T-79.159 Cryptography and Data Security

Lecture 9: Secret Sharing, Threshold Cryptography, MPC

Helger Lipmaa

Helsinki University of Technology

helger@tcs.hut.fi

Outline of the lecture

- Secret Sharing
- Threshold Encryption
- Secure Multi-Party Computation

Key storage: problems

- Reliability and confidentiality of important data:
  - ⋆ Information can be secured by encryption
  - ⋆ After that, many copies of the ciphertext can be made
- How to secure the secret key?
  - ⋆ Encrypting of key — vicious cycle
  - ⋆ Replicating key — insecure
- Idea: Distribute the key to a group, s.t. nobody by itself knows it

Secret Sharing: More Motivations

- USSR: At least two of the three nuclear buttons must have been prssed simultaneously

- Any other process where you might not trust a single authority

- Threshold cryptography, multi-party computation:

  ⋆ Computation can be performed in a distributed way by "trusted" subsets of parties

- Verifiable SS: One can verify that inputs were shared correctly

Secret sharing schemes: Definition

- A dealer shares a secret key between n parties
- Each party $i \in [1, n]$ receives a share
- Predefined groups of participants can cooperate to reconstruct the shares
- Smaller subgroups cannot get *any* information about the secret

(k, n)-threshold schemes: Definition

- A dealer shares a secret key between n parties

- Each party $i \in [1, n]$ receives a share

- A group of any k participants can cooperate to reconstruct the shares

- No group of $k - 1$ participants can get *any* information about the secret

Example (bad)

- Let K be a 100-bit block cipher key. Share it between two parties giving to both parties 50 bits of the key

- Why is this bad?

  $\star$ The requirement 'Smaller subgroups cannot get *any* information about the secret' is violated

- Ciphertext-only attack: Both participants can recover the plaintext by themselves, by doing a 2^{50}-time exhaustive search

$(2,2)$-threshold scheme

- Let $s \in G$ be a secret from group $(G, +)$. Dealer chooses a uniformly random $s_1 \leftarrow_R G$ and lets $s_2 \leftarrow s - s_1$

- The two shares are s_1 and s_2

- Given s_1 and s_2 one can successfully recover $s = s_1 + s_2$

- Given only s_i, $i \in [1, 2]$: s_{2-i} is random

$$\Pr[s = k \mid s_2] = \Pr[s_1 = k - s_2 \mid s_2] = 2^{-|G|} \quad \text{for any } k.$$

Note: group ciphers

- Recall: Group cipher $E_k(m) = k + m$ (additive group)
- Group cipher is *perfect* (Shannon): $\mathsf{Pr}[m | E_k(m)] = \mathsf{Pr}[m]$
- Group ciphers can be used as $(2, 2)$-threshold schemes, $s_1 = k$, $s_2 = D_{s_1}(s) = s - s_1$
- $(2, 2)$-threshold schemes can be used as perfect ciphers with plaintext s, key s_1 and ciphertext s_2
- Really: it will be impossible to get any information about s without knowing *both* key and ciphertext

(n, n)-threshold scheme

- Let s be a secret from group G. Dealer chooses an m-bit uniformly random $s_1, \ldots, s_{n-1}$ and computes $s_n = s - (s_1 + \cdots + s_{n-1})$

- The shares are $(s_1, \ldots, s_n)$

- Given $(s_1, \ldots, s_n)$, one can successfully recover $s = s_1 + \cdots + s_n$

- Given s_i for $i \neq j$: $\sum_{i \neq j} s_i = s - s_j$ is random — no information about s

Shamir's (k, n)-threshold scheme

Mathematical basis:

- Given k points on the plane $(x_1, y_1), \ldots, (x_k, y_k)$, all x_i distinct, there exists an unique polynomial f of degree $\leq k - 1$, s.t. $f(x_i) = y_i$ for all i
  - $\star$ Constructive proof: Given these k points, one can recover f by using the *Lagrange interpolation formula*
- This holds also in the field $\mathbb{Z}_p$, p prime

