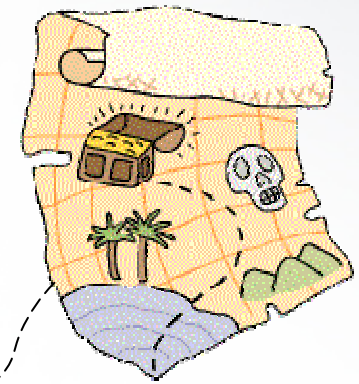
Proactive Secret Sharing
Or:
How to Cope With Perpetual Leakage

Paper by
Amir Herzberg
Stanislaw Jarecki
Hugo Krawczyk
Moti Yung
Presentation by
David Zage

What is Secret Sharing



- Basic Idea ((2, 2)-threshold scheme):
  - Two friends find a map to buried treasure!!
  - Who do we trust?
  - Neither, each gets a “share” of the map
- Based on threshold cryptography

Why Do We Need *Proactive* Sharing?

- Secret sharing is a fundamental tool for protecting sensitive data.
- How do we protect the data from gradual server break-ins?
  - Renew data – Not good for long lived data
  - Renew the Secret – Good!
- Attacker must compromise k+1 servers in a time period instead of the entire life of the system.

System Assumptions

- System
  - (k+1, n)-threshold scheme
  - Secure encryption and signatures exist
  - Synchronized, secure broadcasts over a common medium
  - Data is actually destroyed when erased
- Mobile Adversary
  - Byzantine corruption can occur at any time
  - Adversary can corrupt ***no more*** than k out of n servers, where k < n/2
  - Adversary is connect to the communication medium but cannot interfere with communication

Removing an Adversary from a Server

- Adversaries are removable through reboot procedures
- Honest servers *always* detect and remove misbehaving servers
- DOS attacks on the communication medium not taken into account (further papers address this using asynchronous systems)

Definitions

- Semantically Secure
  - Security is measured in terms of entropy and change
  - The scheme is semantically secure if for any function k computable on the secret, the difference in the probability of learning information between rounds is negligible.
- Robust
  - Scheme guarantees the correct reconstruction of the secret at any time
    - Tolerates up to k Byzantine faults

Cryptographic Tools

- Shamir's Secret Sharing
  - Dealer chooses a function f of degree k over a finite field where f(0) = *secret*
  - Dealer calculates $v_i = f(i)$ and ***secretly*** sends v_i to the server i.
  - The secret can now be reconstructed with k+1 v_i pieces and polynomial interpolation

Cryptographic Tools

- Verifiable Secret Sharing
  - g is an element of a the finite field the equation k was chosen from
  - Values g^{fi} are broadcast to every server before the secret share is broadcast
  - When a server receives a secret share it checks:

$$g_{xi} = (g^{f_0})(g^{f_1})^{i}(g^{f_2})^{i^2}\ldots(g^{f_k})^{i^k}$$

  - If the equation holds, the share is a valid share.

With VSS there is more information to attack?!?

- Information can be learned from each of the g^x
- What can be done:
  - Use a different scheme where extra information is already released
    - ElGamal Signatures
  - Place the secret X in an “envelope”
    - Encode the secret in a longer bit string s

Periodic Share Renewal

- Each server has a pair of public and private keys used for secure communication.
  - Assumption: Attacker **cannot** modify the keys
- System initialization
  - The secret is encoded using Shamir's secret sharing and securely distributed to all servers
  - Time periods for renewal are set arbitrarily by system administrator

Basic Share Renewal Protocol

- Each server P_i picks k random numbers from the finite field and creates a polynomial of degree k

$$\delta_i(z) = \delta_{i1} z^1 + \delta_{i2} z^2 + \ldots + \delta_{ik} z^k$$

- For all other servers Pj, Pi secretly sends out $\delta_i(j)$ to Pj
- Pi computes the new share by:

$$x_i^{(t-1)} \leftarrow x_i^{(t-1)} + (\delta_{1i} + \delta_{2i} + \ldots + \delta_{ni})$$

