6.857 Rivest
3/30/09 L14.1

Admin:
- Project topic drafts due **Wed** (4/1)
- Quiz in-class **Monday** (4/6) [covers through 4/1]

Outline:

- ☐ El Gamal sigs (review)
- ☐ DSS - digital signature standard
- ☐ Secret-sharing (threshold cryptography)
    - ☐ information-theoretic security
    - ☐ computational security

☑ ~~[illegible]~~

El Gamal signatures

Public system parameters p prime

g generator

Keygen: $x \in_R \{0,1,\dots,p-2\}$ $SK = x$

$y = g^x$ $PK = y$

Sign(M): $m = h(M)$

$k \in_R Z_{p-1}^*$ [$\gcd(k, p-1) = 1$]

randomized signing

$r = g^k$ [hard work is indep of M]

$ks + rx = m$ $s = \frac{(m - rx)}{k} \pmod{p-1}$

$\sigma(M) = (r, s)$

Verify: check $0 < r < p$

" $y^r r^s = g^m \pmod p$ where $m = h(M)$

Return True if both checks pass else return False

Correctness: $g^{rx} g^{sk} = \underbrace{g^{rx+sk} \overset{?}{=} g^m}_{\equiv} \pmod p$

$rx + ks \overset{?}{=} m \pmod{p-1}$

$\equiv$

$s = \frac{(m - rx)}{k} \pmod{p-1}$

(if $\gcd(k, p-1) = 1$)

[El Gamal signatures, cont.d]

That was original version.

__Theorem:__ El Gamal is existentially __forgeable__ (without h fn or h = identity)
↑ note: CR!

__Proof:__ Let ~~e~~ $e \in_R Z_{p-1}$

$$r \leftarrow g^e y \pmod p$$

$$s \leftarrow -r \pmod{p-1}$$

(r,s) is sig for message $m = es \pmod{p-1}$

$$y^r r^s \stackrel{?}{=} g^m$$

$$g^{xr}(g^e y)^{-r} = g^{-er} = g^{es} = g^m \quad \text{for } m = es \pmod{p-1}$$

__But:__ It is easy to fix.

__Modified El Gamal__ (Pointcheval/Stern 1996)

sign(M): ~~k~~ $k \in_R Z_p^*$

$$r = g^k \pmod p$$

$$m = h(M \| r) \quad \Longleftarrow ***$$

$$s = \frac{(m - rx)}{k} \bmod p-1$$

$$\sigma(M) = (r,s)$$

__Verify:__ check $0 < r < p$
check $y^r r^s = g^m$ where $m = h(M\|r)$.

<u>Thm</u>: (Modified) El Gamal is existentially unforgeable against adaptive chosen message attack, in ROM, assuming DLP is hard.

Digital Signature Standard (DSS - NIST 1991)

Public parameters:

q prime $\quad |q| = 160$ bits

$p = nq+1$ prime $\quad |p| = 1024$ bits

g_0 generates Z_p^*

$g = g_0^n$ generates G_q - subgroup of Z_p^* of order q

Keygen:

$x \in_R Z_q \quad$ SK $\quad |x| = 160$ bits

$y = g^x \pmod p \quad$ PK $\quad |y| = 1024$ bits

Sign(M):

$k \in_R Z_q^* \quad$ (i.e. $1 \le k < q$)

$r = (g^k \bmod p) \pmod q \quad |r| = 160$ bits

$m = h(M)$

$s = (m + rx)/k \pmod q \quad |s| = 160$ bits

redo if $r = 0$ or $s = 0$

$\sigma(M) = (r, s) \quad |\sigma| = 320$ bits

Verify (PK, M, (r,s))

Check $y^{r/s} g^{m/s} \pmod p \pmod q \stackrel{?}{=} r$

where $m = h(M)$

Correctness:

$g^{(rx+m)/s} \stackrel{?}{=} r \pmod p \pmod q$

$g^k \stackrel{?}{=} r \pmod p \pmod q$ ✓

Security proof works if we had done $m = h(M||r)$, as before.
As it stands, existentially forgeable for h = identity.

Secret Sharing (Threshold cryptography)

Alice has a secret document (or key) s.

She wants to protect it as follows:

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

she wants to give each of them a "share" of s.