Shamir's (k, n)-threshold scheme

Description. Dealing phase:

- Let s be a secret from some $\mathbb{Z}_p$, p prime
- Select a random polynomial $f(x) = f_0 + f_1 x + f_2 x^2 + \cdots f_{k-1} x^{k-1}$, under the condition that $f(0) = s$:
  - $\star$ Select $f_1, \ldots, f_{k-1} \leftarrow_R \mathbb{Z}_p$ randomly
  - $\star$ Set $f_0 \leftarrow s$
- For $i \in [1, n]$, distribute the share $s_i = (i, f(i))$ to the ith party

Shamir's (k, n)-threshold scheme

Theorem The secret s can be reconstucted from every subset of k shares.

Proof: By the Langrange formula, given k points (x_i, y_i), $i = 1, \ldots, k$,

$$f(x) = \sum_{i=1}^{k} y_i \prod_{j=1, j \neq i}^{k} \frac{x - x_j}{x_i - x_j} \pmod{p}$$

and thus

$$s = f(0) = \sum_{i=1}^{k} y_i \prod_{j=1, j \neq i}^{k} \frac{-x_j}{x_i - x_j} \pmod{p} \ .$$

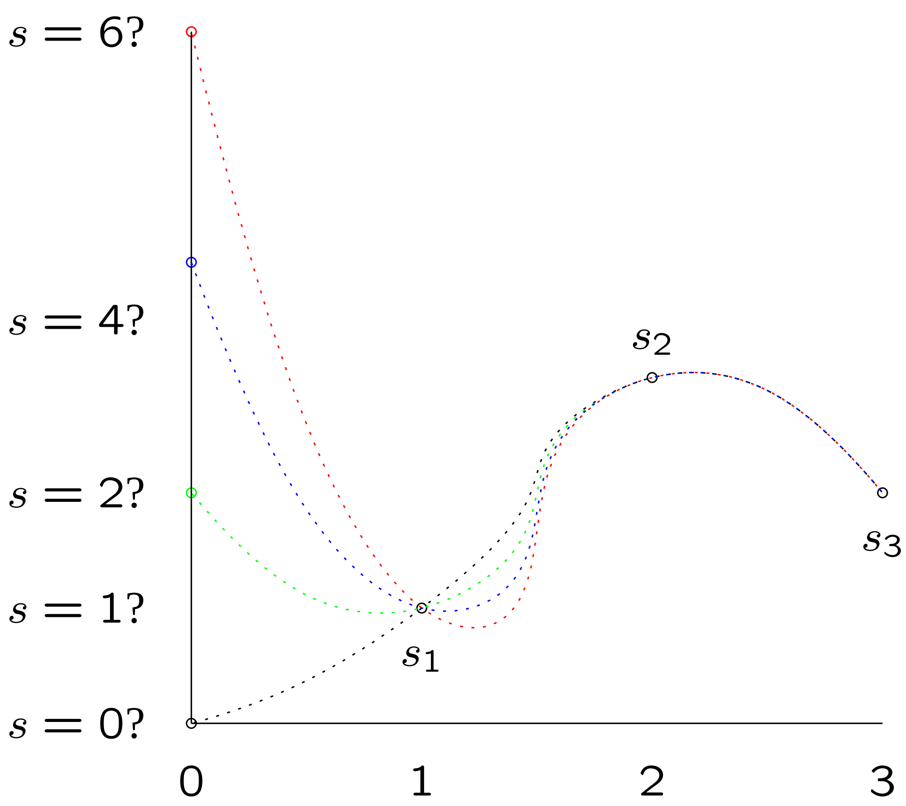
Shamir's (k, n)-threshold scheme

Theorem Any subset of up to $k-1$ shares does not leak any information on the secret.

