Threshold Cryptography

Cloud Security Mechanisms

Björn Groneberg - Summer Term 2013
Threshold Cryptography

• Sharing Secrets
  – Treasure Map
  – Sharing keys on multiple server

• Threshold Encryption
  – Protect top secret document, only group of people can decrypt it

• Threshold Signature
  – Signing checks

• E-Voting
  – Do not trust only one voting authority
Threshold Cryptography

1. Basic Maths
2. Lagrange Polynomial Interpolation
3. Shamir’s Secret Sharing
4. Elgamal Encryption
5. Threshold Elgamal
6. Threshold RSA
7. E-Voting
Basic Maths

• $p$ is a prime 😊
• modulo operator mod:
  – find remainder of division of two numbers

$$20 : 6 = 18 \ R: 2 \Rightarrow 20 \ mod \ 6 = 2$$

• modulo congruent =
  – two numbers are congruent modulo $m$ if they have the same remainder by the division of $m$

$$20 \ mod \ 6 = 2 \ and \ 14 \ mod \ 6 = 2 \Rightarrow 20 = 14 \ mod \ 6$$
Basic Maths

• Residue class
  – Collect all integers which are congruent given a modulo \( m \)
  – Example: \( \text{mod } 6 \)

\[
[0]_6 = \{..., -6, 0, 6, 12, 18, ... \} \quad [1]_6 = \{..., -5, 1, 7, 13, 19, ... \}
[2]_6 = \{..., -4, 2, 8, 14, 20, ... \} \quad [3]_6 = \{..., -3, 3, 9, 15, 21, ... \}
[4]_6 = \{..., -2, 4, 10, 16, 22, ... \} \quad [5]_6 = \{..., -1, 5, 11, 17, 23 ... \}
\]

• Residue class system (ring) \( \mathbb{Z}_n \)
  – Collect all residue classes and have two operations
  – Example:

\[
\mathbb{Z}_6 = \{[0]_6, [1]_6, [2]_6, [3]_6, [4]_6, [5]_6\} = \{0, 1, 2, 3, 4, 5\}
5 + 4 = 3 \quad 3 + 4 = 1 \quad 9 + 12 = 5 \quad \text{mod } 6
5 \cdot 4 = 2 \quad 3 \cdot 4 = 0 \quad 9 \cdot 12 = 0 \quad \text{mod } 6
\]
Threshold Cryptography

1. Basic Maths
2. Lagrange Polynomial Interpolation
3. Shamir’s Secret Sharing
4. Elgamal Encryption
5. Threshold Elgamal
6. Threshold RSA
7. E-Voting
Lagrange Polynomial Interpolation

• Find polynomial to given set of points

(1, 2), (−2, 2), (2, 1)

\[ f(x) = ? \]
Lagrange Polynomial Interpolation

Interpolate polynomial function out of given points

Given: $k + 1$ data points:

$$(x_0, y_0), \ldots, (x_j, y_j), \ldots, (x_k, y_k)$$

where no two $x_j$ are the same

Lagrange polynomial interpolation is:

$$L(x) := \sum_{j=0}^{k} y_j \ell_j = y_0 \ell_1 + \cdots + y_j \ell_j + \cdots + y_k \ell_k$$

where $\ell_j$ is Lagrange basis polynomials:

$$\ell_j := \prod_{0 \leq m \leq k \atop m \neq j} \frac{x - x_m}{x_j - x_m} = \frac{x - x_0}{x_j - x_0} \cdots \frac{x - x_{j-1}}{x_j - x_{j-1}} \frac{x - x_{j+1}}{x_j - x_{j+1}} \cdots \frac{x - x_k}{x_j - x_k}$$

[La13]
Lagrange Example

- Given Points: $(1, 2), (-2, 2), (2, 1)$ \[ k = 2 \]

- Calculate Lagrange basis polynomials

\[
\ell_0 := \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} = \frac{(x + 2)(x - 2)}{(1 + 2)(1 - 2)} = -\frac{1}{3}(x^2 - 4)
\]

\[
\ell_1 := \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_0 - x_2)} = \frac{(x - 1)(x - 2)}{(-2 - 1)(-2 - 2)} = \frac{1}{12}(x^2 - 3x + 2)
\]

\[
\ell_2 := \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} = \frac{(x - 1)(x + 2)}{(2 - 1)(2 + 2)} = \frac{1}{4}(x^2 + x - 2)
\]

