Cryptology

Sabyasachi Karati

Assistant Professor
Cryptology and Security Research Unit (C.S.R.U)
R. C. Bose Centre for Cryptology and Security
Indian Statistical Institute (ISI)
Kolkata, India

Lecture 13

Digital Signatures

- Achieved private communication in the public-key setting.

Introduction

- Achieved private communication in the public-key setting.
- Q. How can we preserve message integrity?

Introduction

- Achieved private communication in the public-key setting.
- Q. How can we preserve message integrity?
- The public-key counterpart of MACs are digital signatures.

Introduction

- Achieved private communication in the public-key setting.
- Q. How can we preserve message integrity?
- The public-key counterpart of MACs are digital signatures.
- Signer uses secret key sk to create a signature σ on a message m.

Introduction

- Achieved private communication in the public-key setting.
- Q. How can we preserve message integrity?
- The public-key counterpart of MACs are digital signatures.
- Signer uses secret key sk to create a signature σ on a message m.
- Verifier uses signer's public key pk to verify the signature σ on a message m.

- Software Patch Release:
  - Company generates a key pair (sk, pk) and publishes pk.

Introduction

- Software Patch Release:
  - Company generates a key pair (sk, pk) and publishes pk.
  - Signs the patch m using sk, let the signature be σ.

- Software Patch Release:
  - Company generates a key pair (sk, pk) and publishes pk.
  - Signs the patch m using sk, let the signature be σ.
  - Client can verify the patch's (i) integrity and (ii) source authentication by verifying the σ using Company's public key pk.

Introduction

- Software Patch Release:
  - Company generates a key pair (sk, pk) and publishes pk.
  - Signs the patch m using sk, let the signature be σ.
  - Client can verify the patch's (i) integrity and (ii) source authentication by verifying the σ using Company's public key pk.
- Digital Certificate:
  - Assumption: public keys are obtained from a read-only public directory.

- Software Patch Release:
  - Company generates a key pair (sk, pk) and publishes pk.
  - Signs the patch m using sk, let the signature be σ.
  - Client can verify the patch's (i) integrity and (ii) source authentication by verifying the σ using Company's public key pk.
- Digital Certificate:
  - Assumption: public keys are obtained from a read-only public directory.
  - However, there is no public directory.

Introduction

- Software Patch Release:
  - Company generates a key pair (sk, pk) and publishes pk.
  - Signs the patch m using sk, let the signature be σ.
  - Client can verify the patch's (i) integrity and (ii) source authentication by verifying the σ using Company's public key pk.
- Digital Certificate:
  - Assumption: public keys are obtained from a read-only public directory.
  - However, there is no public directory.
    1. Alice creates (sk, pk).

- Software Patch Release:
  - Company generates a key pair (sk, pk) and publishes pk.
  - Signs the patch m using sk, let the signature be σ.
  - Client can verify the patch's (i) integrity and (ii) source authentication by verifying the σ using Company's public key pk.
- Digital Certificate:
  - Assumption: public keys are obtained from a read-only public directory.
  - However, there is no public directory.
    1. Alice creates (sk, pk).
    2. Sents pk to Certificate Authority (CA) whose pk_{CA} is known to everyone.

- Software Patch Release:
  - Company generates a key pair (sk, pk) and publishes pk.
  - Signs the patch m using sk, let the signature be σ.
  - Client can verify the patch's (i) integrity and (ii) source authentication by verifying the σ using Company's public key pk.
- Digital Certificate:
  - Assumption: public keys are obtained from a read-only public directory.
  - However, there is no public directory.
    1. Alice creates (sk, pk).
    2. Sents pk to Certificate Authority (CA) whose pk_{CA} is known to everyone.
    3. CA signs the message $m =$ "pk is public key of Alice" and creates certificate $Cert := (m, \sigma_{CA})$.

- Software Patch Release:
  - Company generates a key pair (sk, pk) and publishes pk.
  - Signs the patch m using sk, let the signature be σ.
  - Client can verify the patch's (i) integrity and (ii) source authentication by verifying the σ using Company's public key pk.
- Digital Certificate:
  - Assumption: public keys are obtained from a read-only public directory.
  - However, there is no public directory.
    1. Alice creates (sk, pk).
    2. Sents pk to Certificate Authority (CA) whose pk_{CA} is known to everyone.
    3. CA signs the message $m =$ "pk is public key of Alice" and creates certificate $Cert := (m, \sigma_{CA})$.
    4. CA Sends $Cert$ to Alice, and Alice can share $Cert$ with anyone.

- Software Patch Release:
  - Company generates a key pair (sk, pk) and publishes pk.
  - Signs the patch m using sk, let the signature be σ.
  - Client can verify the patch's (i) integrity and (ii) source authentication by verifying the σ using Company's public key pk.
- Digital Certificate:
  - Assumption: public keys are obtained from a read-only public directory.
  - However, there is no public directory.
    1. Alice creates (sk, pk).
    2. Sents pk to Certificate Authority (CA) whose pk_{CA} is known to everyone.
    3. CA signs the message $m =$ "pk is public key of Alice" and creates certificate $Cert := (m, \sigma_{CA})$.
    4. CA Sends $Cert$ to Alice, and Alice can share $Cert$ with anyone.
    5. Anyone including Alice canverify the $Cert$.

Digital Signature Vs. MAC

- Digital Signature is Publicly verifiable.

Digital Signature Vs. MAC

- Digital Signature is Publicly verifiable.
  - Let (m, σ) be a message-signature pair.

Digital Signature Vs. MAC

- Digital Signature is Publicly verifiable.
  - Let (m, σ) be a message-signature pair.
  - If it is a valid one, then it does not matter who verifies it.

Digital Signature Vs. MAC

- Digital Signature is Publicly verifiable.
  - Let (m, σ) be a message-signature pair.
  - If it is a valid one, then it does not matter who verifies it.
  - Can not be achieved by MAC.

Digital Signature Vs. MAC

- Digital Signature is Publicly verifiable.
  - Let (m, σ) be a message-signature pair.
  - If it is a valid one, then it does not matter who verifies it.
  - Can not be achieved by MAC.
  - Signer share different key's with different receivers.

Digital Signature Vs. MAC

- Digital Signature is Publicly verifiable.
  - Let (m, σ) be a message-signature pair.
  - If it is a valid one, then it does not matter who verifies it.
  - Can not be achieved by MAC.
  - Signer share different key's with different receivers.
  - Therefore, One can verify the MAC specific to him, but can not guarantee for a MAC received by others.

Digital Signature Vs. MAC

- Digital Signature is Publicly verifiable.
  - Let (m, σ) be a message-signature pair.
  - If it is a valid one, then it does not matter who verifies it.
  - Can not be achieved by MAC.
  - Signer share different key's with different receivers.
  - Therefore, One can verify the MAC specific to him, but can not guarantee for a MAC received by others.
- Public verifiability implies that signatures are transferable.

Digital Signature Vs. MAC

- Digital Signature is Publicly verifiable.
  - Let (m, σ) be a message-signature pair.
  - If it is a valid one, then it does not matter who verifies it.
  - Can not be achieved by MAC.
  - Signer share different key's with different receivers.
  - Therefore, One can verify the MAC specific to him, but can not guarantee for a MAC received by others.
- Public verifiability implies that signatures are transferable.
- Non-repudiation.

