Secret Sharing Schemes and their Applications

Giorgio Zanin

Dipartimento di Informatica
Università degli Studi di Roma “La Sapienza”

S.M.A.R.T. periodic meetings

Outline of Part I

Outline of Part I

Outline of Part I

Outline of Part II

Part I

Classical Secret Sharing Schemes

Historical Problem

Liu in [1] considers the following:

Problem

Eleven scientists are working on a secret project. They wish to lock up the documents in a cabinet so that the cabinet can be opened if and only if six or more of the scientists are present. What is the smallest number of locks needed? What is the smallest number of keys to the locks each scientist must carry?

Answers

- 462 locks
- 252 keys per scientist.

Historical Problem

Liu in [1] considers the following:

Problem

Eleven scientists are working on a secret project. They wish to lock up the documents in a cabinet so that the cabinet can be opened if and only if six or more of the scientists are present. What is the smallest number of locks needed? What is the smallest number of keys to the locks each scientist must carry?

Answers

- 462 locks
- 252 keys per scientist.

Secret Sharing
Adi Shamir & George Blakley - 1979

Informally

Any method for distributing a secret amongst a group of individuals (shareholders) each of which is allocated some information (share) related to the secret

- The secret can only be reconstructed when the shares are combined together
- Individual shares are of no use on their own

Secret Sharing
Adi Shamir & George Blakley - 1979

Informally

Any method for distributing a secret amongst a group of individuals (shareholders) each of which is allocated some information (share) related to the secret

- The secret can only be reconstructed when the shares are combined together
- Individual shares are of no use on their own

Secret Sharing
Adi Shamir & George Blakley - 1979

Informally

Any method for distributing a secret amongst a group of individuals (shareholders) each of which is allocated some information (share) related to the secret

- The secret can only be reconstructed when the shares are combined together
- Individual shares are of no use on their own

(t,n)-threshold schemes

Goal

To divide a secret S into n shares $s_0, \dots, s_{n-1}$ such that:

- knowledge of t or more shares makes S easily computable
- knowledge of $t-1$ or less shares leaves S completely undetermined

(t,n)-threshold schemes

Goal

To divide a secret S into n shares $s_0, \dots, s_{n-1}$ such that:

- knowledge of t or more shares makes S easily computable
- knowledge of $t - 1$ or less shares leaves S completely undetermined

(t,n)-threshold schemes

Goal

To divide a secret S into n shares $s_0, \ldots, s_{n-1}$ such that:

- knowledge of t or more shares makes S easily computable
- knowledge of $t-1$ or less shares leaves S completely undetermined

(t,n)-threshold schemes

Goal

To divide a secret S into n shares $s_0, \ldots, s_{n-1}$ such that:

- knowledge of t or more shares makes S easily computable
- knowledge of $t-1$ or less shares leaves S completely undetermined

A Flawed Secret Sharing Scheme

(4,4)-threshold scheme (?)

- S="giorgiozanin"
- shares:

$$s_0 = gio - - - - - - - - -$$
$$s_1 = - - -rgi - - - - - -$$
$$s_2 = - - - - - - oza - --$$
$$s_3 = - - - - - - - - -nin.$$

A Flawed Secret Sharing Scheme

shares	missing	possible values
0	12	26^{12}=95428956661682176
1	9	26^{9}=5429503678976
2	6	26^{6}=308915776
3	3	26^{3}=17576

- knowledge of less than t shares gains information about S
- desirable: even with $t-1$ shares, still 26^{12} possible values

A Flawed Secret Sharing Scheme

shares	missing	possible values
0	12	26^{12}=95428956661682176
1	9	26^{9}=5429503678976
2	6	26^{6}=308915776
3	3	26^{3}=17576

- knowledge of less than t shares gains information about S
- desirable: even with $t-1$ shares, still 26^{12} possible values

Trivial Secret Sharing Schemes

(n,n)

- Encode the secret as an *integer* S. Give to each shareholder i (except one) a random integer r_i. Give to the last shareholder the number $(S - r_1 - r_2 - \ldots - r_{n-1})$. The secret is the sum of the shareholders' shares.
- Encode the secret as a *byte* S. Give to each shareholder i (except one) a random byte b_i. Give to the last shareholder the byte $(S \bigotimes b_1 \bigotimes b_2 \bigotimes \ldots \bigotimes b_{n-1})$ where $\bigotimes$ is bitwise XOR. The secret is the bitwise XOR of the shareholders' shares.

