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
El Gamal Signatures

Public system parameters

\[ p \text{ prime} \]
\[ g \text{ generator} \]

Keygen:
\[ x \in \mathbb{Z}_{p-1} \]
\[ y = g^x \]
\[ \text{SK} = x \]
\[ \text{PK} = y \]

Sign (M):
\[ m = h(M) \]
\[ k \in \mathbb{Z}_{p-1} \]
\[ r = g^k \]
\[ s = \frac{m - rx}{k} \text{ (mod } p-1) \]
\[ \sigma(M) = (r, s) \]

Verify:
\[ \frac{y^s \cdot g^r}{g^m} \text{ (mod } p) \]

Correctness:
\[ g^{rx} \cdot y^s = g^m \text{ (mod } p) \]
\[ r \cdot x + k \cdot s = m \text{ (mod } p-1) \]
\[ s = \frac{m - r \cdot x}{k} \text{ (mod } p-1) \]

(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 = id for sig)

**Proof:** Let \( e \in \mathbb{Z}_{p-1} \)

\[
\begin{align*}
r &\leftarrow g^e \pmod{p} \\
s &\leftarrow -r \pmod{p-1}
\end{align*}
\]

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

\[
y^r r^s = g^m
\]

\[
g^{x r} (g^e)^{-r} = g^{-e^r} = g^{es} = g^m \quad \text{for } m = es \pmod{p-1},
\]

But: It is easy to fix.

**Modified El Gamal** (Pointchev/Stern 1996)

**sign(M):** \( k \in \mathbb{Z}_p^* \)

\[
r = g^k \pmod{p}
\]

\[
m = h(M || r)
\]

\[
s = \frac{m - rx}{k} \pmod{p-1}
\]

\[
\sigma(M) = (r, s)
\]

**Verify:** check \( 0 < r < p \)

check \( y^r r^s = g^m \) where \( m = h(M || r) \),
Thm: (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, \( |q| = 160 \text{ bits} \)
- \( p = nq+1 \) prime, \( |p| = 1024 \text{ bits} \)
- \( g_0 \) generates \( \mathbb{Z}_p^* \)
- \( g = g_0^q \) generates \( G_q \) - subgroup of \( \mathbb{Z}_p^* \) of order \( q \)

**Keygen:**
- \( x \in \mathbb{Z}_q \)
- \( y = g^x \pmod{p} \)
- \( |x| = 160 \text{ bits} \)
- \( |y| = 1024 \text{ bits} \)

**Sign(\( M \))**:
- \( k \in \mathbb{Z}_q^* \) \( (i.e. \ 1 \leq k < q) \)
- \( r = (g^k \pmod{p}) \pmod{q} \)
- \( m = h(M) \)
- \( s = (m + rx)/k \pmod{q} \)
- redo if \( r = 0 \) or \( s = 0 \)
- \( \sigma(M) = (r,s) \)
- \( |\sigma| = 320 \text{ bits} \)

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

Check \( y^{r/s} g^{m/s} \pmod{p} \pmod{q} = r \)
where \( m = h(M) \)

**Correctness**:
- \( g^{(r \times m)/s} = r \pmod{p} \pmod{q} \)
- \( g^k = r \pmod{p} \pmod{q} \)

Security proof works if we had done \( m = h(M||r) \), as before.
As it stands, existentially forgable for \( h = \text{identity} \).
**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" \( t \), \( 1 \leq t \leq n \).
- 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 some effect via shares]

**Example: Signing:**

1. \( s \) (dealer) \( \rightarrow \) splitter
   \[ \ldots \rightarrow S_1 \rightarrow S_2 \rightarrow S_n \]

2. "secret share"

\[ m \rightarrow [\bigcirc] \rightarrow S_s (m) \]

\[ \begin{align*}
  s_1 m & \rightarrow [\bigcirc] \\
  s_2 m & \rightarrow [\bigcirc] \\
  \vdots & \rightarrow [\bigcirc] \\
  s_t m & \rightarrow [\bigcirc] \\
  \text{combiner} & \rightarrow \frac{\sum s_i}{t} = S_s (m)
\end{align*} \]
Secret Sharing:

\( t = 1 \) : \( s_i = s \quad \forall i \leq n \)

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

\( s_n \) chosen so \( s_1 \oplus s_2 \oplus \ldots \oplus s_n = s \)