Digital Signature Vs. MAC

- Digital Signature is Publicly verifiable.
    - Let (m, σ) be a message-signature pair.
    - If it is a valid one, then it does not matter who verifies it.
    - Can not be achieved by MAC.
    - Signer share different key's with different receivers.
    - Therefore, One can verify the MAC specific to him, but can not guarantee for a MAC received by others.
- Public verifiability implies that signatures are transferable.
- Non-repudiation.
    - Assuming a signer S widely publicizes his public key in the first place.

Digital Signature Vs. MAC

- Digital Signature is Publicly verifiable.
  - Let (m, σ) be a message-signature pair.
  - If it is a valid one, then it does not matter who verifies it.
  - Can not be achieved by MAC.
  - Signer share different key's with different receivers.
  - Therefore, One can verify the MAC specific to him, but can not guarantee for a MAC received by others.
- Public verifiability implies that signatures are transferable.
- Non-repudiation.
  - Assuming a signer S widely publicizes his public key in the first place.
  - S signs a message he cannot later deny having done so.

Digital Signature Vs. MAC

- Digital Signature is Publicly verifiable.
  - Let (m, σ) be a message-signature pair.
  - If it is a valid one, then it does not matter who verifies it.
  - Can not be achieved by MAC.
  - Signer share different key's with different receivers.
  - Therefore, One can verify the MAC specific to him, but can not guarantee for a MAC received by others.
- Public verifiability implies that signatures are transferable.
- Non-repudiation.
  - Assuming a signer S widely publicizes his public key in the first place.
  - S signs a message he cannot later deny having done so.
  - Can not be achieved by MAC.

Digital Signature Vs. MAC

- Digital Signature is Publicly verifiable.
  - Let (m, σ) be a message-signature pair.
  - If it is a valid one, then it does not matter who verifies it.
  - Can not be achieved by MAC.
  - Signer share different key's with different receivers.
  - Therefore, One can verify the MAC specific to him, but can not guarantee for a MAC received by others.
- Public verifiability implies that signatures are transferable.
- Non-repudiation.
  - Assuming a signer S widely publicizes his public key in the first place.
  - S signs a message he cannot later deny having done so.
  - Can not be achieved by MAC.
  - MAC key is private.

Digital Signature Vs. MAC

- Digital Signature is Publicly verifiable.
  - Let (m, σ) be a message-signature pair.
  - If it is a valid one, then it does not matter who verifies it.
  - Can not be achieved by MAC.
  - Signer share different key's with different receivers.
  - Therefore, One can verify the MAC specific to him, but can not guarantee for a MAC received by others.
- Public verifiability implies that signatures are transferable.
- Non-repudiation.
  - Assuming a signer S widely publicizes his public key in the first place.
  - S signs a message he cannot later deny having done so.
  - Can not be achieved by MAC.
  - MAC key is private.
  - Even if verifier provides MAC key, there is no way to prove that is the key.

Definition

A signature scheme $\mathfrak{S} = (\mathcal{G}, \mathcal{S}, \mathcal{V})$ is a triple of efficient algorithms, $\mathcal{G}$, $\mathcal{S}$ and $\mathcal{V}$, where $\mathcal{G}$ is called a key generation algorithm, $\mathcal{S}$ is called a signing algorithm, and $\mathcal{V}$ is called a verification algorithm. Algorithm $\mathcal{S}$ is used to generate signatures and algorithm $\mathcal{V}$ is used to verify signatures.

Definition

A signature scheme $\mathfrak{S} = (\mathcal{G}, \mathcal{S}, \mathcal{V})$ is a triple of efficient algorithms, $\mathcal{G}$, $\mathcal{S}$ and $\mathcal{V}$, where $\mathcal{G}$ is called a key generation algorithm, $\mathcal{S}$ is called a signing algorithm, and $\mathcal{V}$ is called a verification algorithm. Algorithm $\mathcal{S}$ is used to generate signatures and algorithm $\mathcal{V}$ is used to verify signatures.

- $\mathcal{G}$ is a probabilistic poly-time algorithm that takes no input. It outputs a pair (sk, pk), where sk is called a secret signing key and pk is called a public verification key.

Definition

A signature scheme $\mathfrak{S} = (\mathcal{G}, \mathcal{S}, \mathcal{V})$ is a triple of efficient algorithms, $\mathcal{G}$, $\mathcal{S}$ and $\mathcal{V}$, where $\mathcal{G}$ is called a key generation algorithm, $\mathcal{S}$ is called a signing algorithm, and $\mathcal{V}$ is called a verification algorithm. Algorithm $\mathcal{S}$ is used to generate signatures and algorithm $\mathcal{V}$ is used to verify signatures.

- $\mathcal{G}$ is a probabilistic poly-time algorithm that takes no input. It outputs a pair (sk, pk), where sk is called a secret signing key and pk is called a public verification key.
- $\mathcal{S}$ is a probabilistic poly-time algorithm that is invoked as $\sigma \xleftarrow{R} \mathcal{S}(sk, m)$, where sk is a secret key (as output by $\mathcal{G}$) and m is a message. The algorithm outputs a signature σ.

Definition

A signature scheme $\mathfrak{S} = (\mathcal{G}, \mathcal{S}, \mathcal{V})$ is a triple of efficient algorithms, $\mathcal{G}$, $\mathcal{S}$ and $\mathcal{V}$, where $\mathcal{G}$ is called a key generation algorithm, $\mathcal{S}$ is called a signing algorithm, and $\mathcal{V}$ is called a verification algorithm. Algorithm $\mathcal{S}$ is used to generate signatures and algorithm $\mathcal{V}$ is used to verify signatures.

- $\mathcal{G}$ is a probabilistic poly-time algorithm that takes no input. It outputs a pair (sk, pk), where sk is called a secret signing key and pk is called a public verification key.
- $\mathcal{S}$ is a probabilistic poly-time algorithm that is invoked as $\sigma \xleftarrow{R} \mathcal{S}(sk, m)$, where sk is a secret key (as output by $\mathcal{G}$) and m is a message. The algorithm outputs a signature σ.
- $\mathcal{V}$ is a deterministic poly-time algorithm invoked as $\mathcal{V}(pk, m, \sigma)$. It outputs either `accept` or `reject`.

Definition (Cont.)

- We require that a signature generated by $\mathcal{S}$ is always accepted by $\mathcal{V}$. That is, for all (sk, pk) output by $\mathcal{G}$ and all messages m, we have

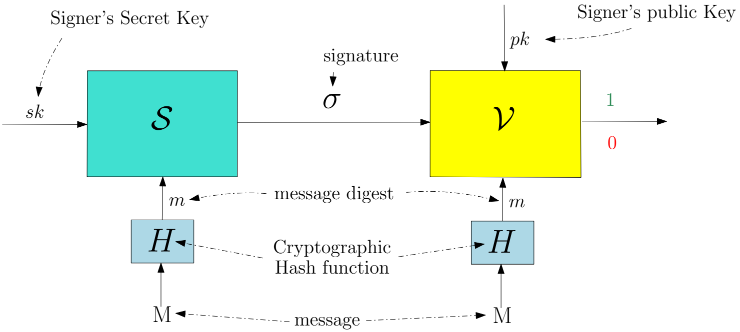
$$\Pr[V(pk, m, S(sk, m)) = \texttt{accept}] = 1.$$

Definition (Cont.)

