Topics in Algebra: Cryptography

Univ.-Prof. Dr. Goulnara ARZHANTSEVA

WS 2018

Digital Signature Scheme

To ensure the non-repudiation of data over an insecure channel:

Definition: Signature scheme is a 5-tuple $(\mathcal{P}, \mathcal{A}, \mathcal{K}, \mathcal{S}, \mathcal{V})$, satisfying:

- $\mathcal{P}$ is a finite set of possible messages;
- $\mathcal{A}$ is a finite set of possible signatures;
- $\mathcal{K}$, the keyspace, is a finite set of possible keys;
- $\mathcal{S} = \{\mathrm{sig}_k : k \in \mathcal{K}\}$ consists of polynomial signing algorithms $\mathrm{sig}_k : \mathcal{P} \to \mathcal{A}$;
- $\mathcal{V} = \{\mathrm{ver}_k : k \in \mathcal{K}\}$ consists of polynomial verification algorithms $\mathrm{ver}_k : \mathcal{P} \times \mathcal{A} \to \{\text{true, false}\}$;
- $\forall x \in \mathcal{P}, \forall y \in \mathcal{A}$: $\mathrm{ver}_k(x, y) = \begin{cases} \text{true,} & \text{if } y = \mathrm{sig}_k(x) \\ \text{false,} & \text{otherwise.} \end{cases}$

A pair (x, y) with $x \in \mathcal{P}, y \in \mathcal{A}$ is called a signed message.

Handwritten signature vs Digital signature

Usual Signature	Digital Signature
A part of the document	Transmitted and stored separately
Verified by comparison with the original	Anyone can verify, efficiently
A copy is distinguished from the original	A copy is identical to the original
Easy to forge	Computationally hard to forge
Efficient signing process	Efficient signing process

Public-key Cryptosystem vs Digital signature

Public-key cryptosystem	Digital Signature
Encrypt with E_k	Sign with D_k
Decrypt with D_k	Verify with E_k

Public-key Cryptosystem vs Digital signature

Public-key cryptosystem	Digital Signature
Encrypt with E_k	Sign with D_k
Decrypt with D_k	Verify with E_k

Mathematics: Is swapping of D_k and E_k a valid operation?

Practice: Is swapping of the corresponding primitives a valid operation?

Key management: not same keys for distinct applications.

RSA Digital signature

Definition: RSA Signature scheme is a 5-tuple $(\mathcal{P}, \mathcal{A}, \mathcal{K}, \mathcal{S}, \mathcal{V})$ such that:

$n = pq$, where p, q are primes, $\mathcal{P} = \mathcal{A} = \mathbb{Z}/n\mathbb{Z}$ and

$$\mathcal{K} = \{(n, p, q, d, e) \ : \ de = 1 \bmod \phi(n)\}$$

For $k = (n, p, q, d, e)$, we define

$$\mathrm{sig}_k(x) = x^d \bmod n \ \text{ and}$$

$$\mathrm{ver}_k(x, y) = \begin{cases} \text{true}, & \text{if } x = y^e \bmod n \\ \text{false}, & \text{otherwise.} \end{cases}$$

Public-key is (n, e) and private-key is (p, q, d).

Test questions

Question 15

1. The DSS requires that $\mathcal{S} = \{\mathrm{sig}_k : k \in \mathcal{K}\}$ consists of polynomial signing algorithms $\mathrm{sig}_k : \mathcal{P} \to \mathcal{A}$ but the RSA Signature scheme involves the exponentiation. Is there a contradiction?
2. The DSS defines $\mathrm{ver}_k(x, y) = \mathrm{true}$, if $y = \mathrm{sig}_k(x)$ but the RSA Signature scheme defines $\mathrm{sig}_k(x) = x^d \bmod n$ and $\mathrm{ver}_k(x, y) =$ true, if $x = y^e \bmod n$. Is there a contradiction?

Attacks on DSS and their goals

Attacks on DSS

Key-only: The attacker knows the public verification key, i.e ver_k.

Known message: The attacker knows some messages (not selected by him) and their keys.

Chosen message: The attacker knows some messages (selected by him) and their keys.

Attacks on DSS and their goals

Attacks on DSS

Key-only: The attacker knows the public verification key, i.e ver_k.

Known message: The attacker knows some messages (not selected by him) and their keys.

Chosen message: The attacker knows some messages (selected by him) and their keys.

Goals of attacks on DSS

Total break: The attacker determines Alice's private key, i.e. sig_k.