- Calculate Lagrange polynomial:

\[
L(x) = y_0 \ell_0 + y_1 \ell_1 + y_2 \ell_2
\]

\[
L(x) = 2 \cdot -\frac{1}{3}(x^2 - 4) + 2 \cdot \frac{1}{12}(x^2 - 3x + 2) + 1 \cdot \frac{1}{4}(x^2 + x - 2) = -\frac{1}{4}x^2 - \frac{1}{4}x + \frac{5}{2}
\]

[La13]
Lagrange Polynomial Interpolation

• Find polynom to given set of points

\[(1, 2), (-2, 2), (2, 1)\]

\[f(x) = -\frac{1}{4}x^2 - \frac{1}{4}x + \frac{5}{2}\]
Threshold Cryptography

1. Basic Maths
2. Lagrange Polynomial Interpolation
3. Shamir’s Secret Sharing
4. Elgamal Encryption
5. Threshold Elgamal
6. Threshold RSA
7. E-Voting
Secret Sharing

• How to distribute secret $s$ to $n$ parties in that way, that
  – Only all $n$ parties together or
  – $k$ out of $n$ parties
can recompute the secret?
Secret Sharing

- Recomputation of the secret
  - all $n$ out of $n$ parties: $(n, n)$ threshold
  - $n - 1$, $n - 2$, ... parties should not be able to recompute the secret
  - Every party (or group of parties) should not be able to retrieve any information about the global secret from their own secret(s)
Secret Sharing

• Recomputation of the secret
  – $k$ out of $n$ parties: $(k, n)$ threshold

  – $k - 1, k - 2, \ldots$ parties should not be able to recompute the secret
  – Every party (or group of parties) should not be able to retrieve any information about the global secret from their own secret(s)
Secret Sharing

• Real world’s solution:
  – Multiple locks with keys → heavy key ring

• Naive solution (bad):
  – Split secret in parts:
    
    1873 7632 8732 3253 2312

    1873 7632 8732 3253 2312

  – Disadvantage:
    • needs \((n, n)\) threshold
    • \(n - 1\) out of \(n\) parties dramatically reduce possible keys
Shamir’s Secret Sharing

- Published 1979 by Adi Shamir
- \((k, n)\) threshold sharing
- Based on Lagrange polynomials

**Dealing Algorithm:**
- Given: \((k, n)\) threshold and secret \(s \in \mathbb{Z}_q\)
- Randomly choose \(k - 1\) coefficients \(a_1, \ldots, a_{k-1}\)
- Set \(a_0 := s\)
- Build polynomial \(f(x) = a_0 + a_1 x + a_2 x^2 + a_{k-1} x^{k-1}\)
- Set \(i = 1, \ldots, n\) and calculate Points \(s_i = (i, f(i)) \mod q\)
- Every party gets (at least) one point \(s_i\)
Shamir’s Secret Sharing - Example

- Dealing Algorithm

**Given:** \((k, n)\) and secret \(s \in \mathbb{Z}_q\)  

\((3, 5)\) threshold \(s = 6 \in \mathbb{Z}_{22}\)

Randomly \(k - 1: a_1, ..., a_{k-1}\)

Set \(a_0 := s\)

\(f(x) = a_0 + a_1 x + a_2 x^2 + a_{k-1} x^{k-1}\)

\(i = 1, ..., n\) calculate

\(s_i = (i, f(i)) \mod q\)

\(s_1 = (1, 9)\)

\(s_2 = (2, 14)\)

\(s_3 = (3, 21)\)

\(s_4 = (4, 8)\)

\(s_5 = (5, 19)\)
Shamir’s Secret Sharing

• Recomputation
  – Given: \( k \) Points \( s_i = (x_i, y_i) \)
  – Goal: find \( f(x) = a_0 + a_1 x + a_2 x^2 + a_{k-1} x^{k-1} \)
    with \( f(0) = a_0 \) as the secret
  – Using \( f(x) = L(x) \),
    \( S \subseteq \{1, \ldots, n\}, |S| = k \) and calculate
    \[
    f(0) = L(0) = \sum_{j \in S} y_j \ell_{j,0,S} \mod q
    \]
    with \( \ell_{j,0} \) as Lagrange basis polynomials with \( x = 0 \) and \( S \):
    \[
    \ell_{j,0,S} := \prod_{\substack{m \in S \setminus \{j\} \atop m \neq j}} \frac{-x_m}{x_j - x_m} \mod q
    \]