- Pi updates its share and erases all other data

Basic Share Renewal Protocol(2)

- Solves share renewal in the face of a passive adversary
- If all of the servers follow the protocol, then the share renewal protocol is **correct**, **robust** and is **secret**
  - Each new round produces a valid set of secret shares
  - Any k+1 servers can re-create the secret at any time
  - With k or less shares, no information is learned

Share Renewal Protocol in the Presence of Active Attackers

- Each server Pi picks k random numbers from the finite field and creates a polynomial of degree k

$$\delta_i(z) = \delta_{i1} z^1 + \delta_{i2} z^2 + \ldots + \delta_{ik} z^k$$

  Each Pi also computes $\varepsilon_{im} = g^{\delta_{im}}$ for each k
- Pi computes $\delta_i(j)$ and broadcasts the set of ε's and δ signed with Pi's signature

Share Renewal Protocol in the Presence of Active Attackers(2)

- Pi computes the new share by:

$$x_i^{(t-1)} \leftarrow x_i^{(t-1)} + (\delta_{1i} + \delta_{2i} + \ldots + \delta_{ni})$$

- and checks the validity by computing:

$$g^{\delta_{j(i)}} = (\varepsilon_{j1})^{i} (\varepsilon_{j2})^{i^2} \ldots (\varepsilon_{jk})^{i^k}$$

- If the messages are correct, Pi broadcasts an accept message
- If not Pi broadcasts an accusation against the misbehaving server(s)

Resolving Accusations

- If faulty server is recognized
  - Do not use the polynomial broadcast by the server
  - Reset the server to “expel” the adversary
- Three types of possible faults:
  - Incorrect message format
  - Zero or greater than One correct message from a server
  - ➢ Verifiability equations do not match
- 3rd type of fault requires extra effort to handle

Resolving Accusations(2)

- If Pi accuses Pj of cheating, Pj must “defend” itself
  - If Pj sent a correct $\delta_j(i)$, then it exposes this value and all servers can check with the ε values already published during the protocol.
- If Pj defends itself, then Pi marked as the fault server, else Pj is marked.
- The share renewal equation becomes:

$$x_i^{(t-1)} \leftarrow x_i^{(t-1)} + \sum_{j \notin B_i} \delta_{ji}$$

Share Recovery Scheme

- Severs must make sure other servers have not had their keys compromised. Otherwise an adversary could cause the secret to be lost by destroying n-k keys.
- Without recovery, we loose security
- For practical schemes:
  - During reboot, a server will loose it share and need recovery

Detecting Corruption

- During initialization, each server stores a the set of $y_j^{(t)} = g^{x_j^{(t)}}$ for all the servers current shares.
- During the secret share update, this set of exponents is also updated in a similar fashion.

$$y_j^{(t)} \leftarrow y_j^{(t-1)} + \sum_{a \notin B_i} g^{\delta aj}$$

- If during the update phase, the value of the new x received and the one calculated do not match, the server needs to have its share recovered.

Basic Share Recovery Protocol

For every failed server *r*

- Every valid Pi picks a random k-degree polynomial such that f(r) = 0
- Every Pi broadcasts f(r)
- Each Pi the creates a new share for r,

$$x'_i \leftarrow x_i + \sum_{j \in D} f(i)$$

  and send it to r
- R receives the shares and interpolates them to find its secret share x_r

Share Recovery

- Verifiability can be added using same technique as earlier
  - Used to detect incorrect reconstructions
- Multiple shares can be recovered in parallel by treating each share as its on secret.

Total Protocol for Proactive Secret Sharing

At the beginning of every time period:

- Private Key Renewal protocol
  - Do not have to assume attacker can not tamper with public/private communication keys
- Share Recovery Protocol (including lost shares detection)
- Share Renewal Protocol

Applications

- Proactively share decryption key for sensitive data
- Proactive function sharing built of Proactive secret sharing
- Proactive digital signatures

Summary

- Semantically *secure* and *robust* proactive secret sharing scheme based on threshold cryptography and verifiable secret scheme.
- Takes many aspects of security and builds them into a cohesive unit.