situations, perhaps any three out of five associates should be able to access the secret as well.
  - Voting:
    - I want to figure out the average age of this class.
    - No one wants to tell others how old they are.
    - How can I figure this out?
- In these two days we will introduce some protocols that would be useful in such situations - today we will talk about secret sharing schemes and zero-knowledge proofs.

Secret sharing schemes

Zelda has a secret number s. She wants to give each of Alice, Bob some information so that

- Alice or Bob alone cannot figure out the secret.
- Alice and Bob together can figure out the secret.

How?

Secret sharing schemes

Method 1: Splitting numbers

- Give first half of s to Alice, second half of s to Bob.
- Is this good?

Some concerns when constructing a secret sharing schemes:

- Correctness: Alice and Bob together can definitely recover the secret.
- Privacy: Alice or Bob alone cannot get any information about the secret.
- Any improvement to Method 1?

Secret sharing schemes

Method 2: Shamir's secret sharing

- Zelda takes a prime $p > s$.
- Zelda takes a random $m \in \mathbb{F}_p$.
- Consider the function $f(x) = mx + s$ from $\mathbb{F}_p$ to itself.
- Zelda gives Alice $(1, f(1))$. Zelda gives Bob $(2, f(2))$.
- How can Alice and Bob figure out the secret together?
- Can Alice or Bob alone figure out what s is?
- Is this safe?

Secret sharing schemes

Method 3: Blakley's secret sharing

- Zelda picks a random number r, and encode her secret as (r, s), a point on the plane.
- Zelda gives Alice one line that passes the point. Zelda gives Bob another line that passes the point.
- How can Alice and Bob figure out the secret together?
- Can Alice or Bob alone figure out what s is?
- Is this safe?
- Is it better or worse if Zelda uses both coordinates to encode her secret?

Method 4: Mignotte's secret sharing

- Zelda picks two distinct primes p_1, p_2 such that $s < p_1 p_2$.
- Zelda gives Alice $s \bmod p_1$, and gives Bob $s \bmod p_2$.
- How can Alice and Bob figure out the secret together?
- Can Alice or Bob alone figure out what s is?
- Is this safe?

Secret sharing schemes

Now let's say Zelda wants to share secret s with Alice, Bob and Carol.

- Does Method 1 (splitting of numbers) work?

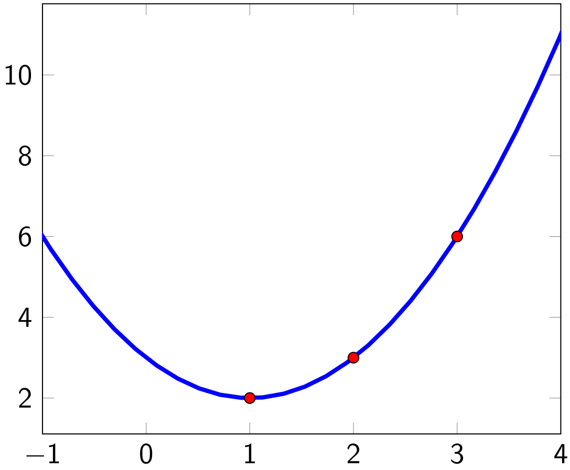
Secret sharing schemes

Does Method 2 (Shamir's scheme using polynomials) work?

- Zelda takes a prime $p > s$.
- Zelda takes random $m_1, m_2 \in \mathbb{F}_p$.
- Consider the function $f(x) = m_1 x^2 + m_2 x + s$ from $\mathbb{F}_p$ to itself.
- Zelda gives Alice $(1, f(1))$, Bob $(2, f(2))$, and Carol $(3, f(3))$.
- How can Alice, Bob and Carol figure out the secret together?
- Can at most two of them figure out what s is?
- Do they gain information though?

Lagrange interpolation

- Given three non-collinear points on the plane, exactly one quadratic polynomial passes through them.



Lagrange interpolation

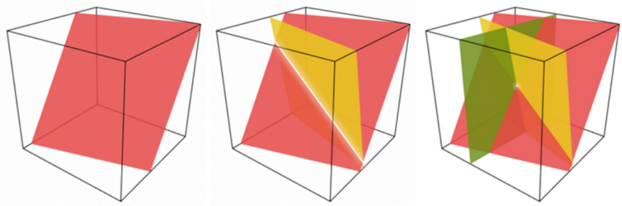
- The three points are $(1, 2), (2, 3), (3, 6)$.
- How to write the quadratic polynomial down directly?
- Lagrange's interpolation method:

$$2 \cdot \frac{(x-2)(x-3)}{(1-2)(1-3)} + 3 \cdot \frac{(x-1)(x-3)}{(2-1)(2-3)} + 6 \cdot \frac{(x-1)(x-2)}{(3-1)(3-2)}$$

- In this case, it simplies to $x^2 - 2x + 3$.

