III – Distributed Cryptography

David Pointcheval
Ecole normale supérieure/PSL, CNRS & INRIA

Outline

Secret Sharing

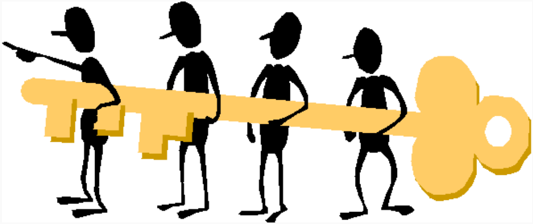
Outline

Key Management

In case of a critical private key (decryption or signing key)

- Abuse: one user can use the secret key alone
- Loss: in case of loss of the key (destruction)

$\Longrightarrow$ share the secret key between several users



Let $S \in \{0,1\}^\ell$ be a secret bit-string to be shared between two people (Alice and Bob):

- one chooses a random $S_1 \in \{0,1\}^\ell$, and sends it to Alice
- one computes $S_2 = S \oplus S_1$, and sends it to Bob

Security:

- Alice knows a random value
- Bob knows a value masked by a random value: a random value!

$\Longrightarrow$ individually, they have no information on S

Together, they can recover $S = S_1 \oplus S_2$

Secret Sharing Schemes

Let $S \in \{0,1\}^\ell$ be a secret bit-string to be shared between n people (U_1, ..., U_n):

- one chooses random values $S_i \in \{0,1\}^\ell$, for $i = 1, \ldots, n-1$ and sends S_i to U_i
- one computes $S_n = S \oplus S_1 \oplus \ldots \oplus S_{n-1}$, and sends it to U_n

Security:

- U_1, ..., U_{n-1} know random values
- U_n knows a value masked by random values: a random value!

$\Longrightarrow$ individually, they have no information on S

$\Longrightarrow$ but also, any subgroup of $(n-1)$ people has no information on S

All together, they can recover $S = S_1 \oplus \ldots \oplus S_n$

Unconditional Security

Any subgroup of $(n-1)$ people has no information on S!

$\Longrightarrow$ if one people does not want / is not able to cooperate:

S is lost forever!

Threshold Secret Sharing

(n, k)-Threshold Secret Sharing

A secret S is shared among n users:

- any subgroup of k people (or more) can recover S
- any subgroup of less than k people has no information about S

Outline

Secret Sharing

Introduction

Shamir Secret Sharing

Verifiable Secret Sharing

Distributed Cryptography

Lagrange Interpolation of Polynomials

Let us be given k points $(x_1, y_1), \ldots, (x_k, y_k)$, with distinct abscissa.
There exists a unique polynomial P

- of degree $k-1$
- such that $P(x_i) = y_i$ for $i = 1, \ldots, k$

$$L_j(X) = \prod_{\substack{i=1\\ i\neq j}}^{i=k} \frac{X - x_i}{x_j - x_i} \quad \begin{cases} L_j(x_j) & = & 1 \\ L_j(x_i) & = & 0 \quad \text{for all } i \neq j \end{cases}$$

As a consequence:

$$P(X) = \sum_{j=1}^{k} y_j L_j(X) \text{ satisfies } \begin{cases} \deg(P) = k-1 \\ P(x_i) = y_i \quad \forall i = 1, \ldots, k \end{cases}$$

Shamir Secret Sharing: (n,k)-Threshold

For any subset $\mathcal{S}$ of k indices:

$$L_{\mathcal{S},j}(X) = \prod_{\substack{i\in\mathcal{S}\\ i\neq j}} \frac{X - x_i}{x_j - x_i} \quad \begin{cases} L_{\mathcal{S},j}(x_j) & = & 1 \\ L_{\mathcal{S},j}(x_i) & = & 0 \quad \text{for all } i \in \mathcal{S}, i \neq j \end{cases}$$

and

$$P(X) = \sum_{j\in\mathcal{S}} y_j L_{\mathcal{S},j}(X) : S = P(0) = \sum_{j\in\mathcal{S}} y_j L_{\mathcal{S},j}(0)$$

If one notes $\lambda_{\mathcal{S},j} = L_{\mathcal{S},j}(0)$ (that can be publicly computed)

$$x = \sum_{j\in\mathcal{S}} y_j \lambda_{\mathcal{S},j}.$$

Outline

Secret Sharing

Verifiable Secret Sharing

[Chor-Goldwasser-Micali-Awerbuch – FOCS '85]

If Eve claims she shared her decryption key: how can we trust her?

- we try to recover the key?
- how to do without revealing additional information?