Selective forgery: With a non-negligible probability, the attacker creates a valid signature on a message chosen by someone else.

Existential forgery: Forge a signature for some message (without the ability to do this for any message).

Universal forgery: Forge signatures of any message.

DSS goal

DSS goal: strongest variant

The resistance against universal forgery under a chosen message attack.

RSA Signature scheme is not resistant

Choose an arbitrary signature $y \in \mathbb{Z}/n\mathbb{Z}$, then compute the message $x = y^e \bmod n$; Thus, y is a valid signature on the message x (because $y = x^d \bmod n$; note that d is private).

Attacks on DSS: Examples

Existential forgery using key-only attack is always possible: Choose an arbitrary signature y, then compute the message x given by $x := E_k(y)$.

To prevent existential forgery: message redundancy or hashing.

Attacks on DSS: Examples

Existential forgery using key-only attack is always possible: Choose an arbitrary signature y, then compute the message x given by $x := E_k(y)$.

To prevent existential forgery: message redundancy or hashing.

If the corresponding one-way function with trapdoor is multiplicative (e.g. in the RSA case: $(xy)^e = x^e \cdot y^e$), then the universal forgery under a chosen message attack is possible. Indeed, to sign x decompose it as $x = x_1 x_2$ with $x_1 \neq x \neq x_2$. Get the signatures y_i of x_i (this is possible as we are under a chosen message attack). Compute $(x, y) = (x, y_1 y_2)$.

Attacks on DSS: Examples

Existential forgery using key-only attack is always possible: Choose an arbitrary signature y, then compute the message x given by $x := E_k(y)$.

To prevent existential forgery: message redundancy or hashing.

If the corresponding one-way function with trapdoor is multiplicative (e.g. in the RSA case: $(xy)^e = x^e \cdot y^e$), then the universal forgery under a chosen message attack is possible. Indeed, to sign x decompose it as $x = x_1 x_2$ with $x_1 \neq x \neq x_2$. Get the signatures y_i of x_i (this is possible as we are under a chosen message attack). Compute $(x, y) = (x, y_1 y_2)$.

The RSA case (and the other one-way functions with trapdoor case): The signature has same length as the message.

DSS + Hashing = Hash-then-sign

Definition: DSS with hashing is a DSS 5-tuple $(\mathcal{P}, \mathcal{A}, \mathcal{K}, \mathcal{S}, \mathcal{V})$ such that:

- $\mathcal{P} = \{0,1\}^*$ and $\mathcal{A} = \{0,1\}^\ell$ for some $\ell \in \mathbb{N}$;
- $h\colon \mathcal{P} \to \mathcal{A}$ a public hash function given by a polynomial algorithm;
- $\mathrm{sig}_k(x) = f_k^{-1}(h(x))$, where $f_k\colon \mathcal{A} \to \mathcal{A}$ is a one-way function with trapdoor.
- $\forall x \in \mathcal{P}, \forall y \in \mathcal{A}$: $\mathrm{ver}_k(x,y) = \begin{cases} \text{true}, & \text{if } f_k(y) = h(x) \\ \text{false}, & \text{otherwise.} \end{cases}$

To avoid the attacks h must be a one-way non-multiplicative function.

h is collision resistant if it is infeasible to find $x_1 \neq x_2$ with $h(x_1) = h(x_2)$.

Hash algorithm in practice: Example

The SHA = Secure Hash Algorithms are cryptographic hash functions published by the National Institute of Standards and Technology (NIST) as a U.S. Federal Information Processing Standard (FIPS).

In 2017 CWI Amsterdam and Google announced they had performed a collision attack against SHA-1. Since 2017 Microsoft, Google, Apple and Mozilla have all announced that their respective browsers stop accepting SHA-1 SSL certificates.

Collisions allow two files to produce the same signature, so a signature may appear valid even though that file was never actually signed.

Current use: SHA-2, Future: SHA-3 (both have various specifications).