Does Method 3 (Blakley's scheme using planes and lines) work?

- Zelda picks two random numbers r_1, r_2, and encode her secret as (r_1, r_2, s), a point on the plane.
- Zelda gives Alice one plane that passes the point, Bob another plane that passes the point, Carol another plane that passes the point.
- What should Zelda pay attention to when she gives out planes?



http://qz.com/97885/

- How can Alice, Bob and Carol figure out the secret together?
- Can at most two of them figure out what s is?
- Do they gain information though?

Secret sharing schemes

Does Method 4 (Mignotte's scheme using Chinese Remainder Theorem) work?

- Zelda picks three distinct primes p_1, p_2, p_3 such that $s < p_1 p_2 p_3$.
- Zelda gives Alice $s \mod p_1$, gives Bob $s \mod p_2$, and gives Carol $s \mod p_3$.
- How can Alice and Bob figure out the secret together?
- Can at most two people figure out what s is?
- Do they gain information though?

Secret sharing schemes

Actually, Zelda has many friends (say n of them) and want to share one secret with them, so that only when all of them are present they can find out the secret.

- Does Method 1 (splitting of numbers) work?
- Does Method 2 (Shamir's scheme using polynomials) work?

Example

For example, four points determine a cubic polynomial.
If $f(1) = 1$, $f(2) = 3$, $f(3) = 6$, $f(4) = 12$, the unique cubic polynomial passing through all of them is

$$1 \cdot \frac{(x-2)(x-3)(x-4)}{(1-2)(1-3)(1-4)} + 3 \cdot \frac{(x-1)(x-3)(x-4)}{(2-1)(2-3)(2-4)} + 6 \cdot \frac{(x-1)(x-2)(x-4)}{(3-1)(3-2)(3-4)} + 12 \cdot \frac{(x-1)(x-2)(x-3)}{(4-1)(4-2)(4-3)}$$

Secret sharing schemes

- Does Method 3 (Blakley's scheme using planes and lines) work?
- Does Method 4 (Mignotte's scheme using Chinese Remainder Theorem) work?

A variant: Zelda gives the secret to n friends, so that if k of them are present, they can find out the secret.

- Does Method 1 (splitting of numbers) work?
- Does Method 2 (Shamir's scheme using polynomials) work? (What is the degree of polynomial should you choose?)
- Does Method 3 (Blakley's scheme using planes and lines) work?
- Does Method 4 (Mignotte's scheme using Chinese Remainder Theorem) work?

These secret sharing schemes are (k, n)-threshold secret sharing schemes.

Secret sharing schemes

Yet another variant: Zelda has trust issues - she trusts Alice more than Bob, Carol or Donna, so she wants the secret to be revealed when

- Alice and one of Bob, Carol or Donna are present.
- Bob, Carol, and Donna are all present.

How can she do that?

- One idea: a *weighted* threshold scheme.
- More generally, one can talk about an access structure - all the possible combinations of people that can access the secret.
- A minimal access structure are all the non-redundant combinations.
- For example, in our case,

  {{Alice, Bob}, {Alice, Carol}, {Alice, Donna}, {Bob, Carol, Donna}}

  form a minimal access structure.

Secret sharing schemes

- An access structure should be monotone - if Alice and Bob together can recover the secret, there is no reason why {Alice, Bob, Carol} can't do the same.
- Given a monotone access structure, does one have a secret sharing scheme that fulfill correctness and privacy properties?
- Yes! See for example, Ito-Saito-Nishizeki 87' or Benaloh-Leichter'98.