$\Longrightarrow$ Verifiable Secret Sharing

For DL Keys [Feldman – FOCS '87]

Eve's keys are, in a group $\mathbb{G} = \langle g \rangle$ of prime order q,

$$sk = x \qquad pk = y = g^x$$

(n, k)-Secret sharing: $x = P(0)$ for $P(X) = \sum_{i=0}^{k-1} a_i X^i$

$\Longrightarrow S_i = P(i)$ for $i = 1 \ldots, n$

For any subset $\mathcal{S}$ of k indices:

- $x = \sum_{j \in \mathcal{S}} S_j \lambda_{\mathcal{S},j}$
- $y = g^x = g^{\sum_{j \in \mathcal{S}} S_j \lambda_{\mathcal{S},j}} = \prod_{j \in \mathcal{S}} (g^{S_j})^{\lambda_{\mathcal{S},j}} = \prod_{j \in \mathcal{S}} v_j^{\lambda_{\mathcal{S},j}}$ for $v_j = g^{S_j}$

Verifiable Secret Sharing for DL Keys

For DL Keys

[Feldman – FOCS '87]

Eve's keys are, in a group $\mathbb{G} = \langle g \rangle$ of prime order q,

$$sk = x \qquad pk = y = g^x$$

(n, k)-Secret sharing: $x = P(0)$ for $P(X) = \sum_{i=0}^{k-1} a_i X^i$

- Eve computes $S_i = P(i)$ for $i = 1 \ldots, n$ and $v_i = g^{S_i}$
- Eve sends each S_i privately to each U_i
- Eve publishes $v_i = g^{S_i}$ for $i = 1, \ldots, n$
- Each U_i can then check its own v_i w.r.t. to its S_i
- Anybody can check

  $$y = \prod_{j \in \mathcal{S}} v_j^{\lambda_{\mathcal{S},j}}$$

  for any subset $\mathcal{S}$ of size k

Distributed Cryptography

Outline

Secret Sharing vs. Distributed Cryptography

If Eve shares her decryption key sk,
the (U_i) will have to cooperate to recover the key sk
and then decrypt the ciphertext

But then, they all know the decryption key sk!

How can the (U_i) use their shares (S_i) to decrypt (or sign),
without leaking any additional information about sk?

$\Longrightarrow$ Multi-party computation

Let us try on ElGamal decryption (with shared DL keys)

Outline

ElGamal Encryption

[ElGamal 1985]

ElGamal Encryption

In a group $\mathbb{G} = \langle g \rangle$ of order q

- $\mathcal{K}(\mathbb{G}, g, q)$: $x \stackrel{R}{\leftarrow} \mathbb{Z}_q$, and $sk \leftarrow x$ and $pk \leftarrow y = g^x$
- $\mathcal{E}_{pk}(m)$: $r \stackrel{R}{\leftarrow} \mathbb{Z}_q$, $c_1 \leftarrow g^r$ and $c_2 \leftarrow y^r \times m$.
  Then, the ciphertext is $c = (c_1, c_2)$
- $\mathcal{D}_{sk}(c)$ outputs c_2 / c_1^x

We assume an (n, k)-secret sharing of x
and a qualified set $\mathcal{S}$: $x = \sum_{j \in \mathcal{S}} S_j \lambda_{\mathcal{S},j}$

$\mathcal{D}_{sk}(c) = c_2 / c_1^x$: one needs to compute c_1^x

$$c_1^x = c_1^{\sum_{j \in \mathcal{S}} S_j \lambda_{\mathcal{S},j}} = \prod_{j \in \mathcal{S}} (c_1^{S_j})^{\lambda_{\mathcal{S},j}}$$

Each user computes $C_j = c_1^{S_j}$, and then $c_1^x = \prod_{j \in \mathcal{S}} C_j^{\lambda_{\mathcal{S},j}}$

Robustness

In a group $\mathbb{G} = \langle g \rangle$ of order q

- $\mathcal{K}(\mathbb{G}, g, q)$: $x \stackrel{R}{\leftarrow} \mathbb{Z}_q$, and $sk \leftarrow x$ and $pk \leftarrow y = g^x$
- $\mathcal{E}_{pk}(m)$: $r \stackrel{R}{\leftarrow} \mathbb{Z}_q$, $c_1 \leftarrow g^r$ and $c_2 \leftarrow y^r \times m$.
  Then, the ciphertext is $c = (c_1, c_2)$
- $\mathcal{D}_{sk}(c)$ outputs c_2 / c_1^x

Given a qualified set $\mathcal{S}$: $x = \sum_{j \in \mathcal{S}} S_j \lambda_{\mathcal{S},j}$

Each user computes $C_j = c_1^{S_j}$, and then $c_1^x = \prod_{j \in \mathcal{S}} C_j^{\lambda_{\mathcal{S},j}}$

Assume Charlie a.k.a. U_1, sends a random C_1:

- the others will compute a wrong decryption
- Charlie will be able to extract the plaintext!