(t,n),$t \leq n$

- Apply n instances of a (t, t) protocol as above and distribute t shares to each individual

Trivial Secret Sharing Schemes

(n,n)

- Encode the secret as an *integer* S. Give to each shareholder i (except one) a random integer r_i. Give to the last shareholder the number $(S - r_1 - r_2 - \ldots - r_{n-1})$. The secret is the sum of the shareholders' shares.
- Encode the secret as a *byte* S. Give to each shareholder i (except one) a random byte b_i. Give to the last shareholder the byte $(S \bigotimes b_1 \bigotimes b_2 \bigotimes \ldots \bigotimes b_{n-1})$ where $\bigotimes$ is bitwise XOR. The secret is the bitwise XOR of the shareholders' shares.

(t,n),$t \leq n$

- Apply n instances of a (t, t) protocol as above and distribute t shares to each individual

Trivial Secret Sharing Schemes

(n,n)

- Encode the secret as an *integer* S. Give to each shareholder i (except one) a random integer r_i. Give to the last shareholder the number $(S - r_1 - r_2 - \ldots - r_{n-1})$. The secret is the sum of the shareholders' shares.
- Encode the secret as a *byte* S. Give to each shareholder i (except one) a random byte b_i. Give to the last shareholder the byte $(S \bigotimes b_1 \bigotimes b_2 \bigotimes \ldots \bigotimes b_{n-1})$ where $\bigotimes$ is bitwise XOR. The secret is the bitwise XOR of the shareholders' shares.

(t,n),$t \leq n$

- Apply n instances of a (t, t) protocol as above and distribute t shares to each individual

Which choice of t and n

Contexts of application

Suitable in applications in which a group of mutually suspicious individuals with conflicting interests must cooperate.

- Sufficiently large majority: take action
- Sufficiently large minority: block action

Tradeoffs

- Secrecy vs Availability

  Management of cryptographic keys

- Safety vs Ease of use

  Digitally Signed checks

Which choice of t and n

Contexts of application

Suitable in applications in which a group of mutually suspicious individuals with conflicting interests must cooperate.

- Sufficiently large majority: take action
- Sufficiently large minority: block action

Tradeoffs

- Secrecy vs Availability

  Management of cryptographic keys

- Safety vs Ease of use

  Digitally Signed checks

Which choice of t and n

Contexts of application

Suitable in applications in which a group of mutually suspicious individuals with conflicting interests must cooperate.

- Sufficiently large majority: take action
- Sufficiently large minority: block action

Tradeoffs

- Secrecy vs Availability

  Management of cryptographic keys

- Safety vs Ease of use

  Digitally Signed checks

Security

Secrecy and Integrity

- Secrecy: the adversary needs to corrupt at least t shareholders and collect their shares in order to learn the secret;
- Integrity: the adversary needs to corrupt at least $n - t + 1$ shareholders to destroy or alter the secret;

Availability

For a given t, the secret Availability increases as n increases...
...and, for a given n the secret's Secrecy and Integrity increase as t increases.

Security

Secrecy and Integrity

- Secrecy: the adversary needs to corrupt at least t shareholders and collect their shares in order to learn the secret;
- Integrity: the adversary needs to corrupt at least $n - t + 1$ shareholders to destroy or alter the secret;

Availability

For a given t, the secret Availability increases as n increases...

...and, for a given n the secret's Secrecy and Integrity increase as t increases.

Security

Secrecy and Integrity

- Secrecy: the adversary needs to corrupt at least t shareholders and collect their shares in order to learn the secret;
- Integrity: the adversary needs to corrupt at least $n - t + 1$ shareholders to destroy or alter the secret;

Availability

For a given t, the secret Availability increases as n increases...

...and, for a given n the secret's Secrecy and Integrity increase as t increases.

Security

Secrecy and Integrity

- Secrecy: the adversary needs to corrupt at least t shareholders and collect their shares in order to learn the secret;
- Integrity: the adversary needs to corrupt at least $n - t + 1$ shareholders to destroy or alter the secret;

Availability

For a given t, the secret Availability increases as n increases...
...and, for a given n the secret's Secrecy and Integrity increase as t increases.

Interpolation

Theorem

Given a function f and t points x_i, $0 \leq i \leq t-1$, there exists a unique polynomial π of degree $t-1$ that interpolates f in x_i:

$$\pi \in \Pi_{t-1}$$

and

$$\pi(x_i) = f_i, \forall i = 0, \ldots, t-1$$

where $f_i = f(x_i)$.