- We require that a signature generated by $\mathcal{S}$ is always accepted by $\mathcal{V}$. That is, for all (sk, pk) output by $\mathcal{G}$ and all messages m, we have

$$\Pr[V(pk, m, S(sk, m)) = \texttt{accept}] = 1.$$

- Messages lie in a finite message space $\mathcal{M}$, and signatures lie in some finite signature space Σ. We say that $\mathfrak{S} = (\mathcal{G}, \mathcal{S}, \mathcal{V})$ is defined over $(\mathcal{M}, \Sigma)$.



Signer's Secret Key
sk
$\mathcal{S}$
signature
σ
Signer's public Key
pk
$\mathcal{V}$
1
0
m
message digest
m
H
Cryptographic
Hash function
H
M
message
M

Existentially Unforgeable under a Chosen Message Attack

A given MAC system $\mathfrak{S} = (\mathcal{G}, \mathcal{S}, \mathcal{V})$, defined over $(\mathcal{M}, \Sigma)$, and a given adversary $\mathcal{A}$, the attack game runs as follows:

Existentially Unforgeable under a Chosen Message Attack

A given MAC system $\mathfrak{S} = (\mathcal{G}, \mathcal{S}, \mathcal{V})$, defined over $(\mathcal{M}, \Sigma)$, and a given adversary $\mathcal{A}$, the attack game runs as follows:

1. The challenger runs $(sk, pk) \xleftarrow{R} \mathcal{G}$, and sends pk to adversary.

Existentially Unforgeable under a Chosen Message Attack

A given MAC system $\mathfrak{S} = (\mathcal{G}, \mathcal{S}, \mathcal{V})$, defined over $(\mathcal{M}, \Sigma)$, and a given adversary $\mathcal{A}$, the attack game runs as follows:

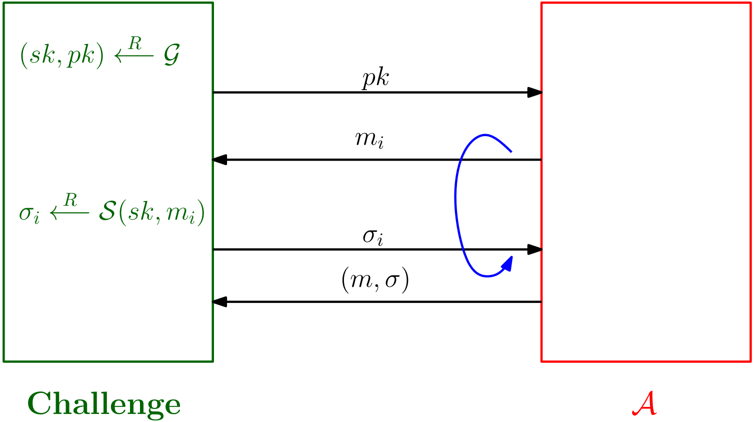
1. The challenger runs $(sk, pk) \xleftarrow{R} \mathcal{G}$, and sends pk to adversary.
2. $\mathcal{A}$ queries the challenger several times. For $i = 1, 2, \ldots$, the i-th signing query is a message $m_i \in \mathcal{M}$. Given m_i, the challenger computes a signature $\sigma_i \xleftarrow{R} \mathcal{S}(sk, m_i)$, and then gives σ_i to $\mathcal{A}$.

Existentially Unforgeable under a Chosen Message Attack

A given MAC system $\mathfrak{S} = (\mathcal{G}, \mathcal{S}, \mathcal{V})$, defined over $(\mathcal{M}, \Sigma)$, and a given adversary $\mathcal{A}$, the attack game runs as follows:

1. The challenger runs $(sk, pk) \xleftarrow{R} \mathcal{G}$, and sends pk to adversary.
2. $\mathcal{A}$ queries the challenger several times. For $i = 1, 2, \ldots$, the i-th signing query is a message $m_i \in \mathcal{M}$. Given m_i, the challenger computes a signature $\sigma_i \xleftarrow{R} \mathcal{S}(sk, m_i)$, and then gives σ_i to $\mathcal{A}$.
3. Eventually $\mathcal{A}$ outputs a candidate forgery pair $(m, \sigma) \in \mathcal{M} \times \Sigma$.

Security of Digital Signature



DS Attcak Game

Advantage of $\mathcal{A}$

- We say that the adversary wins the game if the following two conditions hold:
  - $\mathcal{V}(pk, m, \sigma) = \texttt{accept}$,

Advantage of $\mathcal{A}$

- We say that the adversary wins the game if the following two conditions hold:
  - $\mathcal{V}(pk, m, \sigma) = \texttt{accept}$, and
  - m is new, namely $m \notin \{m_1, m_2, \ldots\}$. (Extra compared to MAC.)

Advantage of $\mathcal{A}$

- We say that the adversary wins the game if the following two conditions hold:
  - $\mathcal{V}(pk, m, \sigma) = \texttt{accept}$, and
  - m is new, namely $m \notin \{m_1, m_2, \ldots\}$. (Extra compared to MAC.)
- In the adversary wins the Attack Game, the pair (m, σ) that it sends to the challenger is called an existential forgery.

Advantage of $\mathcal{A}$

- We say that the adversary wins the game if the following two conditions hold:
  - $\mathcal{V}(pk, m, \sigma) = \texttt{accept}$, and
  - m is new, namely $m \notin \{m_1, m_2, \ldots\}$. (Extra compared to MAC.)
- In the adversary wins the Attack Game, the pair (m, σ) that it sends to the challenger is called an existential forgery.
- We define $\mathcal{A}$'s advantage with respect to $\mathfrak{S}$, denoted $\mathsf{SIGadv}[\mathcal{A}, \mathfrak{S}]$, as the probability that $\mathcal{A}$ wins the game.

Advantage of $\mathcal{A}$

- We say that the adversary wins the game if the following two conditions hold:
  - $\mathcal{V}(pk, m, \sigma) = \texttt{accept}$, and
  - m is new, namely $m \notin \{m_1, m_2, \ldots\}$. (Extra compared to MAC.)
- In the adversary wins the Attack Game, the pair (m, σ) that it sends to the challenger is called an existential forgery.
- We define $\mathcal{A}$'s advantage with respect to $\mathfrak{S}$, denoted $\mathsf{SIGadv}[\mathcal{A}, \mathfrak{S}]$, as the probability that $\mathcal{A}$ wins the game.
- We say that $\mathcal{A}$ is a Q-query MAC adversary if $\mathcal{A}$ issues at most Q signing queries.