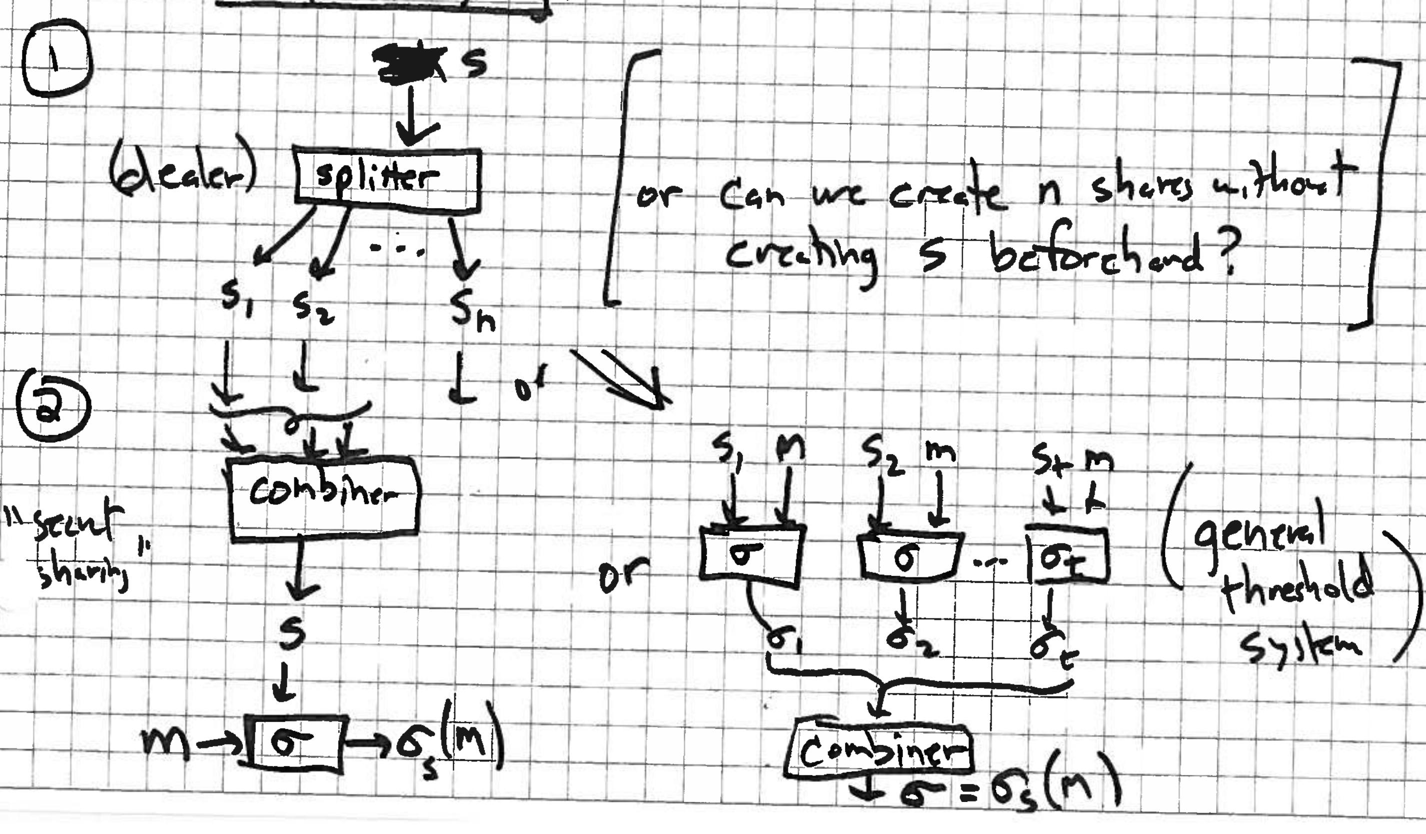
she picks a "threshold" t, $1 \le t \le n$.

she wants it to be possible that any t or more of her friends can reconstruct (or use) s, while any set of $< t$ friends can not.

[If s is secret key, note distinction between

- reconstructing s (for use)
- getting same effect via shares]

Example: signing:



[or can we create n shares without creating s beforehand?]

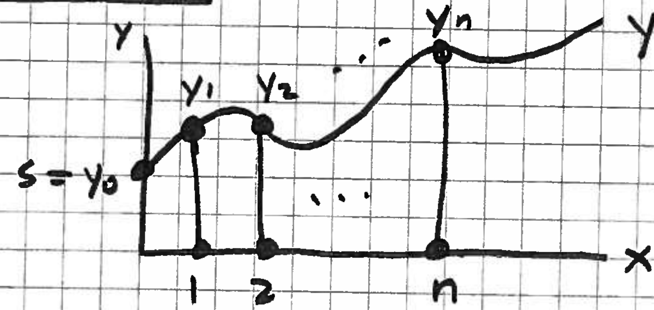
Secret Sharing:

$t=1$: $s_1 = s \quad 1 \le i \le n$

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

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

[redacted]$< t < n$??:



$y = f(x)$ degree $t-1$

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

$\pmod p$

t points (x_i, y_i) $1 \le i \le t$ determine a unique polynomial of degree $t-1$.