Observations

- $\forall \pi \in \Pi_{t-1}, \pi = \sum_{i=0}^{t-1} a_i x^i$
- x^i are linearly independent

To determine a polynomial in Π_{t-1}, t conditions are necessary

Interpolation

Theorem

Given a function f and t points x_i, $0 \leq i \leq t-1$, there exists a unique polynomial π of degree $t-1$ that interpolates f in x_i:

$$\pi \in \Pi_{t-1}$$

and

$$\pi(x_i) = f_i, \forall i = 0, \ldots, t-1$$

where $f_i = f(x_i)$.

Observations

- $\forall \pi \in \Pi_{t-1}, \pi = \sum_{i=0}^{t-1} a_i x^i$
- x^i are linearly independent

To determine a polynomial in Π_{t-1}, t conditions are necessary

Interpolation

Theorem

Given a function f and t points $x_i,\ 0 \leq i \leq t-1$, there exists a unique polynomial π of degree $t-1$ that interpolates f in x_i:

$$\pi \in \Pi_{t-1}$$

and

$$\pi(x_i) = f_i, \forall i = 0, \dots, t-1$$

where $f_i = f(x_i)$.

Observations

- $\forall \pi \in \Pi_{t-1}, \pi = \sum_{i=0}^{t-1} a_i x^i$
- x^i are linearly independent

To determine a polynomial in Π_{t-1}, t conditions are necessary

Interpolation

In order to find the coefficients of polynomial π, solve:

$$\begin{pmatrix} x_0^0 & \dots & x_0^{t-1} \\ x_1^0 & \dots & x_1^{t-1} \\ \vdots & \vdots & \vdots \\ x_{t-1}^0 & \dots & x_{t-1}^{t-1} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{t-1} \end{pmatrix} = \begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_{t-1} \end{pmatrix}$$

The Vandermonde matrix is not singular, then the interpolating polynomial is unique.

Lagrangian Interpolation

Lagrangian base

$$L_i(x) = \prod_{j=0, j\neq i}^{t-1} \frac{x - x_j}{x_i - x_j}$$

with $x_i \neq x_j$ for $i \neq j$.

Observations

- polynomials $L_i(x),\ x = 0, \ldots, t-1$ are exactly t (they are a base for Π_{t-1}).
- $\prod_{j=0, j\neq i}^{t-1} x_i - x_j$ is a constant.

Lagrangian Interpolation

$$\begin{cases} L_i(x_j) = 0, & j \neq i \\ L_i(x_i) = 1 \end{cases}$$

Coefficients β_i of the interpolating polynomial are such that:

$$\pi(x) = \sum_{i=0}^{t-1} \beta_i L_i(x)$$

with $\pi(x_k) = f_k,\ k = 0, \dots, n$
but

$$\pi(x_k) = \sum_{i=0}^{t-1} \beta_i L_i(x_k) = \beta_k L_k(x_k) = \beta_k = f_k$$

Hence:

$$\pi(x) = \sum_{i=0}^{t-1} f_i L_i(x)$$

Shamir's Secret Sharing

Properties

- Information theoretically secure: less than t shares of the secret provide no information about the secret.
- Makes use of (Lagrangian) interpolation
- Space-efficient: each share has the same size as the original secret

Shamir's Secret Sharing
(t,n)-threshold scheme, [2]

Setup Phase

The Dealer:

1. chooses a large prime q
2. selects a polynomial $\pi \in \Pi_{t-1}$ over Z_q^* such that $\pi(0) \equiv S \pmod{q}$
3. computes $s_i \equiv \pi(i) \pmod{q}$, $i = 1, \ldots, n$.
4. distributes s_i to the shareholders D_i, $i = 1, \cdots, n$

Reconstruction Phase

Any group Γ of t shareholders

- compute $\pi(0) \equiv \sum_{i \in \Gamma} s_i L_i(0) \pmod{q}$

Note that $L_i(0) \equiv \prod_{j \in \Gamma, j \neq i} \frac{j}{j-i} \pmod{q}$ are nonsecret constants and can be precomputed.

Shamir's Secret Sharing

(t,n)-threshold scheme, [2]

Setup Phase

The Dealer:

1. chooses a large prime q
2. selects a polynomial $\pi \in \Pi_{t-1}$ over Z_q^* such that $\pi(0) \equiv S \pmod{q}$
3. computes $s_i \equiv \pi(i) \pmod{q}$, $i = 1, \ldots, n$.
4. distributes s_i to the shareholders D_i, $i = 1, \cdots, n$

Reconstruction Phase

Any group Γ of t shareholders

- compute $\pi(0) \equiv \sum_{i \in \Gamma} s_i L_i(0) \pmod{q}$

Note that $L_i(0) \equiv \prod_{j \in \Gamma, j \neq i} \frac{j}{j-i} \pmod{q}$ are nonsecret constants and can be precomputed.