Security of Digital Signature

Advantage of $\mathcal{A}$

- We say that the adversary wins the game if the following two conditions hold:
  - $\mathcal{V}(pk, m, \sigma) = \texttt{accept}$, and
  - m is new, namely $m \notin \{m_1, m_2, \ldots\}$. (Extra compared to MAC.)
- In the adversary wins the Attack Game, the pair (m, σ) that it sends to the challenger is called an existential forgery.
- We define $\mathcal{A}$'s advantage with respect to $\mathfrak{S}$, denoted $\mathsf{SIGadv}[\mathcal{A}, \mathfrak{S}]$, as the probability that $\mathcal{A}$ wins the game.
- We say that $\mathcal{A}$ is a Q-query MAC adversary if $\mathcal{A}$ issues at most Q signing queries.

Secure Digital Signature

A digital signature system $\mathfrak{S}$ is secure if for all efficient adversaries $\mathcal{A}$, the value $\mathsf{SIGadv}[\mathcal{A}, \mathfrak{S}]$ is negligible.

Text-Book RSA

$\mathcal{G}$

1. Let the security parameter be n.

Text-Book RSA

$\mathcal{G}$

1. Let the security parameter be n.
2. Choose two primes p and q of bit-length almost n.

Text-Book RSA

$\mathcal{G}$

1. Let the security parameter be n.
2. Choose two primes p and q of bit-length almost n.
3. Compute $N = pq$ and $\phi(N) = (p-1)(q-1)$.

Text-Book RSA

$\mathcal{G}$

1. Let the security parameter be n.
2. Choose two primes p and q of bit-length almost n.
3. Compute $N = pq$ and $\phi(N) = (p-1)(q-1)$.
4. Choose e such that $\gcd(e, \phi(N)) = 1$.

Text-Book RSA

$\mathcal{G}$

1. Let the security parameter be n.
2. Choose two primes p and q of bit-length almost n.
3. Compute $N = pq$ and $\phi(N) = (p-1)(q-1)$.
4. Choose e such that $\gcd(e, \phi(N)) = 1$.
5. Compute $d \equiv e^{-1} \pmod{\phi}(N)$.

Text-Book RSA

$\mathcal{G}$

1. Let the security parameter be n.
2. Choose two primes p and q of bit-length almost n.
3. Compute $N = pq$ and $\phi(N) = (p-1)(q-1)$.
4. Choose e such that $\gcd(e, \phi(N)) = 1$.
5. Compute $d \equiv e^{-1} \pmod{\phi}(N)$.
6. $sk = (N, d)$ and $pk = (N, e)$.

Text-Book RSA

$\mathcal{G}$

1. Let the security parameter be n.
2. Choose two primes p and q of bit-length almost n.
3. Compute $N = pq$ and $\phi(N) = (p-1)(q-1)$.
4. Choose e such that $\gcd(e, \phi(N)) = 1$.
5. Compute $d \equiv e^{-1} \pmod{\phi}(N)$.
6. $sk = (N, d)$ and $pk = (N, e)$.

$\mathcal{S}(sk, m \in \mathbb{Z}_N)$

1. Compute $\sigma \longleftarrow m^d \pmod{N}$.

Text-Book RSA

$\mathcal{G}$

1. Let the security parameter be n.
2. Choose two primes p and q of bit-length almost n.
3. Compute $N = pq$ and $\phi(N) = (p-1)(q-1)$.
4. Choose e such that $\gcd(e, \phi(N)) = 1$.
5. Compute $d \equiv e^{-1} \pmod{\phi}(N)$.
6. $sk = (N, d)$ and $pk = (N, e)$.

$\mathcal{S}(sk, m \in \mathbb{Z}_N)$

1. Compute $\sigma \longleftarrow m^d \pmod{N}$.

$\mathcal{V}(pk, (m, \sigma) \in \mathbb{Z}_N \times \mathbb{Z}_N)$

1. Compute $m \stackrel{?}{=} \sigma^e \pmod{N}$.

A no-message attack

1. Based on the public key alone.

A no-message attack

1. Based on the public key alone.
2. Choose $\sigma \xleftarrow{R} \mathbb{Z}_n$.

A no-message attack

1. Based on the public key alone.
2. Choose $\sigma \xleftarrow{R} \mathbb{Z}_n$.
3. Set $m := \sigma^e \pmod{N}$.

Text-Book RSA: Attacks

A no-message attack

1. Based on the public key alone.
2. Choose $\sigma \xleftarrow{R} \mathbb{Z}_n$.
3. Set $m := \sigma^e \pmod{N}$.
4. Forgery: (m, σ) is a valid pair without any query.

Text-Book RSA: Attacks

A no-message attack

1. Based on the public key alone.
2. Choose $\sigma \xleftarrow{R} \mathbb{Z}_n$.
3. Set $m := \sigma^e \pmod{N}$.
4. Forgery: (m, σ) is a valid pair without any query.

Forging a signature on an arbitrary message

1. Obtain (m_1, σ_1) and (m_2, σ_2).

Text-Book RSA: Attacks

A no-message attack

1. Based on the public key alone.
2. Choose $\sigma \xleftarrow{R} \mathbb{Z}_n$.
3. Set $m := \sigma^e \pmod{N}$.
4. Forgery: (m, σ) is a valid pair without any query.

Forging a signature on an arbitrary message

1. Obtain (m_1, σ_1) and (m_2, σ_2).
2. Set $\sigma := \sigma_1 \cdot \sigma_2 \pmod{N}$ and $m := m_1 \cdot m_2 \pmod{N}$.

Text-Book RSA: Attacks

A no-message attack

1. Based on the public key alone.
2. Choose $\sigma \xleftarrow{R} \mathbb{Z}_n$.
3. Set $m := \sigma^e \pmod{N}$.
4. Forgery: (m, σ) is a valid pair without any query.

Forging a signature on an arbitrary message

1. Obtain (m_1, σ_1) and (m_2, σ_2).
2. Set $\sigma := \sigma_1 \cdot \sigma_2 \pmod{N}$ and $m := m_1 \cdot m_2 \pmod{N}$.
3. Forgery: (m, σ) is a valid pair.

Text-Book RSA: Attacks

A no-message attack

1. Based on the public key alone.
2. Choose $\sigma \xleftarrow{R} \mathbb{Z}_n$.
3. Set $m := \sigma^e \pmod{N}$.
4. Forgery: (m, σ) is a valid pair without any query.

Forging a signature on an arbitrary message

1. Obtain (m_1, σ_1) and (m_2, σ_2).
2. Set $\sigma := \sigma_1 \cdot \sigma_2 \pmod{N}$ and $m := m_1 \cdot m_2 \pmod{N}$.
3. Forgery: (m, σ) is a valid pair.

$$\sigma^e = (\sigma_1 \cdot \sigma_2)^e = \left(m_1^d \cdot m_2^d\right)^e = m_1^{ed} \cdot m_2^{ed} = m_1 \cdot m_2 = m \pmod{N}.$$

Hashed RSA

- Let $H : \{0,1\}^* \longrightarrow \mathbb{Z}_N$ be a collision resistant hash function.

$\mathcal{G}$

1. Let the security parameter be n.

Hashed RSA

- Let $H : \{0,1\}^* \longrightarrow \mathbb{Z}_N$ be a collision resistant hash function.

$\mathcal{G}$

1. Let the security parameter be n.
2. Choose two primes p and q of bit-length almost n.

Hashed RSA

- Let $H : \{0,1\}^* \longrightarrow \mathbb{Z}_N$ be a collision resistant hash function.