Hash algorithm in practice: Concept

A stream cipher: one bit or byte at a time (e.g. Caesar, Vernam).
A block cipher: blocks of bits at a time (e.g. Vigenère, Feistel)

Symmetric key algorithms: DES'1975 (64-bits blocks), 3DES=TDES'1998 (64-bits blocks), AES'2000 (128-bits blocks)

Hash algorithm in practice: Concept

A stream cipher: one bit or byte at a time (e.g. Caesar, Vernam).
A block cipher: blocks of bits at a time (e.g. Vigenère, Feistel)

Symmetric key algorithms: DES'1975 (64-bits blocks), 3DES=TDES'1998 (64-bits blocks), AES'2000 (128-bits blocks)

Definition: Hash functions from block ciphers

Let $\mathcal{P} = \mathcal{K} = \mathcal{C} = \{0,1\}^\ell$ for some $\ell \in \mathbb{N}$ and E be a block cipher:

$$E\colon \mathcal{P} \times \mathcal{K} \to \mathcal{C}, (x, e) \mapsto E_e(x).$$

Define $h(x_1, \ldots, x_r) \in \{0,1\}^\ell$ with $x_i \in \{0,1\}^\ell$ recursively, by $h(\emptyset) = 0$, and

$$h(x_1, \ldots, x_r) = E_{e_h}(x_r) + e_h, \text{ where } e_h = h(x_1, \ldots, x_{r-1}).$$

Hash algorithm in practice: Concept

A stream cipher: one bit or byte at a time (e.g. Caesar, Vernam).
A block cipher: blocks of bits at a time (e.g. Vigenère, Feistel)

Symmetric key algorithms: DES'1975 (64-bits blocks), 3DES=TDES'1998 (64-bits blocks), AES'2000 (128-bits blocks)

Definition: Hash functions from block ciphers

Let $\mathcal{P} = \mathcal{K} = \mathcal{C} = \{0,1\}^\ell$ for some $\ell \in \mathbb{N}$ and E be a block cipher:

$$E\colon \mathcal{P} \times \mathcal{K} \to \mathcal{C}, (x, e) \mapsto E_e(x).$$

Define $h(x_1, \ldots, x_r) \in \{0,1\}^\ell$ with $x_i \in \{0,1\}^\ell$ recursively, by $h(\emptyset) = 0$, and

$$h(x_1, \ldots, x_r) = E_{e_h}(x_r) + e_h, \text{ where } e_h = h(x_1, \ldots, x_{r-1}).$$

SHA-1: $\{0,1\}^* \to \{0,1\}^{160}$ is such example.

DSS + Public-key cryptosystem

Alice sends a signed encrypted message to Bob

1. Given $x \in \mathcal{P}$, she computes her signature $y = \mathrm{sig}_{d_{\mathrm{Alice}}}(x)$.
2. She encrypts both x and y using Bob's public key $z = E_{e_{\mathrm{Bob}}}(x, y)$.
3. She sends z to Bob, who decrypts it $D_{d_{\mathrm{Bob}}}(z) = (x, y)$.
4. He uses her public verification function to check whether $\mathrm{ver}_{e_{\mathrm{Alice}}}(x, y) = \mathrm{true}$.

First signed, then encrypted.

DSS + Public-key cryptosystem

Alice sends a signed encrypted message to Bob

1. Given $x \in \mathcal{P}$, she computes her signature $y = \mathrm{sig}_{d_{\mathrm{Alice}}}(x)$.
2. She encrypts both x and y using Bob's public key $z = E_{e_{\mathrm{Bob}}}(x, y)$.
3. She sends z to Bob, who decrypts it $D_{d_{\mathrm{Bob}}}(z) = (x, y)$.
4. He uses her public verification function to check whether $\mathrm{ver}_{e_{\mathrm{Alice}}}(x, y) = \mathrm{true}$.

First signed, then encrypted.

Question 16

What if in the DSS + Public-key cryptosystem scheme we inverse the order of operations: what if Alice first encrypts x, and then signs the result?

ElGamal variant of Digital Signature

Definition: ElGamal Signature scheme

Let p be a prime and g a primitive element mod p.
Let $\mathcal{P} = (\mathbb{Z}/p\mathbb{Z})^\times$, $\mathcal{A} = (\mathbb{Z}/p\mathbb{Z})^\times \times (\mathbb{Z}/(p-1)\mathbb{Z})$ and define

$$\mathcal{K} = \{(p, g, d, y) : y = g^d \bmod p\}.$$

For $k = (p, g, d, y)$, and for a secrete random $r \in (\mathbb{Z}/(p-1)\mathbb{Z})^\times$, define

$$\mathrm{sig}_k(x; r) = (y_1, y_2), \text{ where}$$

$$y_1 = g^r \bmod p, \quad \text{and} \quad y_2 = (x - dy_1)r^{-1} \bmod p - 1.$$

For $x, y_1 \in (\mathbb{Z}/p\mathbb{Z})^\times$ and $y_2 \in \mathbb{Z}/(p-1)\mathbb{Z}$, define

$$\mathrm{ver}_k(x, (y_1, y_2)) = \text{true} \Leftrightarrow y^{y_1}(y_1)^{y_2} \equiv g^x \bmod p$$

Public key is (p, g, y) and private key is d.