Proof: Given $k-1$ shares (x_i, y_i), *every* candidate secret $s' \in \mathbb{Z}_p$ corresponds to an unique polynomial of degree $k-1$ for which $f(0) = s'$. From the construction of polynomials, for all $s' \in \mathbb{Z}_p$, probabilities $\Pr[s = s']$ are equal. Q.E.D.

Conclusion: Shamir's scheme is perfectly secure and does not depend on the computational power of any party.

Security of Shamir's scheme illustrated



Shamir's scheme: Effiency

- Lagrange interpolation requires $O(k \log^2 k)$ steps.

- Instead of sharing a singe long s, one can divide s into j smaller pieces and share every piece. Complexity reduces from $O(k \log^2 k)$ to $O(k(\log k - \log j)^2)$

- Size of each share s_i = size of the secret s

Shamir's scheme: Flexibility

- One can increase n and add new shares without affecting other shares
- Existing shares can be removed without affecting other shares (as long as the share is really destroyed)
- It is possible to replace all the shares (or even k) without changing the secret and without revealing any information on the secret by selecting a new polynomial $\widehat{f}(x)$ and a new set of shares
- Some parties can be given more than one share

Shamir's scheme: Remarks

- Example: the president has 3 shares, prime minister has 2 shares, other ministers have 1 share. Then by using a $(3, n)$-threshold scheme the secret will be recovered by
  - $\star$ the president, or
  - $\star$ the prime minister and another minister, or
  - $\star$ any three ministers.
- Shamir's scheme = Reed-Solomon error-correcting code

General Secret Sharing

- Assume authorized sets have the monotonicity property: if A is authorized and $A \subseteq B$ then B is authorized

- The set of authorized sets is called the *access structure*

- Brickell etc: Any monotone access structure $\mathcal{A}$ is valid

- That is, there exists a secret sharing scheme where sets from $\mathcal{A}$ can find the secret, and other sets will get no information about the secret

Threshold Cryptosystems

- Goal:
  - ⋆ Private key is shared among a set of receivers, so that
  - ⋆ Only authorized sets of users can decrypt messages
- Key generation protocol G: key is generated jointly by all participants
- Encryption protocol E: (ideally) it is hidden from the sender that the cryptosystem is thresholded
- Decryption protocol D: An authorized set can decrypt a ciphertext without explicitly reconstructing the private key

Threshold ElGamal Cryptosystem

- Secret $s \in \mathbb{Z}_p$
- Every participant A_j possesses a share s_j, where s_j was generated according to Shamir's scheme
- A_j commits to share s_j by publishing

$$h_j = g^{s_j} \quad .$$

Threshold ElGamal Cryptosystem, cont.

- **Correctness:** From the Lagrange IF, since $s = \sum c_j s_j$ for some c_j, then g^s can be established as $\prod_{j \in X} (g^{s_j})^{c_j}$ from public values alone, where X is any subset of k authorities

- **Security:** No single participant learns s, but s is only computationally hidden (w.r.t. the DL problem)

- $h = g^s$ is announced as the public key

Threshold ElGamal: Decryption

Recall: $h = g^s$, $s = \sum c_j s_j$. To decrypt $(y, x) = (mh^r, g^r)$, the users A_j do:

1. Each A_j broadcasts $w_j = x^{s_j}$, and proves in ZK that $\log_g h_j = \log_x w_j$
2. Let X be any subset of k authorities who passed the ZK proof. The plaintext can be recovered as
$$m' = \frac{y}{\prod_{j \in X} w_j^{c_j}}$$

Correctness: $w_j^{c_j} = x^{c_j s_j} = g^{r c_j s_j}$, thus $m' = m g^{rs} / \prod g^{r c_j s_j} = m$.

How to prove equality of DLs?

A proves $PK(x = g^\mu \wedge y = h^\mu)$:

$$
\begin{array}{lcl}
\quad\quad A & & \quad\quad\quad\quad B \\
r \leftarrow_R \mathbb{Z}_q; a := g^r, b := h^r & \xrightarrow{(a,b)} & \\
 & \xleftarrow{c} & c \leftarrow \{0,1\}^{80} \\
z \leftarrow r + \mu c & \xrightarrow{z} & g^z \stackrel{?}{=} a x^c, h^z \stackrel{?}{=} b y^c
\end{array}
$$

(Chaum-Pedersen. Note similarity to the Schnorr protocol.)