$\mathcal{G}$

1. Let the security parameter be n.
2. Choose two primes p and q of bit-length almost n.
3. Compute $N = pq$ and $\phi(N) = (p-1)(q-1)$.

Hashed RSA

- Let $H : \{0,1\}^* \longrightarrow \mathbb{Z}_N$ be a collision resistant hash function.

$\mathcal{G}$

1. Let the security parameter be n.
2. Choose two primes p and q of bit-length almost n.
3. Compute $N = pq$ and $\phi(N) = (p-1)(q-1)$.
4. Choose e such that $\gcd(e, \phi(N)) = 1$.

Hashed RSA

- Let $H : \{0,1\}^* \longrightarrow \mathbb{Z}_N$ be a collision resistant hash function.

$\mathcal{G}$

1. Let the security parameter be n.
2. Choose two primes p and q of bit-length almost n.
3. Compute $N = pq$ and $\phi(N) = (p-1)(q-1)$.
4. Choose e such that $\gcd(e, \phi(N)) = 1$.
5. Compute $d \equiv e^{-1} \pmod{\phi}(N)$.

Hashed RSA

- Let $H : \{0,1\}^* \longrightarrow \mathbb{Z}_N$ be a collision resistant hash function.

$\mathcal{G}$

1. Let the security parameter be n.
2. Choose two primes p and q of bit-length almost n.
3. Compute $N = pq$ and $\phi(N) = (p-1)(q-1)$.
4. Choose e such that $\gcd(e, \phi(N)) = 1$.
5. Compute $d \equiv e^{-1} \pmod{\phi}(N)$.
6. $sk = (N, d)$ and $pk = (N, e)$.

Hashed RSA

- Let $H : \{0,1\}^* \longrightarrow \mathbb{Z}_N$ be a collision resistant hash function.

$\mathcal{G}$

1. Let the security parameter be n.
2. Choose two primes p and q of bit-length almost n.
3. Compute $N = pq$ and $\phi(N) = (p-1)(q-1)$.
4. Choose e such that $\gcd(e, \phi(N)) = 1$.
5. Compute $d \equiv e^{-1} \pmod{\phi}(N)$.
6. $sk = (N, d)$ and $pk = (N, e)$.

$\mathcal{S}(sk, m)$

1. Compute $\sigma \longleftarrow H(m)^d \pmod{N}$.

Hashed RSA

- Let $H : \{0,1\}^* \longrightarrow \mathbb{Z}_N$ be a collision resistant hash function.

$\mathcal{G}$

1. Let the security parameter be n.
2. Choose two primes p and q of bit-length almost n.
3. Compute $N = pq$ and $\phi(N) = (p-1)(q-1)$.
4. Choose e such that $\gcd(e, \phi(N)) = 1$.
5. Compute $d \equiv e^{-1} \pmod{\phi}(N)$.
6. $sk = (N, d)$ and $pk = (N, e)$.

$\mathcal{S}(sk, m)$

1. Compute $\sigma \longleftarrow H(m)^d \pmod{N}$.

$\mathcal{V}(pk, (m, \sigma))$

1. Compute $H(m) \stackrel{?}{=} \sigma^e \pmod{N}$.

Hashed RSA

A no-message attack **not possible**

1. Based on the public key alone.

Hashed RSA

A no-message attack **not possible**

1. Based on the public key alone.
2. Choose $\sigma \xleftarrow{R} \mathbb{Z}_n$.

Hashed RSA

A no-message attack **not possible**

1. Based on the public key alone.
2. Choose $\sigma \xleftarrow{R} \mathbb{Z}_n$.
3. Set $m^* := \sigma^e \pmod{N}$.

Hashed RSA

A no-message attack **not possible**

1. Based on the public key alone.
2. Choose $\sigma \xleftarrow{R} \mathbb{Z}_n$.
3. Set $m^* := \sigma^e \pmod{N}$.
4. Compute m such that $m^* = H(m)$.

A no-message attack not possible

1. Based on the public key alone.
2. Choose $\sigma \xleftarrow{R} \mathbb{Z}_n$.
3. Set $m^* := \sigma^e \pmod{N}$.
4. Compute m such that $m^* = H(m)$.
5. H is collision Resistant.

Hashed RSA

A no-message attack **not possible**

1. Based on the public key alone.
2. Choose $\sigma \xleftarrow{R} \mathbb{Z}_n$.
3. Set $m^* := \sigma^e \pmod{N}$.
4. Compute m such that $m^* = H(m)$.
5. H is collision Resistant.

Forging a signature on an arbitrary message **not possible**

1. Obtain (m_1, σ_1) and (m_2, σ_2).

Hashed RSA

A no-message attack **not possible**

1. Based on the public key alone.
2. Choose $\sigma \xleftarrow{R} \mathbb{Z}_n$.
3. Set $m^* := \sigma^e \pmod{N}$.
4. Compute m such that $m^* = H(m)$.
5. H is collision Resistant.

Forging a signature on an arbitrary message **not possible**

1. Obtain (m_1, σ_1) and (m_2, σ_2).
2. Set $\sigma := \sigma_1 \cdot \sigma_2 \pmod{N}$.

Hashed RSA

A no-message attack **not possible**

1. Based on the public key alone.
2. Choose $\sigma \xleftarrow{R} \mathbb{Z}_n$.
3. Set $m^* := \sigma^e \pmod{N}$.
4. Compute m such that $m^* = H(m)$.
5. H is collision Resistant.

Forging a signature on an arbitrary message **not possible**

1. Obtain (m_1, σ_1) and (m_2, σ_2).
2. Set $\sigma := \sigma_1 \cdot \sigma_2 \pmod{N}$.
3. Find m such that $H(m) = H(m_1) \cdot H(m_2) \pmod{N}$.

Hashed RSA

A no-message attack not possible

1. Based on the public key alone.
2. Choose $\sigma \xleftarrow{R} \mathbb{Z}_n$.
3. Set $m^* := \sigma^e \pmod{N}$.
4. Compute m such that $m^* = H(m)$.
5. H is collision Resistant.

Forging a signature on an arbitrary message not possible

1. Obtain (m_1, σ_1) and (m_2, σ_2).
2. Set $\sigma := \sigma_1 \cdot \sigma_2 \pmod{N}$.
3. Find m such that $H(m) = H(m_1) \cdot H(m_2) \pmod{N}$.
4. Again, H is collision Resistant.

Hashed RSA

Security of hashed RSA

It is possible to prove the security-of hashed RSA in an idealized model where H is a truly random function.

- Out-of-the-scope of the course.

- We will study the following three digital signatures whose security relies on the hardness of discrete logarithm problem.
  1. ElGamal Digital Signature,
  2. The Schnorr Digital Signature Algorithm, and
  3. Digital Signature Algorithm (DSA).

DL based Digital Signature

- We will study the following three digital signatures whose security relies on the hardness of discrete logarithm problem.
  1. ElGamal Digital Signature,
  2. The Schnorr Digital Signature Algorithm, and
  3. Digital Signature Algorithm (DSA).
- But there is no known proof based on the discrete logarithm (or any other) assumption.