ElGamal variant of Digital Signature

Verification step: a signature will be accepted by the verifier

$$y^{y_1}(y_1)^{y_2} \equiv (g^d)^{y_1} g^{r(x-dy_1)r^{-1}} \bmod p \equiv g^x \bmod p$$

Reminder (a consequence of Fermat's little theorem):
Since g is primitive mod$p$ it has order $p-1$.
Therefore, $g^{a-b} \equiv 1 \bmod p \Leftrightarrow a \equiv b \bmod p-1$.

This verification can be done using only public information.

ElGamal variant of Digital Signature

Security assumptions: computationally hard to forge a signature

To forge a signature of a given message x without knowing d an attacker chooses an arbitrary y_1 and then tries to find y_2:

$$y_2 \equiv \log_{y_1} g^x y^{-y_1} \bmod p,$$

so he must solve the DLP in $(\mathbb{Z}/p\mathbb{Z})^\times$.

ElGamal variant of Digital Signature

Security assumptions: computationally hard to forge a signature

To forge a signature of a given message x without knowing d an attacker chooses an arbitrary y_1 and then tries to find y_2:

$$y_2 \equiv \log_{y_1} g^x y^{-y_1} \bmod p,$$

so he must solve the DLP in $(\mathbb{Z}/p\mathbb{Z})^\times$.

Alternatively, he chooses an arbitrary y_2 and then tries to find y_1:

$$y^{y_1} y_1^{y_2} \equiv g^x \bmod p,$$

so he must solve this equation with the unknown y_1.

ElGamal variant of Digital Signature

Security assumptions: computationally hard to forge a signature

To forge a signature of a given message x without knowing d an attacker chooses an arbitrary y_1 and then tries to find y_2:

$$y_2 \equiv \log_{y_1} g^x y^{-y_1} \bmod p,$$

so he must solve the DLP in $(\mathbb{Z}/p\mathbb{Z})^\times$.

Alternatively, he chooses an arbitrary y_2 and then tries to find y_1:

$$y^{y_1} y_1^{y_2} \equiv g^x \bmod p,$$

so he must solve this equation with the unknown y_1.

Assumption: Both problems $\notin$ BPP

ElGamal signature scheme: Example of misuse

Proposition: same key twice

The total break holds whenever the same key is used at least twice.

Proof: Let (y_1, y_2) a signature of x_1 and (y_1, z_2) a signature of x_2. Then

$y^{y_1} y_1^{y_2} \equiv g^{x_1} \bmod p, \quad y^{y_1} y_1^{z_2} \equiv g^{x_2} \bmod p,$ thus, $g^{x_1 - x_2} \equiv y_1^{y_2 - z_2} \bmod p.$

Since $y_1 = g^r$, we have an equation with the unknown r:
$g^{x_1 - x_2} \equiv g^{r(y_2 - z_2)} \bmod p$, which is equivalent (see the Reminder), to

$$x_1 - x_2 \equiv r(y_2 - z_2) \bmod p - 1.$$

ElGamal signature scheme: Example of misuse

We want to solve: $x_1 - x_2 \equiv r(y_2 - z_2) \bmod p - 1$.

Let $s = \gcd(y_2 - z_2, p - 1)$. Then $s \mid (x_1 - x_2)$ and we define

$$x' = \frac{x_1 - x_2}{s},\ y' = \frac{y_2 - z_2}{s},\ p' = \frac{p - 1}{s}.$$

ElGamal signature scheme: Example of misuse

We want to solve: $x_1 - x_2 \equiv r(y_2 - z_2) \bmod p - 1$.
Let $s = \gcd(y_2 - z_2, p - 1)$. Then $s \mid (x_1 - x_2)$ and we define

$$x' = \frac{x_1 - x_2}{s},\ y' = \frac{y_2 - z_2}{s},\ p' = \frac{p - 1}{s}.$$

Then the equation becomes: $x' \equiv ry' \bmod p'$. Since $\gcd(y', p') = 1$, we compute $z' = (y')^{-1} \bmod p'$. Then $r = x'z' \bmod p'$. This gives s candidates for r:

