COMPSCI 4CR3 - Applied Cryptography

Jake Doliskani

McMaster University

Digital Signatures

This lecture

- The principle of digital signatures
- Security services
- The RSA digital signature scheme
- The Elgamal digital signature scheme

Digital signature

A digital signature scheme consists of three algorithms:

- **Gen**: generates a key pair (k_{pr}, k_{pub}).
- **Sig**: takes a private key k_{pr} and a message x. Outputs a signature s.
- **Ver**: takes a public key k_{pub}, a signature s and a message x. Outputs a bit $b \in \{0, 1\}$.

For a signature scheme to be practical, all these algorithms must be efficient.

Digital signature

generate keys: $(k_{pr}, k_{pub}) \leftarrow \mathsf{Gen}$
publish the public key k_{pub}

$\xleftarrow{\quad k_{pub} \quad}$

sign message: $s = \mathsf{Sig}(k_{pr}, x)$

$\xleftarrow{\quad (x, s) \quad}$

verify signature:
$b = \mathsf{Ver}(x, s, k_{pub})$
accept if $b = 1$; reject if $b = 0$

Security Services (core)

1. Confidentiality
   - Information is kept secret from all but authorized parties.
2. Integrity
   - Messages have not been modified in transit.
3. Message Authentication
   - The sender of a message is authentic. An alternative term is data origin authentication.
4. Nonrepudiation
   - The sender of a message can not deny the creation of the message.

Security Services (other)

5. Identification
   - Establish and verify the identity of an entity, e.g., a person, a computer or a credit card.
6. Access control
   - Restrict access to the resources to privileged entities.
7. Availability
   - Assures that the electronic system is reliably available.
8. Auditing
   - Provide evidence about security-relevant activities, e.g., by keeping logs about certain events.

The RSA signature scheme

generate $d, (n, e)$
publish the public key (n, e)

$$\xleftarrow{\quad (n, e) \quad}$$

sign message: $s = x^d \bmod n$

$$\xleftarrow{\quad (x, s) \quad}$$

verify signature:
$y = s^e \bmod n$
$b = (y \stackrel{?}{=} x)$
accept if $b = 1$; reject if $b = 0$

Proof of correctness:

$$s^e = (x^d)^e = x^{de} = x \bmod n$$

The RSA signature scheme (example)

choose $p = 5, q = 17, n = pq = 85$
compute $\phi(n) = (5-1)(17-1) = 64$
choose $e = 9$
compute $d = e^{-1} = 57 \bmod 64$

$\xleftarrow{\quad (85, 9) \quad}$

$x = 6$
sign: $s = 6^{57} = 11 \bmod 85$

$\xleftarrow{\quad (6, 11) \quad}$

verify signature:
$y = 11^9 = 6 \bmod 85$
$b = (6 \stackrel{?}{=} 6) = 1$
accept.

Security

- Algorithmic attacks
  - attack the underlying RSA scheme by computing the private key d
- Existential Forgery
  - generate a valid signature for some message x

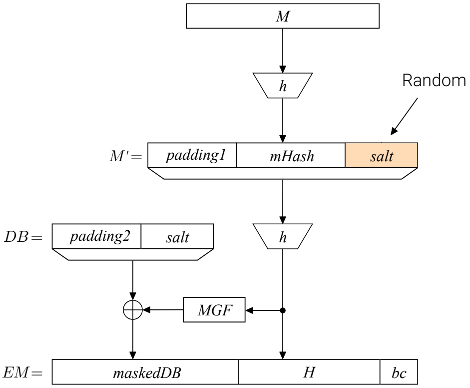
Existential Forgery: work backwards to generate a valid signature!