Given s: let $y_0 = s = a_0$

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

$s_i = \text{pair } (i, y_i) \; : \; y_i = f(i) \quad 1 \le i \le n$

How to reconstruct??

Evaluation

dual is Interpolation

$$t \text{ pt-value pairs} \underset{\text{Interpolation}}{\overset{\text{Evaluation}}{\rightleftarrows}} t \text{ coeffs}$$

Interpolation

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

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

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

is clearly ok

But $f_i(x) = \dfrac{\prod_{j \ne i}(x - x_j)}{\prod_{j \ne i}(x_i - x_j)}$ $\quad$ polynomial of degree $t-1$

$\therefore$ f is poly of degree $t-1$

$\Big(= \underset{x_j \ne x_i}{0}$ or $\underset{x_j = x_i}{1}$ at x_j as appropriate... $\Big)$

Evaluating $f(0)$ to get $x_0 = s$: simplifies to

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

Thm: Secret-sharing is <u>information-theoretically secure</u>

Adversary with $< t$ shares has <u>no</u> information re s.

Pf: Can create consistent polynomial of degree $t-1$ going through given points at <u>any</u> point $(0, s)$ $\quad 0 \le s < p$.

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

Note: We have $0 \le m < p$ so each share is "as big as" m.

Shares are (i, y_i); each y_i "same size as" p or m.

If file is very large: can break into pieces, share each piece. (bytes) $GF(2^8)$

But still, each person gets total share as big as original m.

How to reduce share sizes?

Example: I have 100 MB file to share among $n=20$ friends;
I want $t=10$ to be able to reconstruct it.
With original scheme, each gets 100 MB share.
Goal: each has $100\text{MB}/t = 10$ MB share. ?

Idea 1: Give up information-theoretic (unconditional) security for computational security.

Idea 2: Encrypt message, then share ~~the~~ ciphertext, & share key.

Idea 3: In secret-sharing of ciphertext, secret is set of all t coefficients, not just constant term. (No longer info-theoretic secure; each point on curve eliminates some polynomials. t shares reduces possibilities from p^t down to 1. Each share reduces # polynomials by factor of p.)

Given (M, n, t)

M = message to be shared (100 MB)

K = AES key $= s$ (128 bits)

Share $s \Rightarrow n$ shares $s_1, s_2, \ldots, s_n$
s.t. any t can reconstruct s
(info-theoretically secure)
Give i^{th} party s_i, $1 \le i \le n$. (and i)

Encrypt: $C = AES_K(M)$ (100 MB)

Break C into chunks $C_1, C_2, \ldots, C_\ell$ (each t byte
$\ell = 100\text{ MB}/t$

Let $C' = C_k = \boxed{\;|\;|\;|\;|\;|\;|\;|\;|\;}$ t bytes
$a_0^{(k)} a_1^{(k)} \cdots\cdots a_{t-1}^{(k)}$ over $GF(2^8)$

$$f_k(x) = a_0^{(k)} + a_1^{(k)}x + \cdots + a_{t-1}^{(k)}x^{t-1}$$ [all bytes used as coefs — not random]

Give party i share $f_k(i)$ of k^{th} chunk
← 1 byte long, chunk has size t bytes.

Total size of party i's share = 128 bits (for s_i)
+ 10 bits (for i)
+ $100\text{MB}/t$ for $f_1(i), \ldots, f_\ell(i)$

$\approx 100\text{ MB}/t$

Reconstruction: t parties
reconstruct $s = K$
reconstruct $f_k(x)$ for $k = 1, 2, \ldots, \ell$
$\Rightarrow C$
decrypt $\Rightarrow M$

Thm: If encryption function (e.g. AES) is secure, then so is this secret sharing scheme. Even if adversary gets C, he gets no information on M. (aside from length)

Threshold crypto (fully distributed)

- Can we avoid having secret s (in secret sharing of key) ever exist?
  - Dealing – make up shares s_i, s is implicitly generated; never explicitly exists
  - Reconstruction – don't reconstruct s instead, use shares ~~to pro~~ of key to produce shares of signature, then reconstruct signature

How about RSA? [Fouque/Stern paper]

- generate $PK = (n, e)$ in distributed manner
- no one party ever knows factorization $n = p \cdot q$
- yet each party has share d_i of decryption key
  yet p & q exist & have suitable properties (size, etc.)
- given message m, each party can compute share $\sigma(d_i, m)$ of signature
- signature shares can be combined in public manner to produce $\sigma(m)$