Digital Signature Schemes (数字签名机制)

Sheng Zhong Yuan Zhang

Computer Science and Technology Department
Nanjing University

Outline

What is a digital signature?

The digital signature scheme is a public-key cryptographic primitive that protects the integrity or authenticity of received messages.

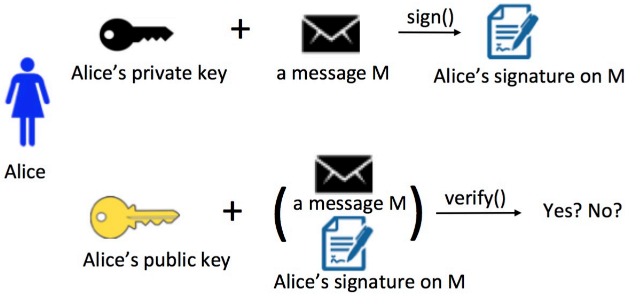

图 1: A digital signature allows the receiver to verify whether the message is sent (and signed) from Alice.

Comparison to Message Authentication Codes

Despite both digital signatures and MACs are both used to ensure the integrity of transmitted message, they have several differences:

- Digital signatures are publicly verifiable, while MACs are not.
- Digital signatures provide the important property of non-repudiation, while MACs do not.
- And all differences between private-key cryptosystems and public-key cryptosystems, e.g. regarding the key distribution and management, the efficiency...

The hash-and-sign paradigm

Digital signature schemes are generally less efficient than MACs. A possible way to mitigate this issue is to use the hash-and-sign paradigm.

$$\text{A long message } m \Rightarrow \text{Hash function } H(\cdot) \Rightarrow \mathit{Sign}(\cdot)$$

- The efficiency is good (cause hash operation is fast.).
- The security is also guaranteed given proper hash and signature scheme are used. (We will see this shortly.)

Formal definitions

DEFINITION 12.1

A (digitial) signature scheme consists of three PPT algorithms (*Gen*, *Sign*, *Vrfy*) such that:

- The **key-generation algorithm** *Gen* takes input a security parameter 1^n and outputs a pair of keys (pk, sk), where pk is called the **public key** and sk is called the **private key**.
- The **signing algorithm** *Sign* takes as input a private key sk and a message m from some message space (that may depend on pk). It outputs a signature σ, and we write this as $\sigma \leftarrow Sign_{sk}(m)$.
- The deterministic **verification algorithm** *Vrfy* takes as input a public key pk, a message m, and a signature σ. It outputs a bit b, with $b = 1$ meaning **valid** and $b = 0$ meaning **invalid**. We write $b := Vrfy_{pk}(m, \sigma)$.

It is required that except with negligible probability over (pk, sk) output by $Gen(1^n)$, it holds that $Vrfy_{pk}(m, Sign_{sk}(m)) = 1$ for every legal message m.

Security of signature schemes

We define the security of a signature scheme with the following experiment:

The signature experiment Sig-forge$_{\mathcal{A},\Pi}(n)$:

1. $Gen(1^n)$ is run to obtain keys (pk, sk).
2. Adversary $\mathcal{A}$ is given pk and access to an oracle $Sign_{sk}(\cdot)$. Let $\mathcal{Q}$ denote the set of all queries that $\mathcal{A}$ asked its oracle.
3. Then, $\mathcal{A}$ outputs (m, σ).
4. $\mathcal{A}$ **succeeds** if (1) $Vrfy_{pk}(m, \sigma) = 1$ and (2) $m \notin \mathcal{Q}$. In this case, the output of the experiment Sig-forge$_{\mathcal{A},\Pi}(n)$ is defined to 1; otherwise the output equals 0.

Security of signature schemes

With the signature experiment, we can define the security of a signature scheme as follows:

DEFINTION 12.2

A signature scheme $\Pi = (\mathit{Gen}, \mathit{Sign}, \mathit{Vrfy})$ is existentially unforgeable under an adaptive chosen-message attack, or just **secure**, if for all PPT adversary $\mathcal{A}$, there is a negligible function *negl* such that:

$$Pr[\text{Sig-forge}_{\mathcal{A},\Pi}(n) = 1] \leq \mathit{negl}(n).$$

- An attacker can do "existential forgery" if it can forge a signature for any message (even this message may be meaningless, and thus the attack is not harmful.).
- "adaptive" means the attacker can choose its target adaptively (based on its interactions with the oracle and challenger), and at the last minute of its attack.