Shamir's Secret Sharing

Observations

- The size of each share does not exceed the size of the secret
- Keeping t fixed, shares can be easily added or removed, without affecting other shares
- It is easy to change the shares, keeping the same secret
- It is possible to provide more than one share per individual: hierarchy

Shamir's Secret Sharing
Observations

- The size of each share does not exceed the size of the secret
- Keeping t fixed, shares can be easily added or removed, without affecting other shares
- It is easy to change the shares, keeping the same secret
- It is possible to provide more than one share per individual: hierarchy

Shamir's Secret Sharing
Observations

- The size of each share does not exceed the size of the secret
- Keeping t fixed, shares can be easily added or removed, without affecting other shares
- It is easy to change the shares, keeping the same secret
- It is possible to provide more than one share per individual: hierarchy

Shamir's Secret Sharing
Observations

- The size of each share does not exceed the size of the secret
- Keeping t fixed, shares can be easily added or removed, without affecting other shares
- It is easy to change the shares, keeping the same secret
- It is possible to provide more than one share per individual: hierarchy

Joint Secret Sharing

Setup Phase

- n individuals $D_i,\ i = 1, \ldots, n$ agree on a certain large prime q
- each D_i of them:
  1. randomly selects a polynomial $\pi_i \in \Pi_{t-1}$ over Z_q^*, such that $\pi_i(0) \equiv S_i \pmod q$.
  2. computes $s_i^j \equiv \pi_i(j) \pmod q$, $j = 1, \ldots, n$
  3. securely sends these partial shares to the other D_j's (keeping s_i^i for itself).

Each D_j: $s_j \equiv \sum_{i=1}^{n} s_i^j \pmod q$

Reconstruction Phase

- Secret shared by the n shareholders: $S \equiv \sum_{i=1}^{n} S_i \pmod q$

Joint Secret Sharing

Setup Phase

- n individuals $D_i,\ i = 1, \ldots, n$ agree on a certain large prime q
- each D_i of them:
  1. randomly selects a polynomial $\pi_i \in \Pi_{t-1}$ over Z_q^*, such that $\pi_i(0) \equiv S_i \pmod q$.
  2. computes $s_i^j \equiv \pi_i(j) \pmod q$, $j = 1, \ldots, n$
  3. securely sends these partial shares to the other D_j's (keeping s_i^i for itself).

Each D_j: $s_j \equiv \sum_{i=1}^{n} s_i^j \pmod q$

Reconstruction Phase

- Secret shared by the n shareholders: $S \equiv \sum_{i=1}^{n} S_i \pmod q$

Joint Secret Sharing

Setup Phase

- n individuals $D_i,\ i = 1, \ldots, n$ agree on a certain large prime q
- each D_i of them:
  1. randomly selects a polynomial $\pi_i \in \Pi_{t-1}$ over Z_q^*, such that $\pi_i(0) \equiv S_i \pmod q$.
  2. computes $s_i^j \equiv \pi_i(j) \pmod q$, $j = 1, \ldots, n$
  3. securely sends these partial shares to the other D_j's (keeping s_i^i for itself).

Each D_j: $s_j \equiv \sum_{i=1}^n s_i^j \pmod q$

Reconstruction Phase

- Secret shared by the n shareholders: $S \equiv \sum_{i=1}^n S_i \pmod q$

The Adversary

Who it is

- A malicious individual that misbehaves in different manners, in order to compromise the secret.
- More malicious individuals can conspire

What it does

It may be able to:

- disclose the secret
- destroy/alter the secret
- be admitted as a recognized shareholder i.e. acquire/steal a share;
- cheat while being a shareholder, by using incorrect shares.

The Adversary

Who it is

- A malicious individual that misbehaves in different manners, in order to compromise the secret.
- More malicious individuals can conspire

What it does

It may be able to:

- disclose the secret
- destroy/alter the secret
- be admitted as a recognized shareholder i.e. acquire/steal a share;
- cheat while being a shareholder, by using incorrect shares.

The Adversary

- In [3], two adversary models are presented:

Models

Long-term constrained adversary: can corrupt at most t shareholders during an entire life

Short-term constrained adversary can corrupt at most t shareholders during any period of life.