$$r = x'z' + ip' \bmod p - 1,\ \text{for } 0 \leqslant i \leqslant s - 1.$$

ElGamal signature scheme: Example of misuse

We want to solve: $x_1 - x_2 \equiv r(y_2 - z_2) \bmod p - 1$.
Let $s = \gcd(y_2 - z_2, p - 1)$. Then $s \mid (x_1 - x_2)$ and we define

$$x' = \frac{x_1 - x_2}{s},\ y' = \frac{y_2 - z_2}{s},\ p' = \frac{p - 1}{s}.$$

Then the equation becomes: $x' \equiv ry' \bmod p'$. Since $\gcd(y', p') = 1$, we compute $z' = (y')^{-1} \bmod p'$. Then $r = x'z' \bmod p'$. This gives s candidates for r:

$$r = x'z' + ip' \bmod p - 1,\ \text{for } 0 \leqslant i \leqslant s - 1.$$

We determine the unique correct value by testing the condition $y_1 \equiv g^r \bmod p$.
Now that r is known, the attacker can compute d. Indeed, $\gcd(y_1, p - 1) = 1$, then

$$d = (x - ry_2)(y_1)^{-1} \bmod p - 1$$

EC variant of Digital Signature

ElGamal Signature Scheme: a suitable signature scheme, not just use of the ElGamal cryptosystem in the DSS.

Schnorr Signature Scheme: ElGamal Signature in a subgroup of size q of $(\mathbb{Z}/p\mathbb{Z})^\times$ (DLP in a subgroup $\notin$ BPP) and the hashing is integrated in the signing (opposite to the hash-and-sign).

Digital Signature Algorithm (DSA): Schnorr Signature Scheme + hash-and-sign (SHA-1)

ECDSA: EC variant of the DSA

ECDSA: EC variant of Digital Signature

p prime, $\mathbf{k} = (\mathbb{Z}/p\mathbb{Z})^\times$, $E = E(\mathbf{k})$, $P \in E$ of prime order q.

Definition: ECDSA, hash-and-sign

$\mathcal{P} = \{0,1\}^*, \mathcal{A} = (\mathbb{Z}/q\mathbb{Z})^\times \times (\mathbb{Z}/q\mathbb{Z})^\times$ and

$$\mathcal{K} = \{(p, q, E, P, d, Q) \;:\; Q = dP\}, \text{ where } 0 \leqslant d \leqslant q-1.$$

For $k = (p, q, E, P, d, Q)$, and a secrete random r, $1 \leqslant r \leqslant q-1$, define

$$\mathrm{sig}_k(x, r) = (t, s), \text{ where } rP = (u, v) \text{ with}$$

$$\begin{aligned} t &= u \bmod q \\ s &= r^{-1}(h(x) + dt) \bmod q \end{aligned}$$

If either $t = 0$ or $s = 0$, a new random value of r is chosen.
The public key is (p, q, E, P, Q) and the private key is d.

ECDSA: EC variant of Digital Signature

p prime, $\mathbf{k} = (\mathbb{Z}/p\mathbb{Z})^\times$, $E = E(\mathbf{k})$, $P \in E$ of prime order q

Definition: ECDSA, verification

For $x \in \{0,1\}^*$ and $t, s \in (\mathbb{Z}/q\mathbb{Z})^\times$, we compute

$$\begin{aligned}
&w = s^{-1} \bmod q \\
&i = wh(x) \bmod q \\
&j = wt \bmod q \\
&(u, v) = iP + jQ \\
&\mathrm{ver}_k(x, (t, s)) = \text{true} \Leftrightarrow u \bmod q = t.
\end{aligned}$$

ElGamal Signature scheme versus ECDSA

$d = \log_P Q$ the discrete log: similar to $y = g^d \bmod p$ in the ElGamal SS

The order of P is a large prime q: similar to the order of a primitive g

Computation of rP: similar to computation of g^r in the ElGamal SS

Computation of t, the first coordinate of the elliptic curve point rP, $\bmod q$: similar to computation of $g^r \bmod p$ to get y_1, the first component of the signature (y_1, y_2)

s is computed from t, d, r, x: similar to computation of y_2 from y_1, d, r, x.

Test questions

Question 17

1. What if in the argument showing that the Existential forgery is always possible we first choose an arbitrary x and then compute the corresponding signature y?
2. Assume that the hash function is not collision-resistant. Is an existential forgery using a known message attack possible?

Test questions

Question 18

Why is the ElGamal signature scheme not just the use of the ElGamal cryptosysytem in the DSS? Compare with the RSA signature scheme.

Question 19

Does the ElGamal Signature scheme provide the authentication? Compare to the ECDSA.