Security of the hash-and-sign paradigm

We have the following result regarding the security of the hash-and-sign paradigm:

THEOREM 12.4

If Π is a secure signature scheme of length l and Π_H is a collision resistant hash function of length l. Then the hash-and-sign scheme constructed with Π and Π_H is a secure signature scheme for arbitrary-length messages.

Why?

Plain RSA signature scheme

A simple, RSA-based signature that we call "the plain RSA signature scheme" is as follows:

CONSTRUCTION 12.5: The plain RSA signature scheme

- *Gen*: on input 1^n run $GenRSA(1^n)$ to obtain (N, e, d). The public key is $\langle N, e\rangle$ and the private $\langle N, d\rangle$.
- *Sign*: on input a private key $sk = \langle N, d\rangle$ and a message $m \in \mathbb{Z}_N^*$, compute the signature

$$\sigma := [m^d \mod N].$$

- *Vrfy*: on input a public key $pk = \langle N, e\rangle$, a message $m \in \mathbb{Z}_N^*$ and a signature $\sigma \in \mathbb{Z}_N^*$, output 1 if and only if

$$m = [\sigma^e \mod N].$$

The underlying idea of the plain RSA signature

The underlying idea of the plain RSA signature is similar to a false idea that views digital signatures as the “inverse” of public-key encryption:

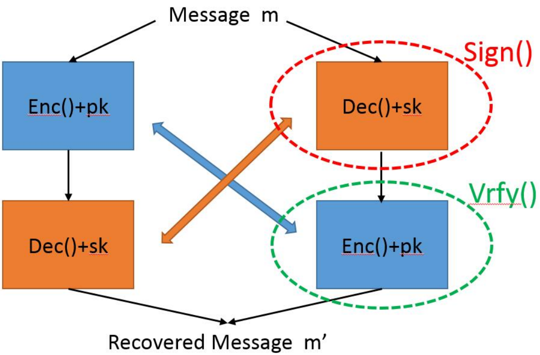

图 2: An incorrect view of the digital signature and public-key encryption

Why is this view incorrect?

Digital signatures are not the inverse of public-key encryption due to the following reasons:

- The construction as in Fig 1. may NOT function correctly:
  - $Dec()$ may be not applicable on m, $Enc()$ may be not well-defined for the input of a ciphertext.
  - Operations $Dec()$ and $Enc()$ may be not *commutative*(i.e. $Dec(Enc(x)) = Enc(Dec(x))$ holds.).
- More importantly, the construction as in Fig 1. is NOT secure.

Why is plain RSA signature insecure?

We can show the plain RSA signature is not secure under the following two attacks:

- Forging a signature without obtaining any signatures from legitimate signer (A no-message attack):

  A signature σ is valid $\Leftrightarrow \sigma^e = m$.

  To generate a valid signature, the adversary chooses an arbitrary $\hat{\sigma} \in \mathbb{Z}_N^*$, computes $\hat{m} = \hat{\sigma}^e$, and outputs $(\hat{m}, \hat{\sigma})$.

Q: Can you see what causes the vulnerability?

A: **Easy to invert**: m can be easily computed from its corresponding signature.

Why is plain RSA signature insecure?

- Forging a signature on an arbitrary message m by querying two signatures:

  Adversary chooses arbitrary messages $m_1, m_2 \in \mathbb{Z}_N^*$ such that $m_1 \cdot m_2 = m$.

  To generate a valid signature on m, the adversary uses the oracle to get m_1's signature σ_1 and m_2's signature σ_2.

  The adversary outputs $\sigma = \sigma_1 \cdot \sigma_2$ as the signature of m.

Q: Can you see what causes the vulnerability?

A: The plain RSA signature is malleable.

1. An Overview of Digital Signatures
2. RSA Signatures
   - Plain RSA signature (NOT SECURE)
   - RSA-FDH signature scheme
3. Signatures from the Discrete-Logarithm Problem
4. Certificates and Public-Key Infrastructure

To prevent previous attacks

To prevent previous attacks to plain RSA signature scheme, before signing, we can applying some transformation $H(\cdot)$ that satisfies the following requirements to messages:

- $H(\cdot)$ should be one-way or hard to invert.
- $H(\cdot)$ should be non-malleable.
- In addition, $H(\cdot)$ should be collision-resistant.

Based on above ideas, one can construct the **RSA-FDH signature scheme**.