- Victims can be arbitrarily chosen
- t is a fixed robustness parameter of the scheme

The Adversary

- In [3], two adversary models are presented:

Models

Long-term constrained adversary: can corrupt at most t shareholders during an entire life

Short-term constrained adversary can corrupt at most t shareholders during any period of life.

- Victims can be arbitrarily chosen
- t is a fixed robustness parameter of the scheme

The Adversary

- In [3], two adversary models are presented:

Models

Long-term constrained adversary: can corrupt at most t shareholders during an entire life

Short-term constrained adversary can corrupt at most t shareholders during any period of life.

- Victims can be arbitrarily chosen
- t is a fixed robustness parameter of the scheme

Proactivity [4]

Considerations

Refreshing a secret S could be inefficient, BUT...

...periodically refreshing the single shares would *dramatically* decrease the corruption time window

Proactive Secret Sharing Schemes

- Shares for a secret S are provided (Shamir)
- Periodically renew the shares, without changing the secret S
- Any information learned gets obsolete after the shares' update

Proactivity [4]

Considerations

Refreshing a secret S could be inefficient, BUT...

...periodically refreshing the single shares would *dramatically* decrease the corruption time window

Proactive Secret Sharing Schemes

- Shares for a secret S are provided (Shamir)
- Periodically renew the shares, without changing the secret S
- Any information learned gets obsolete after the shares' update

Proactivity
[4]

Considerations

Refreshing a secret S could be inefficient, BUT...

...periodically refreshing the single shares would *dramatically* decrease the corruption time window

Proactive Secret Sharing Schemes

- Shares for a secret S are provided (Shamir)
- Periodically renew the shares, without changing the secret S
- Any information learned gets obsolete after the shares' update

Proactivity

The Scheme

Each shareholder D_i:

1. picks at random $\pi_i \in \Pi_{t-1}$ such that $\pi_i(0) \equiv 0 \pmod{q}$
2. distributes to any other shareholder D_j a partial update: $u_{i,j} \equiv \pi_i(j) \pmod{q}$
3. receives its partial updates and updates its share: $s_i \equiv s_i + \sum_j u_{j,i} \pmod{q}$
4. destroys the old share

- Participants are not malicious
  - Compute correct subshares
  - Destroy old shares
- *Eager* adversary

Proactivity

The Scheme

Each shareholder D_i:

1. picks at random $\pi_i \in \Pi_{t-1}$ such that $\pi_i(0) \equiv 0 \pmod{q}$
2. distributes to any other shareholder D_j a partial update: $u_{i,j} \equiv \pi_i(j) \pmod{q}$
3. receives its partial updates and updates its share: $s_i \equiv s_i + \sum_j u_{j,i} \pmod{q}$
4. destroys the old share

- Participants are not malicious
  - Compute correct subshares
  - Destroy old shares
- *Eager* adversary

Proactivity

The Scheme

Each shareholder D_i:

1. picks at random $\pi_i \in \Pi_{t-1}$ such that $\pi_i(0) \equiv 0 \pmod q$
2. distributes to any other shareholder D_j a partial update: $u_{i,j} \equiv \pi_i(j) \pmod q$
3. receives its partial updates and updates its share: $s_i \equiv s_i + \sum_j u_{j,i} \pmod q$
4. destroys the old share

- Participants are not malicious
  - Compute correct subshares
  - Destroy old shares
- *Eager* adversary

Proactivity

Correctness

$$S^{(\tau)} \equiv \sum_{i\in\Gamma} L_i s_i^{(\tau)} \equiv \sum_{i\in\Gamma} L_i \left(s_i^{(\tau-1)} + \sum_{j=1}^{n} \pi_j(i) \right) \equiv$$

$$\sum_{i\in\Gamma} L_i s_i^{(\tau-1)} + \sum_{j=1}^{n} \sum_{i\in\Gamma} L_i \pi_j(i) \equiv$$

$$S^{(\tau-1)} + \sum_{j=1}^{n} \pi_j(0) \equiv S^{(\tau)} \pmod q$$

Scalability

s_{new}

SSS based on polynomial interpolation are scalable: new shares for new shareholders

With a TD

- The Trusted Dealer simply computes the value of the polynomial in the new point: $s_{new} \equiv \pi(new) \pmod{q}$

Without any TD

t shareholders in Γ collaborating in the following protocol:

- Each shareholder D_i: $ps_i(new) \equiv s_i L_i(new) \pmod{q}$ for the new member
- The new member D_{new}: $\sum_{i \in \Gamma} ps_i(new) \equiv \sum_{i \in \Gamma} s_i L_i(new) \equiv s_{new} \pmod{q}$.