Exercise: Prove that it is secure!

E-voting/auctions again

- In the previous lecture, talking about auctions, we said that a cheating authority can get additional information

- Idea: use a threshold homomorphic encryption

  ⋆ Homomorphism allows limited computation with shares

E-voting (Cramer, Gennaro, Schoenmakers)

- ith voter encodes and encrypts his vote b_i as $c_i = E_K(B^{b_i})$, by using the threshold ElGamal. She broadcasts c_i to all n authorities A_j
- A_j gathers all c_i and computes his local copy of $c = \prod c_i$
- Authorities compare their copies of c
- If we assume that $k > n/2$ authorities are correct then majority of c-s coincide
- Use any subset of k authorities from this majority to decrypt c. Compute the votes per candidate from c

Multi-party computation

- We saw how to do limited computation (decryption, plaintext addition) in a threshold manner
- How to do every computation?
- Is it possible to do every computation in a threshold manner? Yes!
- Idea (Ben-Or, Goldwasser, Wigderson): work in a finite field $\mathrm{GF}(q)$. Every possible function in $\mathrm{GF}(q)$ is a polynomial
- Required to show how to do multiplication and addition, everything else follows!

MPC by BGW: Basic idea (1/2)

- Work in GF(q), use Shamir's (k, n), $k > n/2$, secret sharing scheme

- Every participant A_j has a share $f_i(j)$, where f_i is the Lagrange-interpolated polynomial with $f_i(0) = s_i$ (the ith secret)

- Given $f_1(j)$ and $f_2(j)$, one can just add the shares: Then participants share the polynomial $f_1 + f_2$ with $(f_1 + f_2)(0) = s_1 + s_2$.

MPC by BGW: Basic idea (2/2)

- Multiplication: if $g = (f_1 \cdot f_2)$ then $g(0) = s_1 \cdot s_2$
- However, g would have degree $\deg f_1 + \deg f_2 = 2k - 2$
- Also, the coefficients of g would not be randomly distributed
- Solution: after every multiplication perform a simple protocol between all authorities that reduces the degree of g and adds uniformly random values to all coefficients of g, except to g_0

MPC by BGW: Summary

- To work correctly, requires that $k > 2/3n$
- Information-theoretically secure multi-party computation of an arbitrary function f (polynomial in $\mathrm{GF}(q)$)
- Addition: local, multiplication: requires communication
- Even some very simple functions f have complex representing polynomials, thus generic MPC is not always very efficient

MPC by BGW: Examples

- Electronic voting:
  - ⋆ Must compute $f(x_1, \ldots, x_n) = \sum_i x_i$ securely. A simple polynomial, can be done efficiently
- Electronic auctions:
  - ⋆ Must compute $f(x_1, \ldots, x_n) = \max(x_1, \ldots, x_n)$ securely. A complex polynomial, cannot be done efficiently
  - ⋆ Current auction schemes are either less efficient, or leak more information, compared to the voting schemes

Yao's Two-Party Protocol

- BGW does not work for two parties (majority must be honest)
- Idea: present f as a Boolean circuit with AND, OR and NOT gates
- "Garble" inputs to the circuit. "Garble" every gate so that no information about intermediate results will be known
- "Ungarble" outputs
- Efficient for functions that have a simple *Boolean* representation, for example $f(x_1, x_2) = x_1 \oplus x_2$ (coin-tossing)

MPC: theoretical limitations

- All functions can be computed securely
- Information-theoretical security: $k > 2/3n$
- Computational security: $k > 1/2n$
- Several conceptually different models (Yao, BGW, ...)
- Efficiency can be improved, but for most of the practical protocols, general MPC is too slow