1. Choose a signature $s \in \mathbb{Z}_n$
2. Compute the message $x = s^e \bmod n$
3. The signature is (x, s)
4. The signature is valid since $x = s^e \bmod n$

RSA padding: The Probabilistic Signature Standard

- Pad the plain RSA to prevent the the existential forgery attack
- The padding method is in fact an encoding method called Encoding Method for Signature with Appendix (EMSA)
- EMSA is probabilistic!
- In practice, we sign the hash of the message instead of the message itself

The signature: $s = EM^d \bmod n$



The Elgamal signature scheme

Key generation:

1. Choose a large prime p
2. Choose a generator $\alpha \in \mathbb{Z}_p^\times$
   - We can also use a generator $\alpha \in G$ for some subgroup $G \leq \mathbb{Z}_p^\times$
3. Choose a random integer $d \in \{2, 3, \ldots, p-2\}$
4. Compute $\beta = \alpha^d \bmod p$

- Public parameters: (p, α)
- Public key: β
- Private key: d

The Elgamal signature scheme

Signature generation:
Input: x, d

1. Choose a random ephemeral key $k_E \in \{0, 1, \cdots, p-2\}$ such that $\gcd(k_E, p-1) = 1$
2. Compute $r = \alpha^{k_E} \bmod p$ and $s = (x - d \cdot r)k_E^{-1} \bmod p - 1$
3. Return (r, s)

Signature verification:
Input: $x, (r, s)$

1. Compute $t = \beta^r \cdot r^s \bmod p$
2. Return “invalid” if $t \neq \alpha^x \bmod p$; otherwise return “valid”

Correctness

If we rewrite $s = (x - d \cdot r)k_E^{-1} \bmod p - 1$, we get $x = d \cdot r + k_E s \bmod p - 1$

So, by Fermat's little theorem

$$\alpha^x = \alpha^{d \cdot r + k_E s} \bmod p.$$

On the other hand

$$\begin{aligned} \beta^r r^s &= (\alpha^d)^r (\alpha^{k_E})^s \bmod p \\ &= \alpha^{d \cdot r + k_E s} \bmod p. \end{aligned}$$

Therefore, if (r, s) is a valid signature,

$$\alpha^x = \beta^r r^s \bmod p. \qquad \square$$

The Elgamal signature scheme (example)

Public parameters:

- prime $p = 53$
- generator $27 \in \mathbb{Z}_{53}^{\times}$

choose $d = 25$
compute $\beta = \alpha^d = 51 \bmod 53$

$$\xleftarrow{\quad 51 \quad}$$

message: $x = 41$
choose $k_E = 19$
compute $r = \alpha^{k_E} = 31 \bmod 53$
compute $s = (x - d \cdot r)k_E^{-1} = 38 \bmod 52$

$$\xleftarrow{\quad (41, (31, 38)) \quad}$$

verify signature:
$t = \beta^r r^s = 34 \bmod 53$
$\alpha^x = 34 \bmod 53$
$t = \alpha^x \implies$ valid signature.

Security

- Computing Discrete Logarithms
- Reuse of the Ephemeral Key
- Existential Forgery Attack

Computing DLP:

- Trudy can obtain d, k_E from $\beta = \alpha^d$ and $r = \alpha^{k_E} \bmod p$
- He can sign arbitrary messages

Security

Reuse of the Ephemeral Key:

$$s_1 = (x_1 - d \cdot r)k_E^{-1} \bmod p-1$$
$$s_2 = (x_2 - d \cdot r)k_E^{-1} \bmod p-1$$
$$\Downarrow$$
$$s_1 k_E + d \cdot r = x_1 \bmod p-1$$
$$s_2 k_E + d \cdot r = x_2 \bmod p-1$$

May have multiple solutions,
Trudy has to find the correct one.

Trudy can compute d, k_E, and sign arbitrary messages.

Security

Existential Forgery Attack:

1. Select integers i, j such that $\gcd(j, p-1) = 1$
2. Compute $r = \alpha^i \beta^j \mod p$ and $s = -rj^{-1} \mod p-1$
3. Compute $x = si \mod p-1$
4. The message and signature are $x, (r, s)$

Countermeasure: hash the message:
$s = (h(x) - d \cdot r)k_E^{-1} \mod p-1$

Verification:

$$
\begin{aligned}
t &= \beta^r r^s \mod p \\
&= \alpha^{dr} \alpha^{(i+jd)s} \mod p \\
&= \alpha^{dr} \alpha^{(i+jd)(-rj^{-1})} \mod p \\
&= \alpha^{dr-dr} \alpha^{-rij^{-1}} \mod p \\
&= \alpha^{si} \mod p \\
&= \alpha^{x} \mod p
\end{aligned}
$$