Lagrange:
\[
L(x) := \sum_{j=0}^{k} y_j \ell_j
\]
\[
\ell_j := \prod_{\substack{0 \leq m \leq k \atop m \neq j}} \frac{x - x_m}{x_j - x_m}
\]

[Sha79]
Shamir’s Secret Sharing - Example

- Recomputation of basis polynomials:

\[
l_{2,0,\{2,4,5\}} = \frac{-x_4}{(x_2 - x_4)} \frac{-x_5}{(x_2 - x_5)} = \frac{-4}{(2 - 4)} \frac{-5}{(2 - 5)} = 10 \cdot 3^{-1} = 10 \cdot 15 = 18 \mod 22
\]

\[
l_{4,0,\{2,4,5\}} = \frac{-x_2}{(x_4 - x_2)} \frac{-x_5}{(x_4 - x_5)} = \frac{-2}{(4 - 2)} \frac{-5}{(4 - 5)} = -5 = 17 \mod 22
\]

\[
l_{5,0,\{2,4,5\}} = \frac{-x_2}{(x_5 - x_2)} \frac{-x_4}{(x_5 - x_4)} = \frac{-2}{(5 - 2)} \frac{-4}{(5 - 4)} = 8 \cdot 3^{-1} = 8 \cdot 15 = 10 \mod 22
\]

\[s_4 = (4, 8)\]
\[s_5 = (5, 19)\]

Bob
Chris

Felix
George

Trusted dealer
Dave

„Shamir’s Lagrange“:

\[
L(0) = \sum_{j \in S} y_j l_{j,0,S}
\]

\[
l_{j,0,S} := \prod_{m \in S} \frac{-x_m}{x_j - x_m}
\]
Shamir’s Secret Sharing - Example

- Recomputation:
  \[ \ell_{2,0,\{2,4,5\}} = 18, \quad \ell_{4,0,\{2,4,5\}} = 17, \quad \ell_{5,0,\{2,4,5\}} = 10 \]

  \[ s = L(0) = y_2 \cdot \ell_{2,0,\{2,4,5\}} + y_4 \cdot \ell_{4,0,\{2,4,5\}} + y_5 \cdot \ell_{5,0,\{2,4,5\}} \]
  \[ s = L(0) = 14 \cdot 18 + 8 \cdot 17 + 19 \cdot 10 \mod 22 \]
  \[ s = 6 \]

\[ s_4 = (4, 8) \]
\[ s_5 = (5, 19) \]

"Shamir’s Lagrange":

\[ L(0) = \sum_{j \in S} y_j \ell_{j,0,S} \]

\[ \ell_{j,0,S} := \prod_{m \in S \atop m \neq j} \frac{-x_m}{x_j - x_m} \]
Shamir’s Secret Sharing - Remarks

• Graphical Interpretation

• Flexibility
  – Increase $n$ and compute new shares without affecting other shares
  – Removing existing shares (shares have to be destroyed)
  – Replace shares without changing the secret: new polynomial $f^*(x)$
  – One party can have more than one share

[Li04]
Threshold Cryptography

1. Basic Maths
2. Lagrange Polynomial Interpolation
3. Shamir’s Secret Sharing
4. Elgamal Encryption
5. Threshold Elgamal
6. Threshold RSA
7. E-Voting
Elgamal Encryption

- Published 1985 by Taher Elgamal
- Based on Diffie-Hellman key exchange

- Public / private key encryption:
  - Generation: pub, priv
  - Encryption: cipher = $\text{enc}_{\text{pub}}(m)$
  - Decryption: $m = \text{dec}_{\text{priv}}(\text{cipher})$
Elgamal Encryption - Example

• Public / private key generation

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>large prime $p$ with generator $g$</td>
<td>$p = 23$, $g = 5$</td>
</tr>
<tr>
<td>2.</td>
<td>randomly $a \in {1, ..., p - 1}$</td>
<td>$a = 6$</td>
</tr>
<tr>
<td>3.</td>
<td>Calculate $A = g^a \mod p$</td>
<td>$A = 5^6 = 8 \mod 23$</td>
</tr>
<tr>
<td>4.</td>
<td>pub = $(p, g, A)$ priv = $a$</td>
<td>pub = $(23, 5, 8)$ priv = 6</td>
</tr>
</tbody>
</table>

Alice

priv = 6

pub = (23,5,8)

Bob

pub = (23,5,8)

