Cryptography

Lecture 23

Arkady Yerukhimovich

November 18, 2024

Outline

1. Lecture 22 Review
2. Defining Digital Signatures
3. Digital Signatures from RSA

Lecture 22 Review

- ElGamal encryption scheme
- RSA encryption scheme
- Hybrid encryption
- CCA security

Outline

1. Lecture 22 Review
2. Defining Digital Signatures
3. Digital Signatures from RSA

Digital Signatures

Integrity in the Public-Key Setting

Goal of digital signatures is to provide authenticity of messages and sender.

- Public-key variant of a MAC

Digital Signatures

Integrity in the Public-Key Setting

Goal of digital signatures is to provide authenticity of messages and sender.

- Public-key variant of a MAC

Digital Signature Goals:

Digital Signatures

Integrity in the Public-Key Setting

Goal of digital signatures is to provide authenticity of messages and sender.

- Public-key variant of a MAC

Digital Signature Goals:

- A signature should authenticate sender and content of message

Digital Signatures

Integrity in the Public-Key Setting

Goal of digital signatures is to provide authenticity of messages and sender.

- Public-key variant of a MAC

Digital Signature Goals:

- A signature should authenticate sender and content of message
- Only someone knowing *sk* can sign

Digital Signatures

Integrity in the Public-Key Setting

Goal of digital signatures is to provide authenticity of messages and sender.

- Public-key variant of a MAC

Digital Signature Goals:

- A signature should authenticate sender and content of message
- Only someone knowing *sk* can sign
- Anyone knowing *pk* can verify

Digital Signatures

Integrity in the Public-Key Setting

Goal of digital signatures is to provide authenticity of messages and sender.

- Public-key variant of a MAC

Digital Signature Goals:

- A signature should authenticate sender and content of message
- Only someone knowing *sk* can sign
- Anyone knowing *pk* can verify

Advantages over a MAC:

Digital Signatures

Integrity in the Public-Key Setting

Goal of digital signatures is to provide authenticity of messages and sender.

- Public-key variant of a MAC

Digital Signature Goals:

- A signature should authenticate sender and content of message
- Only someone knowing *sk* can sign
- Anyone knowing *pk* can verify

Advantages over a MAC:

- Public verifiability

Digital Signatures

Integrity in the Public-Key Setting

Goal of digital signatures is to provide authenticity of messages and sender.

- Public-key variant of a MAC

Digital Signature Goals:

- A signature should authenticate sender and content of message
- Only someone knowing sk can sign
- Anyone knowing pk can verify

Advantages over a MAC:

- Public verifiability
- Transferability – Can show (m, σ) to a third party

Digital Signatures

Integrity in the Public-Key Setting

Goal of digital signatures is to provide authenticity of messages and sender.

- Public-key variant of a MAC

Digital Signature Goals:

- A signature should authenticate sender and content of message
- Only someone knowing sk can sign
- Anyone knowing pk can verify

Advantages over a MAC:

- Public verifiability
- Transferability – Can show (m, σ) to a third party
- Non-repudiation – After a party S signs a message, he cannot deny having sent it

Applications of Digital Signatures

1. Software updates

Applications of Digital Signatures

1. Software updates
   - Companies want to authenticate software updates

Applications of Digital Signatures

1. Software updates
   - Companies want to authenticate software updates
   - Include pk in original software

Applications of Digital Signatures

1. Software updates
   - Companies want to authenticate software updates
   - Include pk in original software
   - All valid updates signed using sk, installer verifies before installing

Applications of Digital Signatures

1. Software updates
   - Companies want to authenticate software updates
   - Include pk in original software
   - All valid updates signed using sk, installer verifies before installing
2. Blockchain transactions

Applications of Digital Signatures

1. Software updates
   - Companies want to authenticate software updates
   - Include pk in original software
   - All valid updates signed using sk, installer verifies before installing
2. Blockchain transactions
   - When issuing transactions, users need to prove identity and guarantee that transactions are not modified

Applications of Digital Signatures

1. Software updates
   - Companies want to authenticate software updates
   - Include pk in original software
   - All valid updates signed using sk, installer verifies before installing
2. Blockchain transactions
   - When issuing transactions, users need to prove identity and guarantee that transactions are not modified
   - User identity is their pk

Applications of Digital Signatures

1. Software updates
   - Companies want to authenticate software updates
   - Include pk in original software
   - All valid updates signed using sk, installer verifies before installing
2. Blockchain transactions
   - When issuing transactions, users need to prove identity and guarantee that transactions are not modified
   - User identity is their pk
   - User signs transaction with identity

Applications of Digital Signatures

1. Software updates
   - Companies want to authenticate software updates
   - Include pk in original software
   - All valid updates signed using sk, installer verifies before installing
2. Blockchain transactions
   - When issuing transactions, users need to prove identity and guarantee that transactions are not modified
   - User identity is their pk
   - User signs transaction with identity
   - Miners verify signature before adding transaction to block

Applications of Digital Signatures

1. Software updates
   - Companies want to authenticate software updates
   - Include pk in original software
   - All valid updates signed using sk, installer verifies before installing
2. Blockchain transactions
   - When issuing transactions, users need to prove identity and guarantee that transactions are not modified
   - User identity is their pk
   - User signs transaction with identity
   - Miners verify signature before adding transaction to block
3. Certificate Authorities (CAs)

