<u>6.857 Computer and Network Security</u>
Lecture 7

Admin:

- Notes from previous semester (only read the section on secret sharing)

<u>Today</u>:

- Shamir's "secret sharing"

TOPIC:

DATE:

FILE UNDER:

PAGE: L19.10

Key management

Start with "secret sharing" (threshold cryptography).

- Assume Alice has a secret s. (e.g. a key)
- She wants to protect s as follows:

  She has n friends $A_1, A_2, \ldots, A_n$

  She picks a "threshold" t, $1 \le t \le n$.

  She wants to give each friend A_i,

  a "share" s_i of s, so that

  - any t or more friends can reconstruct s
  - any set of $< t$ friends can not.

Easy cases:

$t = 1$: $s_i = s$

$t = n$: $s_1, s_2, \ldots, s_{n-1}$ random

s_n chosen so that

$$s = s_1 \oplus s_2 \oplus \cdots \oplus s_n$$

What about $1 < t < n$?

Shamir's method ("How to Share a Secret", 1979)

Idea: 2 points determine a line
3 points determine a quadratic
...
t points determine a degree $(t-1)$ curve

Let $f(x) = a_{t-1}x^{t-1} + a_{t-2}x^{t-2} + \cdots + a_1 x + a_0$

There are t coefficients. Let's work modulo prime p.

We can have t points: (x_i, y_i) for $1 \le i \le t$

They determine coefficients, and vice versa.

$$\begin{tikzcd}
\{(x_i, y_i)\} \arrow[r, bend right, "\text{Polynomial Interpolation}"'] & (a_{t-1}, a_{t-2}, \ldots, a_1, a_0) \arrow[l, bend right, "\text{Polynomial Evaluation}"']
\end{tikzcd}$$

Pt/value pairs — Coefficients

To share secret s (here $0 \le s < p$):

Let $y_0 = a_0 = s$

Pick $a_1, a_2, \ldots, a_{t-1}$ at random from $\mathbb{Z}_p$

Let share $s_i = (i, y_i)$ where $y_i = f(i)$, $1 \le i \le n$.

Evaluation is easy.

Interpolation

Given (x_i, y_i) $\quad 1 \le i \le t \quad$ (wlog)

Then $f(x) = \sum_{i=1}^{t} f_i(x) \cdot y_i$

where $f_i(x) = \begin{cases} 1 & \text{at } x = x_i \\ 0 & \text{for } x = x_j,\ j \ne i,\ 1 \le j \le t \end{cases}$

Furthermore:

$$f_i(x) = \frac{\prod_{j \ne i} (x - x_j)}{\prod_{j \ne i} (x_i - x_j)}$$

This is a polynomial of degree $t-1$. So f also has degree $t-1$.

Evaluating $f(0)$ to get s simplifies to

$$s = f(0) = \sum_{i=1}^{t} y_i \cdot \frac{\prod_{j \ne i} (-x_j)}{\prod_{j \ne i} (x_i - x_j)} .$$

Theorem: Secret sharing with Shamir's method is information-theoretically secure. Adversary with $< t$ shares has no information about s.

Pf: A degree $t-1$ curve can go through any point $(0, s)$ as well as any given d pts (x_i, y_i), if $d < t$. ☒

Refs: Reed-Solomon codes, erasure codes, error correction, information dispersal (Rabin).

TOPIC:

DATE:

FILE UNDER:

PAGE: L19.2

"Gap group" is one in which

- DDH is easy ("Decision Diffie Hellman")

  [Recall: given (g, g^a, g^b, g^c), to decide if $ab = c \pmod{\text{order}(g)}$]

but

- CDH is hard ("Computational Diffie Hellman")

  [Recall: given (g, g^a, g^b), to compute g^{ab}

(Note that CDH easy $\Rightarrow$ DDH easy)

This difference in difficulty between DDH ("easy") and CDH ("hard") forms a "gap".

— How can one construct a "gap group"?

— What good would that be?

TOPIC:

DATE:

FILE UNDER:

Bilinear maps

Suppose: G_1 is group of prime order q, with generator g

G_2 is group of prime order q, with generator h

[we use multiplicative notation for both groups]

and there exists a (bilinear) map

$$e : G_1 \times G_1 \longrightarrow G_2$$

such that

$e(g,g) = h$

$$\boxed{(\forall a,b)\ e(g^a, g^b) = h^{ab}} \quad !!!$$

$$= e(g, g^{ab})$$

$$= e(g,g)^{ab}$$

$$= e(g, g^b)^a$$

$$= e(g, g^a)^b$$

$$= e(g^b, g^a)$$

...

Bilinear maps also called "pairing functions"

They have an enormous number of applications.*

We are, of course, interested in efficiently computable bilinear maps.