Alice

[El85]
Elgamal Encryption - Example

- **Encryption**

<table>
<thead>
<tr>
<th>Given: message  ( m \in {0, \ldots, p - 1} )</th>
<th>( m = 12 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Randomly  ( b \in {1, \ldots, 1 - p} )</td>
<td>( b = 3 )</td>
</tr>
<tr>
<td>Calculate  ( B = g^b \mod p )</td>
<td>( B = 5^3 = 10 \mod 23 )</td>
</tr>
<tr>
<td>( c = A^b m \mod p )</td>
<td>( c = 8^3 \cdot 12 = 3 \mod 23 )</td>
</tr>
<tr>
<td>Cipher text is cipher = ((B, c))</td>
<td>cipher = ((10, 3))</td>
</tr>
</tbody>
</table>

From: Bob
To: Alice
cipher = \((10, 3)\)

Alice

Bob

Alice

From: Bob
To: Alice
\( m = 12 \)

From: Bob
To: Alice
pub = \((23, 5, 8)\)
Elgamal Encryption - Example

• Decryption

<table>
<thead>
<tr>
<th><strong>Given:</strong> cypher = (B, c) and priv = a</th>
<th>cypher = (10,3) priv = 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calculate ( x = p - 1 - a )</td>
<td>( x = 23 - 1 - 6 = 16 )</td>
</tr>
<tr>
<td>Calculate ( m = B^x c \mod p )</td>
<td>( m = 10^{16} \cdot 3 = 12 \mod 23 )</td>
</tr>
<tr>
<td>Encrypted message ( m )</td>
<td>( m = 12 )</td>
</tr>
</tbody>
</table>

• General Idea: \( m = (B^a)^{-1} \cdot c = B^{(p-1-a)} \mod p \)

From: Bob To: Alice
\( m = 12 \)

From: Bob To: Alice
cypher = (10, 3)

Alice
pub = (23,5,8)

Alice

Bob

[El85]
Threshold Cryptography

1. Basic Maths
2. Lagrange Polynomial Interpolation
3. Shamir’s Secret Sharing
4. Elgamal Encryption
5. **Threshold Elgamal**
6. Threshold RSA
7. E-Voting
Threshold Elgamal

• Using Elgamal encryption scheme in a threshold environment
• Generation:
  – Generate \( \text{pub} = (p, g, A) \) \( \text{priv} = a \) like normal Elgamal encryption
  – Share \( \text{priv} = a \) among \( n \) parties, using Shamir’s secret sharing with \( q = \varphi(p) = p - 1 \)
  – Every party \( j \) gets (at least) one point \( s_j = (x_j, y_j) \)

Example: \( \text{pub} = (23, 5, 8) \) \( \text{priv} = 6 \) (3,5)-threshold

\[ \begin{align*}
    s_1 &= (1, 9) \\
    s_2 &= (2, 14) \\
    s_3 &= (3, 21) \\
    s_4 &= (4, 8) \\
    s_5 &= (5, 19)
\end{align*} \]

[Ca06]
Threshold Elgamal

• Encryption
  – Normal Elgamal encryption with message $m$ and pub = $(p, g, A)$

From: Alice
To: BCDFG
pub = (23,5,8)

BCDFG cipher = (10, 3)

[Ca06]
Threshold Elgamal

- **Decryption**
  - Trusted dealer and every party can receive cipher = \((B, c)\)
  - at least \(k\) parties have to compute decryption share \(d_j = B^{y_j} \mod p\)
  - Trusted dealer can compute \(m\) with set \(S\) of \(j \in \{1, \ldots, n\}\) which returned their \(d_j\)
  - Party:
    \(d_j = B^{y_j} \mod p\)
  - Trusted Dealer:
    \(m = \left(\prod_{j \in S} d_j^{\ell_{j,0,s}}\right)^{-1} \cdot c \mod p\)

[Ca06]
Threshold Elgamal - Example

- **Decryption**
  - Every party computes decryption share:
    \[ d_2 = B^{y_2} = 10^{14} = 12 \mod 23 \]
    \[ d_4 = B^{y_5} = 10^8 = 2 \mod 23 \]
    \[ d_5 = B^{y_5} = 10^{19} = 21 \mod 23 \]
  - Trusted dealer computes \( \ell_{j,0,S} \):
    \[ \ell_{2,0,\{2,4,5\}} = 18 \]
    \[ \ell_{4,0,\{2,4,5\}} = 17 \]
    \[ \ell_{5,0,\{2,4,5\}} = 10 \]