The RSA-FDH signature scheme

Suppose H is some random function that can be modeled as a random oracle that maps its inputs uniformly onto $\mathbb{Z}_N^*$. Belows we construct the RSA full-domain hash (RSA-FDH) signature scheme.

- *Gen*: on input 1^n, run $GenRSA(1^n)$ to compute (N, e, d). The public key is $\langle N, e\rangle$ and the private key is $\langle N, d\rangle$. In addition, a random function $H : \{0,1\}^* \to \mathbb{Z}_N^*$ is specified.
- *Sign*: on input a private key $\langle N, d\rangle$ and a message $m \in \{0,1\}^*$, compute

$$\sigma := [H(m)^d \mod N].$$

- *Vrfy*: on input a public key $\langle N, e\rangle$, a message m, and a signature σ, output 1 if and only if

$$\sigma^e = H(m) \mod N.$$

- Regarding the security, we have the following theorem:

THEOREM 12.7

If the RSA problem is hard relative to *GenRSA* and H is modeled as a random oracle, then RSA-FDH signature is secure.

The proof of this theorem is not required in this course. See the textbook if you are interested.

- RSA PKCS #1 v2.1 includes a variant of RSA-FDH.

Discrete logarithm-based signatures

Examples of signature schemes from discrete-logarithm problem includes:

- The Schnorr signature scheme
- The DSA and ECDSA signature scheme.
- Schnorr, DSA, ECDSA are all constructed based on the identification scheme.

What is an identification scheme?

- The identification scheme is used to prove your identity to somebody.
- Here, we restrict our attention to public-key identification schemes.
- In public-key setting, the identification scheme allows you to prove to someone (called the verifier) that you are the one that corresponds to a specific public key.

  But how to prove this?
- We prove it by proving we know/have the private key that corresponds to the public key.

How to prove you know a private key?

Consider a "discrete logarithm style" public key $y = g^x$ (x is the private key) where $x \in \mathbb{Z}_q$ for an example. As the *prover*, we want to prove we know x.

Proposal 1

1. The prover sends x to the verifier.
2. The verifier verify whether

$$y = g^x.$$

Any problem here??

The private key is supposed to kept secret always (even to the verifier).

How to prove you know a private key?

So our problem becomes to prove you know x without revealing it to the verifier.

Proposal 2: the Schnorr identification scheme

1. The prover chooses a uniformly random $b \xleftarrow{\$} \mathbb{Z}_q$, computes
$$I = g^b$$
and sends I to the verifier.
2. The verifier generates a uniformly random $a \xleftarrow{\$} \mathbb{Z}_q$ (called a "*challenge*") and sends it to the prover.
3. In response, the prover sends $X = [ax + b \mod q]$ to the verifier.
4. The verifier accepts if and only if $y^a \cdot I = g^X$.

Do you see the difference between Proposals 1 and 2 here?
Instead of x, a random X = ax+b is revealed.

The correctness of Schnorr identification scheme

Schnorr identification scheme is essentially a proof system, the correctness of which requires two criteria:

- **Completeness**: "If a prover knows, it can pass the verification."
  It is easy to see
  $$y^a \cdot I = g^{ax} \cdot g^b = g^X$$
- **Soundness**: "If a prover does not know, it cannot pass."
  Why?

The correctness of Schnorr identification scheme

Specifically, assuming the discrete logarithm problem on $\mathbb{G} = \langle g \rangle$ is hard, if the prover passes the verification with high probability, it has to know the private key x.
Otherwise, y should be a random element in $\mathbb{G}$ for the prover, and we know

- The prover is able to compute correct response X_1, X_2 to at least two different challenge a_1, a_2 with a non-negligible probability. (Can you see why?)
- Then, the prover is able to solve $x = \log_g y$ by solving the following equations: $g^{a_1 x + b} = g^{X_1}, g^{a_2 x + b} = g^{X_2}$. This violates our assumption.

The security of Schnorr identification scheme

We claim the Schnorr identification scheme is **secure against an eavesdropper (or a passive attacker)**, or just **secure**, in the sense that no PPT eavesdropper can gain any additional knowledge about the private key by participating or eavesdropping the identification scheme.

- The claim can be proved by 1) constructing a *simulator* $\mathcal{S}$ that does NOT know x, but can simulate the view or transcript (i.e. the messages it sees) of an eavesdropper in the scheme.
- and 2) proving NO PPT adversaries can differentiate the simulated view and the real view.

The security of Schnorr identification scheme

How to construct $\mathcal{S}$?