The Digital Signature Algorithm (DSA)

Key Generation:

1. Generate a prime $2^{1023} < p < 2^{1024}$
2. Find a prime divisor q of $p-1$ such that $2^{159} < q < 2^{160}$
3. Find an element $\alpha \in \mathbb{Z}_p^\times$ of order q
4. Choose a random integer $0 < d < q$
5. Compute $\beta = \alpha^d \bmod p$

Other options:

p	q	signature
1024	160	320
2048	224	448
3072	256	512

- Public parameters: (p, q, α)
- Public key: β
- Private key: d

The DSA

Signature generation:
Input: x, d

1. Choose a random ephemeral key $k_E \in \{0, 1, \cdots, q-1\}$
2. Compute $r = (\alpha^{k_E} \bmod p) \bmod q$ and $s = (h(x) - d \cdot r)k_E^{-1} \bmod q$
3. Return (r, s)

Signature verification:
Input: $x, (r, s)$

1. Compute $w = s^{-1} \bmod q$ and $u_1 = wh(x) \bmod q$ and $u_2 = wr \bmod q$
2. Compute $t = (\alpha^{u_1}\beta^{u_2} \bmod p) \bmod q$
3. Return "valid" if $v = r$; otherwise return "invalid"

Correctness

$$
\begin{aligned}
s = (h(x) - d \cdot r)k_E^{-1} \bmod q \quad &\Longrightarrow \quad k_E = s^{-1}h(x) + s^{-1}rd \bmod q \\
&\Longrightarrow \quad k_E = u_1 + u_2 d \bmod q \\
&\Longrightarrow \quad \alpha^{k_E} = \alpha^{u_1 + u_2 d} \bmod p \\
&\Longrightarrow \quad \alpha^{k_E} = \alpha^{u_1}\beta^{u_2} \bmod p \qquad \text{(since } \alpha^d = \beta\text{)} \\
\text{(reduce both sides mod } q\text{)} \quad &\Longrightarrow \quad (\alpha^{k_E} \bmod p) \bmod q = (\alpha^{u_1}\beta^{u_2} \bmod p) \bmod q \\
&\Longrightarrow \quad r = v \bmod q \qquad \square
\end{aligned}
$$

The DSA (example)

Public parameters:

- primes $p = 53, q = 13$
- generator $\alpha = 10$

choose $d = 8$
compute $\beta = \alpha^d = 24 \bmod 53$

$\xleftarrow{\quad 24 \quad}$

message: $x = 41, h(x) = 6$
choose $k_E = 9$
compute $r = (\alpha^{k_E} \bmod p) \bmod q = 2$
compute $s = (h(x) - d \cdot r)k_E^{-1} = 1 \bmod 13$

$\xleftarrow{\quad (41, (2, 1)) \quad}$

verify signature:
$w = s^{-1} = 1 \bmod 13$
$(u_1, u_2) = (w \cdot h(x), wr) = (6, 2) \bmod 13$
$v = (\alpha^{u_1} \beta^{u_2} \bmod p) \bmod q = 2$
$v = r \implies$ valid signature

Prime generation for DSA

Generate primes p, q such that

- $2^{1023} < p < 2^{1024}$ and $2^{159} < q < 2^{160}$,
- $q \mid p - 1$

1. Find a prime $2^{159} < q < 2^{160}$ using the Miller-Rabin algorithm
2. For $i = 1$ to 4096

   2.1 Generate a random integer $2^{1023} < M < 2^{1024}$
   2.2 Compute $M_r = M \bmod 2q$
   2.3 Let $p = M - M_r + 1$
   2.4 If p is prime, then return (p, q)

3. Go to Step 1

- Each iteration of the For-loop selects a random number of the form $p = 2qk + 1$ in the range $(2^{2023}, 2^{1024})$ and tests whether it is a prime.
- Does not take too many trials