Scalability

s_{new}

SSS based on polynomial interpolation are scalable: new shares for new shareholders

With a TD

- The Trusted Dealer simply computes the value of the polynomial in the new point: $s_{new} \equiv \pi(new) \pmod q$

Without any TD

t shareholders in Γ collaborating in the following protocol:

- Each shareholder D_i: $ps_i(new) \equiv s_i L_i(new) \pmod q$ for the new member
- The new member D_{new}: $\sum_{i \in \Gamma} ps_i(new) \equiv \sum_{i \in \Gamma} s_i L_i(new) \equiv s_{new} \pmod q$.

Verifiability
[5]

Considerations

Malicious shareholders may cheat:

- providing incorrect shares
- providing incorrect subshares

VSS

- VSS permits to verify share correctness
- Solutions based on the hardness of inverting homomorphic functions
- Solution in [5] based on the hardness of computing the discrete logarithm over Z_p, for p prime.

Verifiability
[5]

Considerations

Malicious shareholders may cheat:

- providing incorrect shares
- providing incorrect subshares

VSS

- VSS permits to verify share correctness
- Solutions based on the hardness of inverting homomorphic functions
- Solution in [5] based on the hardness of computing the discrete logarithm over Z_p, for p prime.

Verifiability [5]

Considerations

Malicious shareholders may cheat:

- providing incorrect shares
- providing incorrect subshares

VSS

- VSS permits to verify share correctness
- Solutions based on the hardness of inverting homomorphic functions
- Solution in [5] based on the hardness of computing the discrete logarithm over Z_p, for p prime.

Verifiability

The Protocol

Let be:

- p and q two primes such that $p = mq + 1$, with m small integer
- $\pi(x) = a_0 + a_1x + \ldots + a_{t-1}x^{t-1}$ the polynomial on which shares are computed.

The dealer, after computing the shares:

1. chooses an element $g \in Z_p$ of order q
2. computes the witnesses $w_j \equiv g^{a_j} \pmod{p}$, $j = 0, \ldots, t-1$
3. makes the witnesses public

Each share s_i is verifiable: $g^{s_i} \stackrel{?}{\equiv} \prod_{j=0}^{t-1} (w_j)^{i^j} \pmod{p}$

Verifiability

The Protocol

Let be:

- p and q two primes such that $p = mq + 1$, with m small integer
- $\pi(x) = a_0 + a_1 x + \ldots + a_{t-1} x^{t-1}$ the polynomial on which shares are computed.

The dealer, after computing the shares:

1. chooses an element $g \in Z_p$ of order q
2. computes the witnesses $w_j \equiv g^{a_j} \pmod{p}$, $j = 0, \ldots, t-1$
3. makes the witnesses public

Each share s_i is verifiable: $g^{s_i} \stackrel{?}{\equiv} \prod_{j=0}^{t-1} (w_j)^{i^j} \pmod{p}$

Verifiability

The Protocol

Let be:

- p and q two primes such that $p = mq + 1$, with m small integer
- $\pi(x) = a_0 + a_1 x + \ldots + a_{t-1} x^{t-1}$ the polynomial on which shares are computed.

The dealer, after computing the shares:

1. chooses an element $g \in Z_p$ of order q
2. computes the witnesses $w_j \equiv g^{a_j} \pmod{p}$, $j = 0, \ldots, t-1$
3. makes the witnesses public

Each share s_i is verifiable: $g^{s_i} \stackrel{?}{\equiv} \prod_{j=0}^{t-1} (w_j)^{i^j} \pmod{p}$

Distributed Signatures

Signatures

- A Signature of a message m is a hash of m, encrypted with a secret key S: $Sig(m) = [H(m)]_S$
- The signature needs to be verifiable through a public key P: $[Sig(m)]_P = H(m)$

Distributed Signature

Many (co)signers want to/must sign the same message

- Each separately signs the message
- Impose a signing order over the (co)signers and let each sign one by one
- Repudiation is possible [6]

Distributed Signatures

Signatures

- A Signature of a message m is a hash of m, encrypted with a secret key S: $Sig(m) = [H(m)]_S$
- The signature needs to be verifiable through a public key P: $[Sig(m)]_P = H(m)$

Distributed Signature

Many (co)signers want to/must sign the same message

- Each separately signs the message
- Impose a signing order over the (co)signers and let each sign one by one
- Repudiation is possible [6]