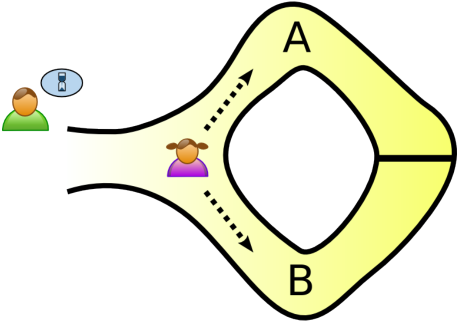
Secret sharing schemes

Some applications:

- Accessing Hardware Security Modules. (HSM) See http://cloudhsm-safenet-docs.s3.amazonaws.com/007-011136-002_lunasa_5-1_webhelp_rev-a/Content/concepts/mofn_about.htm
- DNSSEC root key. See https://www.schneier.com/blog/archives/2010/07/dnssec_root_key.html
- More secure password storage, so that leakage of a single server's data does not allow dictionary attack. (Verisign/Symantec)
- An example: http://passguardian.com/
- An implementation: http://point-at-infinity.org/ssss/

Zero-knowledge proofs

Peggy, Victor, and Alibaba's cave.



https://upload.wikimedia.org/wikipedia/commons/d/dd/Zkip_alibaba1.png

- Peggy claims to know the magic word to open the door in the cave.
- Victor wants to verify that, but Peggy does not want to tell him the secret.
- What can Victor do?

Zero-knowledge proofs

Properties of a zero-knowledge proof:

- Completeness: if Peggy knows the magic word, Victor would be convinced.
- Soundness: if Peggy does not know the magic word, Victor would not be convinced (except for a very small probability).
- Zero-knowledge: Victor doesn't know Peggy's magic word!

More examples:

Example

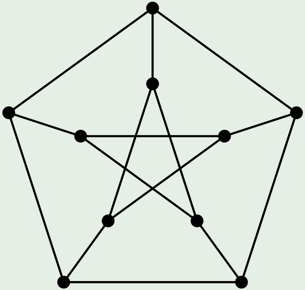
- Victor is color-blind - he cannot distinguish red and green.
- Peggy claims that red and green are different.
- Victor has one red ball and one green ball

Can Peggy prove to him that she can distinguish red and green?

Zero-knowledge proofs

Example (3-coloring problem)

- Given a graph, a 3-coloring is a coloring of each vertex by one of the three colors so that adjacent vertices have different colors.



How can Peggy prove to Victor that she knows how to solve the 3-coloring problem, without ever telling him the algorithm?

Fiat-Shamir identification protocol

One application of zero-knowledge proof is the Fiat-Shamir identification protocol.

Fiat-Shamir

Peggy wants to prove her identity to Victor without ever revealing her secret.

Pre-processing:

- Peggy chooses $n = pq$, where p, q are distinct large primes.
- She also picks a square $v = u^2 \mod n$.
- Finally she chooses the smallest square root s of $v^{-1} \mod n$ - she can compute s because she can factor n.
- Peggy makes (n, v) public, and s private.

Fiat-Shamir identification protocol

Fiat-Shamir

Verification:

- Peggy wants to prove her identity to Victor. She chooses a random $r \mod n$, and sends Victor $x = r^2 \mod n$.
- Victor chooses random $b = 0$ or 1, and send it to Peggy.
- Peggy computes $y = rs^b \mod n$ and sends to Victor.
- Victor verifies that $y^2 = xv^b \mod n$.

- This is based on the hardness of taking square roots mod n, without knowing the factorization of n.
- What if Mallory tries to imitate Peggy?
- Does Victor learn what s is?
- Along the way Peggy and Victor have interactions - thus this is an interactive zero-knowledge proof.

Schnorr identification protocol

Another example is Schnorr's identification protocol, based on hardness of discrete log problem.

Schnorr's identification protocol

Peggy wants to prove her identity to Victor without ever revealing her secret.

Pre-processing:

- Peggy chooses large prime p, primitive root $g \mod p$.
- She also picks her private key $x \mod p-1$, and computes $y \equiv g^x \mod p$.
- Peggy makes (p, g, y) public, and x private.

Schnorr identification protocol

Schnorr's identification protocol

Verification:

- Peggy wants to prove her identity to Victor. She chooses a random $k \bmod p-1$, and sends Victor $r = g^k \bmod p$.
- Victor chooses random $e \bmod p-1$, and send it to Peggy.
- Peggy computes $s = k - xe \bmod p-1$, and sends it to Victor.
- Victor verifies that $g^s \equiv ry^e \bmod p$.

- Why does this work?
- Would Victor learn about Peggy's secret?
- This is again an interactive zero-knowledge proof.
- Can we make things non-interactive?

Coin-flipping problem

Example

Alice and Eve have to settle an issue over the phone. They decide to settle it by a coin flip. Obviously, Alice and Eve don't trust each other. (You probably know why..) How can they do that?

Solution

One method is to use a one-way hash function.