* google: "The pairing-based crypto lounge"

TOPIC:

DATE:

FILE UNDER:

Theorem:

If there is a bilinear map

$$e: G_1 \times G_1 \to G_2$$

between two groups of prime order q,

then DDH is easy in G_1

Proof:

Given (g, g^a, g^b, g^c) (elements of G_1)

then

$$c = ab \pmod{q} \iff \underbrace{e(g^a, g^b) = e(g, g^c)}_{\equiv \underbrace{h^{ab} = h^c}_{\equiv ab = c \pmod{q}}}$$

So: accept (g, g^a, g^b, g^c) iff $e(g^a, g^b) = e(g, g^c)$.

∎

Even though DDH is easy in G_1, CDH may still be hard; we may have a "gap group".

How to construct gap groups (with bilinear maps):

- This is not simple! We give just a sketch.
- G_1 will be "supersingular" elliptic curve

  e.g. elliptic curve defined by points on

  $$y^2 = x^3 + ax + b \pmod p$$

  where $p = 2 \pmod 3$, $p \geq 5$

  $a = 0$

  $b \in Z_p^*$ (can choose $b = 1$)
- G_2 is finite field $\mathbb{F}_{p^k}$ for some small k

  (can use subgroups of G_1 & G_2 by choosing generators of order $\approx 2^{160}$ say...)

- e (bilinear map) is implemented as a "Weil pairing" or a "Tate pairing".

Application 1: Digital signatures

(Boneh, Lynn, Shacham (2001))

Signatures are short (e.g. 160 bits)!

Public: groups G_1, G_2 of prime order q

pairing function $e: G_1 \times G_1 \to G_2$

g = generator of G_1

H = hash fn (C.R.) from messages to G_1

Secret key: x where $0 < x < q$

Public key: $y = g^x$ (in G_1)

To sign message M:

Let $m = H(M)$ (in G_1)

Note: Signature may be short! Just one element of G_1.

Output $\sigma = \sigma_x(M) = m^x$ (in G_1)

To verify (y, M, σ):

Check $e(g, \sigma) \stackrel{?}{=} e(y, m)$ where $m = H(M)$

$e(g,m)^x$ in both cases

Theorem: BLS signature scheme secure against existential forgery under chosen message attack in ROM assuming CDH is hard in G_1.

TOPIC:

DATE:

FILE UNDER:

Application 3:

Identity-based encryption (IBE) [Boneh, Franklin '01]

TTP (trusted third party) publishes

G_1, G_2, e (bilinear map), g (generator of G_1), y

where $y = g^s$ & s is TTP's master secret.

Let H_1 be random oracle mapping names (e.g. "alice@mit.edu") to elements of G_1

Let H_2 be random oracle mapping G_2 to $\{0,1\}^*$ (PRG).

Want to enable anyone to encrypt message for Alice knowing only TTP public parameters & Alice's name

Encrypt(y, name, M):

$r \xleftarrow{R} Z_q^*$ (here prime $q = |G_1| = |G_2|$)

$g_A = e(Q_A, y)$ where $Q_A = H_1(\text{name})$

output $(g^r, M \oplus H_2(g_A^r))$

TOPIC:

DATE:

FILE UNDER:

PAGE: L19.9

Decrypt ciphertext $c = (u, v)$:

- Alice obtains $d_A = Q_A^s$ from TTP (once is enough) where $Q_A = H_1(\text{name})$.

  This is Alice's decryption key.

  Note that TTP also knows it!

  Note that message may be encrypted before Alice gets d_A.

- Compute $v \oplus H_2(e(d_A, u))$

$$
\begin{aligned}
&= v \oplus H_2(e(Q_A^s, g^r)) \\
&= v \oplus H_2(e(Q_A, g)^{rs}) \\
&= v \oplus H_2(e(Q_A, g^s)^r) \\
&= v \oplus H_2(e(Q_A, y)^r) \\
&= v \oplus H_2(g_A^r) \\
&= M
\end{aligned}
$$

Application 2: Three-way key agreement (Joux, generalizing DH)

Recall DH: $A \to B: g^a$
$B \to A: g^b$
key $= g^{ab}$

Joux: Suppose G_1 has generator g
Suppose $e: G_1 \times G_2$ is a bilinear map.

$A \to B, C: g^a$

$B \to A, C: g^b$

$C \to A, B: g^c$

A computes $e(g^b, g^c)^a = e(g,g)^{abc}$

B computes $e(g^a, g^c)^b = e(g,g)^{abc}$ =.

C computes $e(g^a, g^b)^c = e(g,g)^{abc}$

key $= e(g,g)^{abc}$

Secure assuming "BDH" ≡

given g, g^a, g^b, g^c, e

hard to compute $e(g,g)^{abc}$

Four-way key agreement is open problem!

(maybe... see Garg/Gentry/Halevi Proc. Eurocrypt '13)