- We will study the following three digital signatures whose security relies on the hardness of discrete logarithm problem.
  1. ElGamal Digital Signature,
  2. The Schnorr Digital Signature Algorithm, and
  3. Digital Signature Algorithm (DSA).
- But there is no known proof based on the discrete logarithm (or any other) assumption.
- Let H be a collision resistant Hash function.

ElGamal Digital Signature

- ElGamal signature scheme was introduced by Tahir Elgamal in 1985.

ElGamal Digital Signature

- ElGamal signature scheme was introduced by Tahir Elgamal in 1985.

Domain parameter

- Let G be a cyclic multiplicative group.

ElGamal Digital Signature

- ElGamal signature scheme was introduced by Tahir Elgamal in 1985.

Domain parameter

- Let G be a cyclic multiplicative group.
- $O(G) = n$

ElGamal Digital Signature

- ElGamal signature scheme was introduced by Tahir Elgamal in 1985.

Domain parameter

- Let G be a cyclic multiplicative group.
- $O(G) = n$
- $\exists g \in G$ such that $O(g) = n$.

ElGamal Digital Signature

- ElGamal signature scheme was introduced by Tahir Elgamal in 1985.

Domain parameter

- Let G be a cyclic multiplicative group.
- $O(G) = n$
- $\exists g \in G$ such that $O(g) = n$.

$\mathcal{G}$

- Choose $d \xleftarrow{R} \mathbb{Z}_n \setminus \{0, 1\}$.
- Compute $Q \equiv g^d$.

ElGamal Digital Signature

- ElGamal signature scheme was introduced by Tahir Elgamal in 1985.

Domain parameter

- Let G be a cyclic multiplicative group.
- $O(G) = n$
- $\exists g \in G$ such that $O(g) = n$.

$\mathcal{G}$

- Choose $d \xleftarrow{R} \mathbb{Z}_n \setminus \{0, 1\}$.
- Compute $Q \equiv g^d$.
- $sk = (G, g, n, d)$ and $pk = (G, g, n, Q)$.

ElGamal Digital Signature

$\mathcal{S}(sk, M)$

1. Choose $k \xleftarrow{R} \mathbb{Z}_n \setminus \{0, 1\}$.

ElGamal Digital Signature

$\mathcal{S}(sk, M)$

1. Choose $k \xleftarrow{R} \mathbb{Z}_n \setminus \{0, 1\}$.
2. Compute $m = H(M) \in \mathbb{Z}_n$.

ElGamal Digital Signature

$\mathcal{S}(sk, M)$

1. Choose $k \xleftarrow{R} \mathbb{Z}_n \setminus \{0, 1\}$.
2. Compute $m = H(M) \in \mathbb{Z}_n$.
3. Compute $s = g^k$.
4. Compute $t \equiv k^{-1}(m - ds) \pmod{n}$.

ElGamal Digital Signature

$\mathcal{S}(sk, M)$

1. Choose $k \xleftarrow{R} \mathbb{Z}_n \setminus \{0, 1\}$.
2. Compute $m = H(M) \in \mathbb{Z}_n$.
3. Compute $s = g^k$.
4. Compute $t \equiv k^{-1}(m - ds) \pmod{n}$.
5. Signature $\sigma = (s, t)$

ElGamal Digital Signature

$\mathcal{S}(sk, M)$

1. Choose $k \xleftarrow{R} \mathbb{Z}_n \setminus \{0,1\}$.
2. Compute $m = H(M) \in \mathbb{Z}_n$.
3. Compute $s = g^k$.
4. Compute $t \equiv k^{-1}(m - ds) \pmod{n}$.
5. Signature $\sigma = (s,t)$

$\mathcal{V}(pk, (M, \sigma, pk))$

1. Compute $m = H(M) \in \mathbb{Z}_n$.

ElGamal Digital Signature

$\mathcal{S}(sk, M)$

1. Choose $k \xleftarrow{R} \mathbb{Z}_n \setminus \{0, 1\}$.
2. Compute $m = H(M) \in \mathbb{Z}_n$.
3. Compute $s = g^k$.
4. Compute $t \equiv k^{-1}(m - ds) \pmod{n}$.
5. Signature $\sigma = (s, t)$

$\mathcal{V}(pk, (M, \sigma, pk))$

1. Compute $m = H(M) \in \mathbb{Z}_n$.
2. Compute $a_1 = g^m$.
3. Compute $a_2 = Q^s s^t$.

ElGamal Digital Signature

$\mathcal{S}(sk, M)$

1. Choose $k \xleftarrow{R} \mathbb{Z}_n \setminus \{0, 1\}$.
2. Compute $m = H(M) \in \mathbb{Z}_n$.
3. Compute $s = g^k$.
4. Compute $t \equiv k^{-1}(m - ds) \pmod{n}$.
5. Signature $\sigma = (s, t)$

$\mathcal{V}(pk, (M, \sigma, pk))$

1. Compute $m = H(M) \in \mathbb{Z}_n$.
2. Compute $a_1 = g^m$.
3. Compute $a_2 = Q^s s^t$.
4. Check $a_1 \stackrel{?}{=} a_2$.
5. If yes Return `accept`, else Return `reject`.

Correctness

$$a_1 \equiv g^m \equiv g^{tk+ds} = (g^k)^t (g^d)^s \equiv s^t Q^s.$$

The Schnorr Digital Signature Algorithm

- It is a faster modification of ElGamal scheme.
- Introduced by Claus Schnorr in 1989.
- It is known for its efficiency and it generates short signatures

The Schnorr Digital Signature Algorithm

- It is a faster modification of ElGamal scheme.
- Introduced by Claus Schnorr in 1989.
- It is known for its efficiency and it generates short signatures

Domain parameter

- Let G be a cyclic multiplicative group.

The Schnorr Digital Signature Algorithm

- It is a faster modification of ElGamal scheme.
- Introduced by Claus Schnorr in 1989.
- It is known for its efficiency and it generates short signatures

Domain parameter

- Let G be a cyclic multiplicative group.
- $O(G) = n$.

The Schnorr Digital Signature Algorithm

- It is a faster modification of ElGamal scheme.
- Introduced by Claus Schnorr in 1989.
- It is known for its efficiency and it generates short signatures

Domain parameter

- Let G be a cyclic multiplicative group.
- $O(G) = n$.
- $\exists g \in G$ such that $O(g) = r$.

The Schnorr Digital Signature Algorithm

- It is a faster modification of ElGamal scheme.
- Introduced by Claus Schnorr in 1989.
- It is known for its efficiency and it generates short signatures

Domain parameter

- Let G be a cyclic multiplicative group.
- $O(G) = n$.
- $\exists g \in G$ such that $O(g) = r$.
- r is suitably large prime divisor of n.

The Schnorr Digital Signature Algorithm

- It is a faster modification of ElGamal scheme.
- Introduced by Claus Schnorr in 1989.
- It is known for its efficiency and it generates short signatures

Domain parameter

- Let G be a cyclic multiplicative group.
- $O(G) = n$.
- $\exists g \in G$ such that $O(g) = r$.
- r is suitably large prime divisor of n.