- *Alice and Eve agree on a one-way hash function h.*
- *Alice makes a guess G, and sends Eve the hash of the guess $h(G)$.*
- *Eve flips a coin and tells Alice the result.*
- *Alice tells Eve her guess G and declare wining/losing.*
- *Eve can verify Alice's guess by computing the hash $h(G)$.*

Is this fair?

Schnorr signature scheme

- In the coin-flipping situation, Alice *committed* to her guess before Eve flips the coin.
- She cannot change her guess afterwards as long as the hash function is pre-image resistant.
- This is an example of *commitment*. Commitment schemes can be used to make sure that you cannot change your "random" choice afterwards, even though Victor does not know the choice yet at the moment.
- This can be used to make zero-knowledge proof non-interactive.

Schnorr signature scheme

Alice wants to send Bob a message, and she wants to sign the message.

- They agree on a large prime p and a primitive root $g \bmod p$, and a one-way hash function H.
- Alice chooses a private key $x \bmod p-1$. She publishes the public key $y \equiv g^x \bmod p$.
- To sign the message m, Alice generates a random $k \bmod p-1$, and computes $r \equiv g^k \bmod p$. This is her *commitment*.
- Alice computes $e = H(m||r)$. This is the *challenge*.
- Alice computes $s \equiv k - xe \bmod p-1$. This is her *response* to challenge.
- Alice sends Bob the message m with the signature (r, s).

Schnorr signature scheme

Schnorr signature scheme

Verification:

- Bob re-computes $e = H(m||r)$.
- Bob calculates $r' \equiv g^s r^e \mod p$.
- Bob verifies $r' = r$.

- Why does it work?
- Does Bob learn about Alice's secret?

Schnorr signature scheme

Secure Remote Password Protocol (SRP)

- Another application of zero-knowledge proof is SRP.
- Advantage: it does not store password-equivalent data on server - it only stores the verifier.

The naive password protocol:

- When user registers, store the hash of the password.
- When user logins, compare the hash of user's entry with the stored hash.
- Is this good?
- Example: the md5 hash of a possible password 12345678 is

  25d55ad283aa400af464c76d713c07ad
- This is a *dictionary attack*.
- A fix: salted hash - one with padding.

Secure Remote Password Protocol (SRP)

SRP

Registration:

- p is a large prime, g is a primitive root mod p. $H(\cdot)$ is a hash function.
- To establish a password PWD with Steve, Alice picks a small random salt s, computes $x = H(s||PWD)$.
- Alice also computes $v \equiv g^x \mod p$.
- Alice sends Steve (v, s). (Verifier, and Salt)

Secure Remote Password Protocol (SRP)

SRP

Verification:

- When Alice tries to log in, she picks a random a mod $p-1$, and sends Steve $A \equiv g^a \mod p$.
- Steve picks a random b mod $p-1$. He then sends Alice the salt s, and $B = kv + g^b$. Here k is a pre-determined number known by both Alice and Steve.
- Both side compute $u = H(A||B)$.
- Alice calculates $S_{Alice} = (B - kg^x)^{a+ux}$, and hash the result $K_{Alice} = H(S_{Alice})$.
- Steve calculates $S_{Steve} = (Av^u)^b$, and hash the result K_{Steve}.
- They verify that $K_{Alice} = K_{Steve}$. This way both ways are authenticated without PKI.

Secure Remote Password Protocol (SRP)

Why does it work?

- $S_{Alice} = (B - kg^x)^{a+ux}$, which is

$$(B-kg^x)^{a+ux} = (kv+g^b-kg^x)^{a+ux} = (kg^x+g^b-kg^x)^{a+ux} = g^{b(u+ax)}$$

- $S_{Steve} = (Av^u)^b$, which is

$$(Av^u)^b = (g^a \cdot (g^x)^u)^b = (g^{a+ux})^b$$

- So actually $S_{Alice} = S_{Steve}$.

Does Steve ever know about Alice's password?

- Stored information: Salt s and verifier $v \equiv g^x \mod p$, where $x = H(s||PWD)$.
- What Alice sent: $g^a \mod p$ - since $a \mod p-1$ is random, this is random information, and is not related to PWD.
- As hard as discrete log problem!

Secure Remote Password Protocol (SRP)

- Heartbleed attack - a vulnerability of SSL standard discovered in 2014 that may allow attacker to get server private key.
- This means that if the server stores password-equivalent data - your password is leaked! You can be impersonated.
- This is why a protocol like SRP is useful - it has forward secrecy.
- Also allows mutual authentication without PKI.
- One of the available methods in TLS (TLS-SRP), but not widely used yet.