\( 1 < t < n ? ? \) :

\[ y = f(x) \quad \text{degree } t-1 \]

\[ y = a_0 x^{t-1} + a_1 x^{t-2} + \ldots + a_{t-1} + a_0 \]

(mod p)

\( t \) points \( (x_i, y_i) \) \( 1 \leq i \leq t \) 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 \( \mathbb{Z}_p \)

\( s_i = \text{pair } (i, y_i) : y_i = f(i) \) given

How to reconstruct??

\[
\begin{array}{c}
\text{Evaluation} \\
\text{dual is Interpolation} \\
t \text{ pt+value pairs} \\
\text{Interpolation} \\
\end{array}
\]
Interpolation

Given \( (x_i, y_i) \) \( 1 \leq i \leq t \)

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

where \( f_i(x) = \begin{cases} 1 & \text{at } x = x_i; \\ 0 & \text{for } x = x_j \neq x_i, 1 \leq j \leq t \end{cases} \)

is clearly ok

But \( f_i(x) = \frac{-\prod (x-x_j)}{\prod (x_i-x_j)} \)

\( = 0 \) or 1 at \( x_j \) as appropriate

\( x_j \neq x_i \), \( x_i = x \).

Evaluating \( f(0) \) to get \( y_0 = s \): simplify to

\[ s = f(0) = \sum_{i=1}^{t} y_i \cdot \frac{\prod (-x_j)}{\prod (x_i-x_j)} \]

Thm: Secret-sharing is information-theoretically secure

Adversary with < \( t \) shares has no information re s.

PF: Can create consistent polynomial of degree \( t-1 \) going through given points at any point \((0, s)\) \( 0 \leq s \leq p \).

Refs: Reed-Solomon codes, erasure codes, error-correction, information dispersal (Rabin)
Note: We have $0 \leq m < p$ 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. (byte)
Put still, each person gets 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 ciphertext, & share key
of ciphertext.

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

\(M = \text{message to be shared}\)  
\((100 \text{ MB})\)

\(k = \text{AES key} = S\)  
\((128 \text{ bits})\)

Share \(S = n \text{ shares } S_1, S_2, \ldots, S_n\)  
\(\text{s.t. any } t \text{ can reconstruct } S\)  
\((\text{info-theoretically secure})\)

Give \(i^{th}\) party \(S_i\), \(1 \leq i \leq n\). (and \(i\))

**Encrypt:**  
\[ C = \text{AES}_k(M) \]  
\((100 \text{ MB})\)

Break \(C\) into chunks \(C_1, C_2, \ldots, C_l\)  
\((\text{each } t \text{ byte})\)  
\(l = 100 \text{ MB} / t\)

Let \(C' = C_k = \begin{bmatrix} a_0^{(k)} & a_1^{(k)} & \cdots & a_{n-1}^{(k)} \end{bmatrix} \) \(t \) \(\text{bytes}\)

\(f_k(x) = a_0^{(k)} + a_1^{(k)} x + \cdots + a_{n-1}^{(k)} x^{n-1}\)  
\(\text{not random}\)  
\(\left[ \text{all bytes used as keys} \right]\)

Give party \(i\) share \(f_k(i)\) of \(k^{th}\) chunk  
\(\leq 1 \text{ byte long, chunk by size } t \text{ bytes}\)

**Total size of party \(i\)'s share =**  
\(128 \text{ bits (for } S_i)\)  
\(+ 10 \text{ bits (for } i)\)  
\(+ 100 \text{ MB} / t \text{ for } f_1(i), \ldots, f_l(i)\)

\(\approx 100 \text{ MB} / t\)

**Reconstruction:** \(t\) parties  

reconstruct \(S = k\)

reconstruct \(f_k(x)\) for \(k = 1, 2, \ldots, l\)

\(\Rightarrow C\)

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  
has 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 $x_i$ of key to produce shares of signature, then reconstruct signature

How about RSA?  [Fouque/Stern paper-]
- Generate $PK = (n, e)$ is distributed manner
  - No one path ever knows factorization $n = p \cdot q$
  - Yet each path has share $d_i$ of decryption key $d$ yet $p$ & $q$ exist & have suitable properties (size, etc.)
  - Given message $m$, each path can compute share $s_i (d_i, m)$ of signature
  - Signature shares can be combined in public manner to produce $s (m)$