Fraud Detection

Each user computes $C_j = c_1^{S_j}$, and then $c_1^x = \prod_{j \in \mathcal{S}} C_j^{\lambda_{\mathcal{S},j}}$

But U_1, sends a random C_1: instead of $c_1^{S_1}$, knowing also $v_1 = g^{S_1}$
$\Longrightarrow$ Decide a DDH tuple (g, c_1, v_1, C_1)

Robustness

A defrauder can be detected

$\Longrightarrow$ Proof of DDH membership for the tuple (g, c_1, v_1, C_1),
without leakage of any information about S_1

NIZK Diffie-Hellman Language

In a group $\mathbb{G} = \langle g \rangle$ of prime order q,

the $\mathbf{DDH}(g, h)$ assumption states it is hard to distinguish

$$\mathcal{L} = (u = g^x, v = h^x) \text{ from } \mathbb{G}^2 = (u = g^x, v = h^y)$$

- $\mathcal{P}$ knows x, such that $(u = g^x, v = h^x)$ and wants to prove it
- $\mathcal{P}$ chooses $k \stackrel{R}{\leftarrow} \mathbb{Z}_q^\star$, sets $U = g^k$ and $V = h^k$
- $\mathcal{P}$ computes $h = \mathcal{H}(g, h, u, v, U, V) \in \mathbb{Z}_q$
- $\mathcal{P}$ computes $s = k - xh \bmod q$

The proof consists of the pair (h, s):
anybody can check whether $h = \mathcal{H}(g, h, u, v, g^s u^h, h^s v^h)$

This proof allows to detect the defrauder

Outline

Schnorr Signature

- $\mathbb{G} = \langle g \rangle$ of order q and $\mathcal{H}$: $\{0,1\}^\star \to \mathbb{Z}_q$
- Key Generation $\to (y, x)$: $x \in \mathbb{Z}_q^\star$ and $y = g^{-x}$
- Signature of $m \to (r, h, s)$
  $k \stackrel{R}{\leftarrow} \mathbb{Z}_q^\star \quad r = g^k \quad h = \mathcal{H}(m, r) \quad s = k + xh \bmod q$
- Verification of (m, r, s)
  compute $h = \mathcal{H}(m, r)$ and check $r \stackrel{?}{=} g^s y^h$

We assume an (n, k)-secret sharing of x (with the v_i)
and a qualified set $\mathcal{S}$: $x = \sum_{j \in \mathcal{S}} S_j \lambda_{\mathcal{S},j}$

The users generate a common r and then sign (m, r)
with a partial signature s_i under v_i:

$\Longrightarrow$ the linearity leads to a global signature

Distributed Schnorr Signature

- $\mathbb{G} = \langle g \rangle$ of order q and $\mathcal{H}$: $\{0,1\}^\star \to \mathbb{Z}_q$
- Key Generation $\to (y, x)$: $x \in \mathbb{Z}_q^\star$ and $y = g^{-x}$
  We assume an (n, k)-secret sharing of x (with the $v_i = g^{S_i}$)
  and a qualified set $\mathcal{S}$: $x = \sum_{j \in \mathcal{S}} S_j \lambda_{\mathcal{S},j}$
- Signature of $m \to (r, h, s)$
  - each U_i chooses $k_i \stackrel{R}{\leftarrow} \mathbb{Z}_q^\star$ and publishes $r_i = g^{k_i}$
  - they all compute $r = \prod r_i^{\lambda_{\mathcal{S},j}}$ and $h = \mathcal{H}(m, r)$
  - each U_i computes and publishes $s_i = k_i + S_i h \bmod q$

  Then, $s = \sum s_i \lambda_{\mathcal{S},i}$
- Verification of (m, r, s)
  compute $h = \mathcal{H}(m, r)$ and check $r \stackrel{?}{=} g^s y^h$

Each partial signature (m, r_i, s_i) can be checked: $r_i \stackrel{?}{=} g^{s_i} v_i^h$

Outline

Distributed Key Generation

In the previous schemes (ElGamal encryption and Schnorr signature) the keys are generated in a centralized way:

someone knows the secret key!

Distributed cryptography should include a distributed key generation:

the secret key should never exist in one place.

(n, n)-Threshold DL Key Generation

- $\mathbb{G} = \langle g \rangle$ of order q
- Key Generation $\rightarrow (y, x)$:
  - each U_i chooses $x_i \stackrel{R}{\leftarrow} \mathbb{Z}_q^\star$ and publishes $y_i = g^{x_i}$
  - anybody can compute $y = \prod y_i = g^{\sum x_i}$

The public key y corresponds to the "virtual" secret key

$$x = \sum x_i \bmod q$$

Distributed Key Generation

(n, k)-Threshold DL Key Generation

- $\mathbb{G} = \langle g \rangle$ of order q
- Key Generation $\rightarrow (y, x)$:
  - each U_i chooses a polynomial P_i of degree $k - 1$, and sends $S_{i,j} = P_i(j)$ to U_j
  - each U_j can then compute $S_j = \sum_i S_{i,j} = \sum_i P_i(j) = P(j)$, where $P = \sum_i P_i$
  - each U_j computes and publishes $v_j = g^{S_j}$

The $(S_j)_j$ are an (n, k)-secret sharing of the "virtual" secret key x, corresponding to the public key y, that anybody can compute:
For any qualified set $\mathcal{S}$:

- Secretly: $x = \sum_{j \in \mathcal{S}} S_j \lambda_{\mathcal{S},j} \bmod q$
- Publicly: $y = \prod_{j \in \mathcal{S}} v_j^{\lambda_{\mathcal{S},j}}$