Distributed Signatures

Signatures

- A Signature of a message m is a hash of m, encrypted with a secret key S: $Sig(m) = [H(m)]_S$
- The signature needs to be verifiable through a public key P: $[Sig(m)]_P = H(m)$

Distributed Signature

Many (co)signers want to/must sign the same message

- Each separately signs the message
- Impose a signing order over the (co)signers and let each sign one by one
- Repudiation is possible [6]

Distributed Signatures

Signatures

- A Signature of a message m is a hash of m, encrypted with a secret key S: $Sig(m) = [H(m)]_S$
- The signature needs to be verifiable through a public key P: $[Sig(m)]_P = H(m)$

Distributed Signature

Many (co)signers want to/must sign the same message

- Each separately signs the message
- Impose a signing order over the (co)signers and let each sign one by one
- Repudiation is possible [6]

Distributed Signatures

Distributed Signatures via Secret Sharing

- SSS can be adopted: the shared secret is the signing key
- Problem: the schemes above assume the secret is reconstructed
- Solution: multi-signatures (threshold signatures)

Benefits

- the service is distributed among several (co)certifiers
- no single (co)certifier knows the complete signing key but...

  it can collaborate with any group of $t-1$ other (co)certifiers
- full service as a centralized authority would provide
- (verifiability, proactivity, scalability, ...)

Distributed Signatures

Distributed Signatures via Secret Sharing

- SSS can be adopted: the shared secret is the signing key
- Problem: the schemes above assume the secret is reconstructed
- Solution: multi-signatures (threshold signatures)

Benefits

- the service is distributed among several (co)certifiers
- no single (co)certifier knows the complete signing key but...

  it can collaborate with any group of $t - 1$ other (co)certifiers
- full service as a centralized authority would provide
- (verifiability, proactivity, scalability, ...)

Distributed Signatures

Distributed Signatures via Secret Sharing

- SSS can be adopted: the shared secret is the signing key
- Problem: the schemes above assume the secret is reconstructed
- Solution: multi-signatures (threshold signatures)

Benefits

- the service is distributed among several (co)certifiers
- no single (co)certifier knows the complete signing key but...

  it can collaborate with any group of $t-1$ other (co)certifiers
- full service as a centralized authority would provide
- (verifiability, proactivity, scalability, ...)

Distributed Signatures

Distributed Signatures via Secret Sharing

- SSS can be adopted: the shared secret is the signing key
- Problem: the schemes above assume the secret is reconstructed
- Solution: multi-signatures (threshold signatures)

Benefits

- the service is distributed among several (co)certifiers
- no single (co)certifier knows the complete signing key but...

  it can collaborate with any group of $t-1$ other (co)certifiers
- full service as a centralized authority would provide
- (verifiability, proactivity, scalability, ...)

Distributed Signatures

Distributed Signatures via Secret Sharing

- SSS can be adopted: the shared secret is the signing key
- Problem: the schemes above assume the secret is reconstructed
- Solution: multi-signatures (threshold signatures)

Benefits

- the service is distributed among several (co)certifiers
- no single (co)certifier knows the complete signing key but...

  it can collaborate with any group of $t-1$ other (co)certifiers
- full service as a centralized authority would provide
- (verifiability, proactivity, scalability, ...)

Distributed Signatures

Distributed Signatures via Secret Sharing

- SSS can be adopted: the shared secret is the signing key
- Problem: the schemes above assume the secret is reconstructed
- Solution: multi-signatures (threshold signatures)

Benefits

- the service is distributed among several (co)certifiers
- no single (co)certifier knows the complete signing key but...

  it can collaborate with any group of $t - 1$ other (co)certifiers
- full service as a centralized authority would provide
- (verifiability, proactivity, scalability, ...)

Distributed Signatures

Distributed Signatures via Secret Sharing

- SSS can be adopted: the shared secret is the signing key
- Problem: the schemes above assume the secret is reconstructed
- Solution: multi-signatures (threshold signatures)

Benefits

- the service is distributed among several (co)certifiers
- no single (co)certifier knows the complete signing key but...

  it can collaborate with any group of $t-1$ other (co)certifiers
- full service as a centralized authority would provide
- (verifiability, proactivity, scalability, ...)