- The real view is

$$(I = g^b, a, X = ax + b)$$

  given $a, b \xleftarrow{\$} \mathbb{Z}_q$.

- S samples $\tilde{a}, \tilde{X} \xleftarrow{\$} \mathbb{Z}_q$, and generates a simulated view as

$$(\tilde{I} = g^{\tilde{X}}/y^{\tilde{a}}, \tilde{a}, \tilde{X})$$

- The distributions of the above two views are the same.

From identification schemes to signatures

An identification scheme can be converted into a signature scheme using the following steps:

- The signer acts as a prover, runs the identification scheme by itself.
- It generates a challenge a by applying some random function H to I and m.
- It generates the correct response X and uses (a, X) as the signature of m.

The Fiat-Shamir transform

What we have done to the identification is actually a transform from an **interactive**, three-round identification proving scheme to a **non-interactive**, two-round proving scheme:

- The transform is called the Fiat-Shamir transform.
- Usually, H is chosen as a hash function that is modelled as a random oracle.
- It is widely used to construct non-interactive cryptographic protocols.

Why does this transformation results a secure signature scheme?

Informally, we know:

- The signature (a, X) is "bounded" to a specific message m since changing m would result in a completely different a (recall $a = H(I, m)$).
- The signature (a, X) is difficult to forge without knowing the private key (due to the security of the identification scheme).

The Schnorr signature scheme

Formally, we present the Schnorr signature scheme as follows.

CONSTRUCTION 12.12: the Schnorr signature scheme

Let $\mathcal{G}$ be the cyclic group generator that generates groups on which the discrete logarithm problem is hard,

- *Gen*: run $\mathcal{G}(1^n)$ to obtain $(\mathbb{G}, q, g)$. Choose a uniform $x \in \mathbb{Z}_q$, and compute $y = g^x$. The private key is x and the public key is $(\mathbb{G}, q, g, y)$. As part of key generation, a function $H : \{0,1\}^* \to \mathbb{Z}_q$ is specified, we leave this implicit.
- *Sign*: on input a private key x and a message $m \in \{0,1\}^*$, choose uniform $b \in \mathbb{Z}_q$ and set $I := g^b$. Then compute $a := H(I, m)$, followed by $X := [ax + b \mod q]$. Output the signature (a, X).
- *Vrfy*: on input a public key $(\mathbb{G}, q, g, y)$ and a message m, and a signature (a, X), compute $I' := g^X \cdot y^{-a}$, and output 1 if $H(I', m) = a$.

A pic that we have seen

- We saw this picture in our very first lecture:

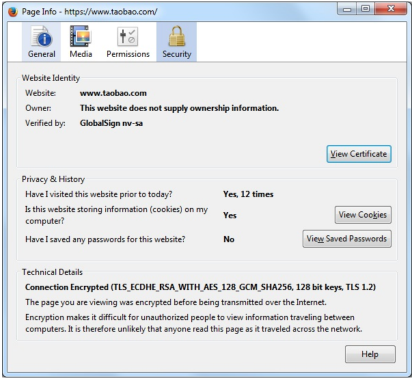

图 3: Page Info of www.taobao.com

- *"Website Identity Verified by GlobalSign nv-sa; Connection Encrypted (TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256, 128 bit keys, TLS 1.2)"*

Who is GlobalSign?

According to its wiki page,

- GlobalSign is a **certificate authority (CAs)** and provider of Identity Services.
- Founded in Belgium in 1996 and acquired in 2007 by GMO group in Japan (formerly GeoTrust Japan).
- By Jan. 2015, Globalsign was the 4th largest CA in the world according to the Netcraft survey.

Certificate Authorities and PKI

- CAs are entities who issue **certificates** to certify the ownership of a public key *pk* by the named entity or actually its dns names. E.g., *"$*$.tmall.com's public key is 12345678"*
- CA signs its assertion with its own public key so that others can verify it.
- Browsers and computer vendors trust a CA by preinstall its certificate which certificates the CA's public key.
- The above trust hierarchy is called the **Public-key Infrastructure (PKI)**.

References I

Katz, J. and Lindell, Y..
Chapter 11 of “Introduction to modern crytography” (2nd ed).
Chapman & Hall/CRC, 2014

The photot of Rivest, Shamir and Adleman in 1978 is downloaded from
http://www.usc.edu/dept/molecular-science/RSApics.htm

The photot of Rivest, Shamir and Adleman in 2003 is downloaded from
http://www.usc.edu/dept/molecular-science/RSA-2003.htm