$\mathcal{G}$

- Choose $d \xleftarrow{R} \mathbb{Z}_r \setminus \{0, 1\}$.
- Compute $Q \equiv g^d$.

The Schnorr Digital Signature Algorithm

- It is a faster modification of ElGamal scheme.
- Introduced by Claus Schnorr in 1989.
- It is known for its efficiency and it generates short signatures

Domain parameter

- Let G be a cyclic multiplicative group.
- $O(G) = n$.
- $\exists g \in G$ such that $O(g) = r$.
- r is suitably large prime divisor of n.

$\mathcal{G}$

- Choose $d \xleftarrow{R} \mathbb{Z}_r \setminus \{0, 1\}$.
- Compute $Q \equiv g^d$.
- $sk = (G, g, n, r, d)$ and $pk = (G, g, n, r, Q)$

The Schnorr Digital Signature Algorithm

$S(sk, M)$

1. Choose session key (d', Q').
   - Choose $d' \xleftarrow{R} \mathbb{Z}_r \setminus \{0, 1\}$.
   - Compute $Q' = g^{d'}$.

The Schnorr Digital Signature Algorithm

$\mathcal{S}(sk, M)$

1. Choose session key (d', Q').
   - Choose $d' \xleftarrow{R} \mathbb{Z}_r \setminus \{0, 1\}$.
   - Compute $Q' = g^{d'}$.
2. Compute $s = H(M \| Q') \in \mathbb{Z}_r$.
3. Compute $t \equiv d' - ds \pmod{r}$.

The Schnorr Digital Signature Algorithm

$S(sk, M)$

1. Choose session key (d', Q').
   - Choose $d' \xleftarrow{R} \mathbb{Z}_r \setminus \{0, 1\}$.
   - Compute $Q' = g^{d'}$.
2. Compute $s = H(M \| Q') \in \mathbb{Z}_r$.
3. Compute $t \equiv d' - ds \pmod{r}$.
4. Signature $\sigma = (s, t)$.

The Schnorr Digital Signature Algorithm

$\mathcal{S}(sk, M)$

1. Choose session key (d', Q').
   - Choose $d' \xleftarrow{R} \mathbb{Z}_r \setminus \{0, 1\}$.
   - Compute $Q' = g^{d'}$.
2. Compute $s = H(M \| Q') \in \mathbb{Z}_r$.
3. Compute $t \equiv d' - ds \pmod{r}$.
4. Signature $\sigma = (s, t)$.

$\mathcal{V}(pk, (M, \sigma))$

1. Compute $u = g^t Q^s$.

The Schnorr Digital Signature Algorithm

$\mathcal{S}(sk, M)$

1. Choose session key (d', Q').
   - Choose $d' \xleftarrow{R} \mathbb{Z}_r \setminus \{0, 1\}$.
   - Compute $Q' = g^{d'}$.
2. Compute $s = H(M \| Q') \in \mathbb{Z}_r$.
3. Compute $t \equiv d' - ds \pmod{r}$.
4. Signature $\sigma = (s, t)$.

$\mathcal{V}(pk, (M, \sigma))$

1. Compute $u = g^t Q^s$.
2. Compute $\hat{s} = H(M \| u) \in \mathbb{Z}_r$.

The Schnorr Digital Signature Algorithm

$\mathcal{S}(sk, M)$

1. Choose session key (d', Q').
   - Choose $d' \xleftarrow{R} \mathbb{Z}_r \setminus \{0, 1\}$.
   - Compute $Q' = g^{d'}$.
2. Compute $s = H(M \| Q') \in \mathbb{Z}_r$.
3. Compute $t \equiv d' - ds \pmod{r}$.
4. Signature $\sigma = (s, t)$.

$\mathcal{V}(pk, (M, \sigma))$

1. Compute $u = g^t Q^s$.
2. Compute $\hat{s} = H(M \| u) \in \mathbb{Z}_r$.
3. Check $s \stackrel{?}{=} \hat{s}$.
4. If yes Return `accept`, else Return `reject`.

Correctness

$$u \equiv g^t Q^s \equiv g^t \left(g^d\right)^s \equiv g^{(t+ds)} \equiv g^{d'} \equiv Q'.$$

- National Institute of Standards and Technology (NIST) proposed DSA in 1991.

Domain parameter(p, q, g)

- Let $G = \mathbb{Z}_p^*$ for some prime p.

DSA Digital Signature

- National Institute of Standards and Technology (NIST) proposed DSA in 1991.

Domain parameter(p, q, g)

- Let $G = \mathbb{Z}_p^*$ for some prime p.
- Let $g \in G$ such that $\langle g \rangle$ is the largest prime subgroup of G.

DSA Digital Signature

- National Institute of Standards and Technology (NIST) proposed DSA in 1991.

Domain parameter(p, q, g)

- Let $G = \mathbb{Z}_p^*$ for some prime p.
- Let $g \in G$ such that $\langle g \rangle$ is the largest prime subgroup of G.
- Let $O(g) = q$.

DSA Digital Signature

- National Institute of Standards and Technology (NIST) proposed DSA in 1991.

Domain parameter(p, q, g)

- Let $G = \mathbb{Z}_p^*$ for some prime p.
- Let $g \in G$ such that $\langle g \rangle$ is the largest prime subgroup of G.
- Let $O(g) = q$.

$\mathcal{G}$

- Choose $d \xleftarrow{R} \mathbb{Z}_q \setminus \{0, 1\}$.

DSA Digital Signature

- National Institute of Standards and Technology (NIST) proposed DSA in 1991.

Domain parameter(p, q, g)

- Let $G = \mathbb{Z}_p^*$ for some prime p.
- Let $g \in G$ such that $\langle g \rangle$ is the largest prime subgroup of G.
- Let $O(g) = q$.

$\mathcal{G}$

- Choose $d \xleftarrow{R} \mathbb{Z}_q \setminus \{0, 1\}$.
- Compute $Q \equiv g^d \pmod{p}$.

DSA Digital Signature

- National Institute of Standards and Technology (NIST) proposed DSA in 1991.

Domain parameter(p, q, g)

- Let $G = \mathbb{Z}_p^*$ for some prime p.
- Let $g \in G$ such that $\langle g \rangle$ is the largest prime subgroup of G.
- Let $O(g) = q$.

$\mathcal{G}$

- Choose $d \xleftarrow{R} \mathbb{Z}_q \setminus \{0, 1\}$.
- Compute $Q \equiv g^d \pmod{p}$.
- $\mathcal{SK} = (G, g, p, q, d)$ and $\mathcal{PK} = (G, g, p, q, Q)$.

$S(sk, M)$

1. Choose $k \xleftarrow{R} \mathbb{Z}_q \setminus \{0, 1\}$.

DSA Digital Signature

$\mathcal{S}(sk, M)$

1. Choose $k \xleftarrow{R} \mathbb{Z}_q \setminus \{0,1\}$.
2. Compute $m = H(M) \in \mathbb{Z}_p$.

DSA Digital Signature

$S(sk, M)$

1. Choose $k \xleftarrow{R} \mathbb{Z}_q \setminus \{0, 1\}$.
2. Compute $m = H(M) \in \mathbb{Z}_p$.
3. Compute $s = \left(g^k \pmod{p}\right) \pmod{q}$.