RSA Signatures

History

- Ron Rivest, Adi Shamir, Len Adleman, MIT 1977
- Clifford Cocks described an equivalent (classified) system in 1973: never deployed

RSA Signatures

Key Generation

1. Choose two large primes p and q such that $p \neq q$, randomly and independently
2. Compute $N = pq$
3. Compute the totient $\phi(N) = (p-1)(q-1)$
4. Choose an integer $1 < e < \phi(N)$ which is coprime to $\phi(N)$
5. Compute d such that $de \equiv 1 \pmod{\phi(N)}$

- Public key: (e,N)
- Private key: (d,N)

- Fermat Little Theorem
- Extended Euclidean Algorithm

RSA Signatures

Key Generation

1. Choose two large primes p and q such that $p \neq q$, randomly and independently
2. Compute $N = pq$
3. Compute the totient $\phi(N) = (p-1)(q-1)$
4. Choose an integer $1 < e < \phi(N)$ which is coprime to $\phi(N)$
5. Compute d such that $de \equiv 1 \pmod{\phi(N)}$

- Public key: (e,N)
- Private key: (d,N)

- Fermat Little Theorem
- Extended Euclidean Algorithm

RSA Signatures

Signature Issuance

In order to sign a message m:

1. $Sig(m) = H(m)^d \pmod{N}$
2. send $< m, Sig(m) >$

Signature Verification

In order to verify a signature for a message m

1. compute $H(m)$
2. $Sig(m)^e \stackrel{?}{\equiv} H(m) \pmod{N}$

RSA Signatures

Signature Issuance

In order to sign a message m:

1. $Sig(m) = H(m)^d \pmod N$
2. send $< m, Sig(m) >$

Signature Verification

In order to verify a signature for a message m

1. compute $H(m)$
2. $Sig(m)^e \stackrel{?}{\equiv} H(m) \pmod N$

RSA Multi-signatures

- A multi-signature scheme, enables a given group of shareholders to act as a signing authority in a totally distributed environment
- Each (co)signer signs the same message separately and independently
- The individual (partial) signatures are combined into a multi-signature

RSA Multi-signatures

Protocol [7]

- Setup: $N, e, d, \pi(0) = e, D_j \leftarrow s_j \equiv \pi(j)L_j(0) \pmod{N}$
- Each (co)certifier D_j issues a partial signature $psig_j(m) = m^{s_j} \pmod{N}$
- The verifier checks whether

$$\left(\prod_{j\in\Gamma} psig_j(m)\right)^d \stackrel{?}{\equiv} \left(\prod_{j\in\Gamma} m^{s_j}\right)^d \equiv \left(m^{\sum_{j\in\Gamma}\pi(j)L_j(0)}\right)^d \equiv$$

$$\equiv (m^e)^d \equiv m \pmod{N}$$

Unfortuantely the scheme is NOT verifiable

RSA Multi-signatures

Protocol [7]

- Setup: $N, e, d, \pi(0) = e, D_j \leftarrow s_j \equiv \pi(j)L_j(0) \pmod{N}$
- Each (co)certifier D_j issues a partial signature $psig_j(m) = m^{s_j} \pmod{N}$
- The verifier checks whether

$$\Big(\prod_{j\in\Gamma} psig_j(m)\Big)^d \stackrel{?}{\equiv} \Big(\prod_{j\in\Gamma} m^{s_j}\Big)^d \equiv \Big(m^{\sum_{j\in\Gamma}\pi(j)L_j(0)}\Big)^d \equiv$$
$$\equiv (m^e)^d \equiv m \pmod{N}$$

Unfortuantely the scheme is NOT verifiable

References

C. L. Liu, *Introduction to Combinatorial Mathematics*. New York: McGraw-Hill, 1968.

A. Shamir, “How to share a secret.” *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979.

H. Luo and S. Lu, “Providing robust and ubiquitous security support for mobile ad hoc networks.” Dept. of Computer Science, UCLA, Tech. Rep. TR-200030, 2000.

A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung, “Proactive secret sharing or: How to cope with perpetual leakage,” in *CRYPTO ’95: Proceedings of the 15th Annual International Cryptology Conference on Advances in Cryptology*. London, UK: Springer-Verlag, 1995, pp. 339–352.

P. Feldman, “A practical scheme for non-interactive verifiable secret sharing,” in *FOCS ’87: Proceedings of the 28th IEEE symposium on Foundations of Computer Science*, 1987, pp. 427–438.

S.-P. Shieh, C.-T. Lin, W.-B. Yang, and H.-M. Sun, “Digital multisignature scheme for authenticating delegates in mobile code systems,” *IEEE Transactions on Vehicular Technology*, vol. 49, no. 4, pp. 1464–1473, 2000.

J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang, “Providing robust and ubiquitous security support for mobile ad hoc networks.” in *ICNP*, 2001, pp. 251–260.