  → Shamir’s secret sharing, slide 20

\[ \ell_{j,0,S} := \prod_{\substack{m \in S \setminus\{j\} \\cap S\neq\{j\}}} \frac{-x_m}{x_j - x_m} \]

Threshold Elgamal cipher = \((B, c)\)

\[ d_j = B^{y_j} \mod p \]

\[ m = \left( \prod_{j \in S} d_j^{\ell_{j,0,S}} \right)^{-1} \cdot c \mod p \]

From: Alice
To: BCDFG
cipher = \((10, 3)\)
Threshold Elgamal - Example

• Decryption

\[ d_2 = 12, \quad d_4 = 2, \quad d_5 = 21 \]
\[ \ell_{2,0,\{2,4,5\}} = 18, \quad \ell_{4,0,\{2,4,5\}} = 17, \quad \ell_{5,0,\{2,4,5\}} = 10 \]

– Trusted dealer computes \( m \):

\[ m = \left( d_2 \ell_{2,0,\{2,4,5\}} \cdot d_4 \ell_{4,0,\{2,4,5\}} \cdot d_5 \ell_{5,0,\{2,4,5\}} \right)^{-1} \cdot c \mod p \]
\[ m = (12^{18} \cdot 2^{17} \cdot 21^{10})^{-1} \cdot 3 \mod 23 \]
\[ m = (6)^{-1} \cdot 3 \mod 23 \]
\[ m = 4 \cdot 3 \mod 23 \]
\[ m = 12 \]

Note: \( (6)^{-1} = 4 \mod 23 \)
(Extended Euclidean algorithm)
Threshold Cryptography

1. Basic Maths
2. Lagrange Polynomial Interpolation
3. Shamir‘s Secret Sharing
4. Elgamal Encryption
5. Threshold Elgamal
6. **Threshold RSA**
7. E-Voting
**RSA Threshold Signatures**

- **Signatures**
  - Requires: Public / private key and hash function $H(x)$
  - Sign a message:
    - Hash message $m$ and encrypt with private key: $\text{sign} = \text{enc}_{\text{priv}}(H(m))$
  - Verify signature
    - Decrypt signature with public key and check hash: $\text{dec}_{\text{pub}}(\text{sign}) = H(m)$

From: Bob
To: Alice
$m = \ldots$

Bob

priv = \ldots

Alice

sign from: Bob

Bob

pub = \ldots

Alice

sign from: Bob

Bob

pub = \ldots

[Ca06]

09.07.2013 Threshold Cryptography
Every party signs with own private key

Trusted dealer can compute global signature

\[ \text{Party } i: \quad \text{sign}_i = \text{enc}_{\text{priv}_i}(H(m)) \]

\[ \text{Trusted dealer:} \quad \text{sign} = \text{collect(}\text{sign}_1, \ldots, \text{sign}_n) \]

V. Shoup: “Practical threshold signatures” shows threshold signature scheme with RSA [Sh]
Threshold Cryptography

1. Basic Maths
2. Lagrange Polynomial Interpolation
3. Shamir’s Secret Sharing
4. Elgamal Encryption
5. Threshold Elgamal
6. Threshold RSA
7. E-Voting
E-Voting

- Secret voting using Elgamal threshold encryption
- Voter encrypts vote with public key
- Private key is shared among voting authorities

09.07.2013
Threshold Cryptography
E-Voting

- Voting authorities “counting” encrypted votes
- Decrypt result of “counting” with shared secrets

From: Bob
\[ v_{Bob} = (B, c) \]

From: Alice
\[ v_{Alice} = (B, c) \]

From: Chris
\[ v_{Chris} = (B, c) \]

\[ v_{Result} = \text{count}(v_{Bob}, v_{Alice}, v_{Chris}) \]
\[ v_{Result} = (B, c) \]

\[ \text{Result} = \text{dec}_{\text{priv}_1}(v_{Result}) \]
\[ \text{Result} = -1 \]

- Cramer, et. al.: "A secure and optimally efficient multi-authority election scheme." [Cr97]
Summary Threshold Cryptography

- Sharing Secrets
- Threshold Encryption
- Threshold Signatures
- E-Voting

- General Problem: Trusted Dealer
- Secret sharing schemes without trusted dealer
References


[Li04] T-79.159 Cryptography and Data Security, 24.03.2004 Lecture 9: Secret Sharing, Threshold Cryptography, MPC, Helger Lipmaa

[Ca06] Security and Fault-tolerance in Distributed Systems, Winter 2006/07, 7 Distributed Cryptography, Christian Cachin, IBM Zurich Research Lab