DSA Digital Signature

$\mathcal{S}(sk, M)$

1. Choose $k \xleftarrow{R} \mathbb{Z}_q \setminus \{0, 1\}$.
2. Compute $m = H(M) \in \mathbb{Z}_p$.
3. Compute $s = \left(g^k \pmod{p}\right) \pmod{q}$.
4. Compute $t = k^{-1}(m + ds) \pmod{q}$.

DSA Digital Signature

$\mathcal{S}(sk, M)$

1. Choose $k \xleftarrow{R} \mathbb{Z}_q \setminus \{0, 1\}$.
2. Compute $m = H(M) \in \mathbb{Z}_p$.
3. Compute $s = \left(g^k \pmod{p}\right) \pmod{q}$.
4. Compute $t = k^{-1}(m + ds) \pmod{q}$.
5. Signature $\sigma = (s, t)$.

DSA Digital Signature

$\mathcal{S}(sk, M)$

1. Choose $k \xleftarrow{R} \mathbb{Z}_q \setminus \{0, 1\}$.
2. Compute $m = H(M) \in \mathbb{Z}_p$.
3. Compute $s = \left(g^k \pmod{p}\right) \pmod{q}$.
4. Compute $t = k^{-1}(m + ds) \pmod{q}$.
5. Signature $\sigma = (s, t)$.

$\mathcal{V}(pk, (M, \sigma))$

1. Compute $m = H(M) \in \mathbb{Z}_p$.

DSA Digital Signature

$\mathcal{S}(sk, M)$

1. Choose $k \xleftarrow{R} \mathbb{Z}_q \setminus \{0, 1\}$.
2. Compute $m = H(M) \in \mathbb{Z}_p$.
3. Compute $s = \left(g^k \pmod{p}\right) \pmod{q}$.
4. Compute $t = k^{-1}(m + ds) \pmod{q}$.
5. Signature $\sigma = (s, t)$.

$\mathcal{V}(pk, (M, \sigma))$

1. Compute $m = H(M) \in \mathbb{Z}_p$.
2. Compute $w \equiv t^{-1} \pmod{q}$.

DSA Digital Signature

$\mathcal{S}(sk, M)$

1. Choose $k \xleftarrow{R} \mathbb{Z}_q \setminus \{0, 1\}$.
2. Compute $m = H(M) \in \mathbb{Z}_p$.
3. Compute $s = \left(g^k \pmod p\right) \pmod q$.
4. Compute $t = k^{-1}(m + ds) \pmod q$.
5. Signature $\sigma = (s, t)$.

$\mathcal{V}(pk, (M, \sigma))$

1. Compute $m = H(M) \in \mathbb{Z}_p$.
2. Compute $w \equiv t^{-1} \pmod q$.
3. Compute $w_1 \equiv mw \pmod q$ and $w_2 \equiv sw \pmod q$.

DSA Digital Signature

$\mathcal{S}(sk, M)$

1. Choose $k \xleftarrow{R} \mathbb{Z}_q \setminus \{0, 1\}$.
2. Compute $m = H(M) \in \mathbb{Z}_p$.
3. Compute $s = \left(g^k \pmod{p}\right) \pmod{q}$.
4. Compute $t = k^{-1}(m + ds) \pmod{q}$.
5. Signature $\sigma = (s, t)$.

$\mathcal{V}(pk, (M, \sigma))$

1. Compute $m = H(M) \in \mathbb{Z}_p$.
2. Compute $w \equiv t^{-1} \pmod{q}$.
3. Compute $w_1 \equiv mw \pmod{q}$ and $w_2 \equiv sw \pmod{q}$.
4. Compute $s' = (g^{w_1} Q^{w_2} \pmod{p}) \pmod{q}$.

DSA Digital Signature

$\mathcal{S}(sk, M)$

1. Choose $k \xleftarrow{R} \mathbb{Z}_q \setminus \{0, 1\}$.
2. Compute $m = H(M) \in \mathbb{Z}_p$.
3. Compute $s = \left(g^k \pmod{p}\right) \pmod{q}$.
4. Compute $t = k^{-1}(m + ds) \pmod{q}$.
5. Signature $\sigma = (s, t)$.

$\mathcal{V}(pk, (M, \sigma))$

1. Compute $m = H(M) \in \mathbb{Z}_p$.
2. Compute $w \equiv t^{-1} \pmod{q}$.
3. Compute $w_1 \equiv mw \pmod{q}$ and $w_2 \equiv sw \pmod{q}$.
4. Compute $s' = (g^{w_1} Q^{w_2} \pmod{p}) \pmod{q}$.
5. Check $s' \stackrel{?}{=} s$; If yes Return `accept`, else Return `reject`.

Correctness

$$s' = g^{w_1} Q^{w_2}$$

Correctness

$$
\begin{aligned}
s' &= g^{w_1} Q^{w_2} \\
&= g^{w_1} g^{d w_2} \qquad \text{as } Q = g^d
\end{aligned}
$$

Correctness

$$
\begin{aligned}
s' &= g^{w_1} Q^{w_2} \\
&= g^{w_1} g^{d w_2} && \text{as } Q = g^d \\
&= g^{wm} g^{dsw} && \text{as } w_1 = mw \text{ and } w_2 = sw
\end{aligned}
$$

Correctness

$$\begin{aligned} s' &= g^{w_1} Q^{w_2} \\ &= g^{w_1} g^{d w_2} && \text{as } Q = g^d \\ &= g^{wm} g^{dsw} && \text{as } w_1 = mw \text{ and } w_2 = sw \\ &= g^{w(m+ds)} \end{aligned}$$

Correctness

$$
\begin{aligned}
s' &= g^{w_1} Q^{w_2} \\
&= g^{w_1} g^{d w_2} && \text{as } Q = g^d \\
&= g^{wm} g^{dsw} && \text{as } w_1 = mw \text{ and } w_2 = sw \\
&= g^{w(m+ds)} \\
&= g^{t^{-1}(m+ds)} && \text{as } w = t^{-1}
\end{aligned}
$$

DSA Digital Signature

Correctness

$$
\begin{aligned}
s' &= g^{w_1} Q^{w_2} & \\
&= g^{w_1} g^{d w_2} & \text{as } Q = g^d \\
&= g^{wm} g^{dsw} & \text{as } w_1 = mw \text{ and } w_2 = sw \\
&= g^{w(m+ds)} & \\
&= g^{t^{-1}(m+ds)} & \text{as } w = t^{-1} \\
&= g^k & \text{as } t = k^{-1}(m+ds)
\end{aligned}
$$

Correctness

$$
\begin{aligned}
s' &= g^{w_1} Q^{w_2} \\
&= g^{w_1} g^{d w_2} && \text{as } Q = g^d \\
&= g^{wm} g^{dsw} && \text{as } w_1 = mw \text{ and } w_2 = sw \\
&= g^{w(m+ds)} \\
&= g^{t^{-1}(m+ds)} && \text{as } w = t^{-1} \\
&= g^k && \text{as } t = k^{-1}(m+ds) \\
&= s && \text{as } s = g^k.
\end{aligned}
$$

End