Applications of Digital Signatures

1. Software updates
   - Companies want to authenticate software updates
   - Include pk in original software
   - All valid updates signed using sk, installer verifies before installing
2. Blockchain transactions
   - When issuing transactions, users need to prove identity and guarantee that transactions are not modified
   - User identity is their pk
   - User signs transaction with identity
   - Miners verify signature before adding transaction to block
3. Certificate Authorities (CAs)
   - Need to have a way to verifiably distribute pk's (e.g., for Google)

Applications of Digital Signatures

1. Software updates
   - Companies want to authenticate software updates
   - Include pk in original software
   - All valid updates signed using sk, installer verifies before installing
2. Blockchain transactions
   - When issuing transactions, users need to prove identity and guarantee that transactions are not modified
   - User identity is their pk
   - User signs transaction with identity
   - Miners verify signature before adding transaction to block
3. Certificate Authorities (CAs)
   - Need to have a way to verifiably distribute pk's (e.g., for Google)
   - CAs pks are known to all (included in Browsers)

Applications of Digital Signatures

1. Software updates
   - Companies want to authenticate software updates
   - Include pk in original software
   - All valid updates signed using sk, installer verifies before installing
2. Blockchain transactions
   - When issuing transactions, users need to prove identity and guarantee that transactions are not modified
   - User identity is their pk
   - User signs transaction with identity
   - Miners verify signature before adding transaction to block
3. Certificate Authorities (CAs)
   - Need to have a way to verifiably distribute pk's (e.g., for Google)
   - CAs pks are known to all (included in Browsers)
   - To distribute new pk for a company, company registers with CA

Applications of Digital Signatures

1. Software updates
   - Companies want to authenticate software updates
   - Include *pk* in original software
   - All valid updates signed using *sk*, installer verifies before installing
2. Blockchain transactions
   - When issuing transactions, users need to prove identity and guarantee that transactions are not modified
   - User identity is their *pk*
   - User signs transaction with identity
   - Miners verify signature before adding transaction to block
3. Certificate Authorities (CAs)
   - Need to have a way to verifiably distribute *pk*'s (e.g., for Google)
   - CAs *pk*s are known to all (included in Browsers)
   - To distribute new *pk* for a company, company registers with CA
   - CA signs company's *pk* using his key – this is a certificate

Applications of Digital Signatures

1. Software updates
   - Companies want to authenticate software updates
   - Include *pk* in original software
   - All valid updates signed using *sk*, installer verifies before installing
2. Blockchain transactions
   - When issuing transactions, users need to prove identity and guarantee that transactions are not modified
   - User identity is their *pk*
   - User signs transaction with identity
   - Miners verify signature before adding transaction to block
3. Certificate Authorities (CAs)
   - Need to have a way to verifiably distribute *pk*'s (e.g., for Google)
   - CAs *pk*s are known to all (included in Browsers)
   - To distribute new *pk* for a company, company registers with CA
   - CA signs company's *pk* using his key – this is a certificate
   - All can verify that the certificate is from trusted authority and valid

Defining Digital Signatures

Digital Signature

A digital signature scheme $\Pi = (\text{Gen}, \text{Sign}, \text{Verify})$

- $\text{Gen}(1^n)$: Outputs (pk, sk)
- $\text{Sign}_{sk}(m)$: Output a signature $\sigma \leftarrow \text{Sign}_{sk}(m)$
- $\text{Verify}_{pk}(m, \sigma)$: Output a bit $b = \text{Verify}_{pk}(m, \sigma)$

Defining Digital Signatures

Digital Signature

A digital signature scheme $\Pi = (\text{Gen}, \text{Sign}, \text{Verify})$

- $\text{Gen}(1^n)$: Outputs (pk, sk)
- $\text{Sign}_{sk}(m)$: Output a signature $\sigma \leftarrow \text{Sign}_{sk}(m)$
- $\text{Verify}_{pk}(m, \sigma)$: Output a bit $b = \text{Verify}_{pk}(m, \sigma)$

Correctness: With all but negligible probability over the choice of $(pk, sk) \leftarrow \text{Gen}(1^n)$

$$\text{Verify}_{pk}(m, \text{Sign}_{sk}(m)) = 1$$

Defining Digital Signatures

Digital Signature

A digital signature scheme $\Pi = (\text{Gen}, \text{Sign}, \text{Verify})$

- $\text{Gen}(1^n)$: Outputs (pk, sk)
- $\text{Sign}_{sk}(m)$: Output a signature $\sigma \leftarrow \text{Sign}_{sk}(m)$
- $\text{Verify}_{pk}(m, \sigma)$: Output a bit $b = \text{Verify}_{pk}(m, \sigma)$

Correctness: With all but negligible probability over the choice of $(pk, sk) \leftarrow \text{Gen}(1^n)$

$$\text{Verify}_{pk}(m, \text{Sign}_{sk}(m)) = 1$$

Notation:

- pk is often called the verification key

Defining Digital Signatures

Digital Signature

A digital signature scheme $\Pi = (\text{Gen}, \text{Sign}, \text{Verify})$

- $\text{Gen}(1^n)$: Outputs (pk, sk)
- $\text{Sign}_{sk}(m)$: Output a signature $\sigma \leftarrow \text{Sign}_{sk}(m)$
- $\text{Verify}_{pk}(m, \sigma)$: Output a bit $b = \text{Verify}_{pk}(m, \sigma)$

Correctness: With all but negligible probability over the choice of $(pk, sk) \leftarrow \text{Gen}(1^n)$

$$\text{Verify}_{pk}(m, \text{Sign}_{sk}(m)) = 1$$

Notation:

- pk is often called the verification key
- sk is often called the signing key

Defining Digital Signature Security

Let Π be a digital signature scheme. Consider the following game between an adversary $\mathcal{A}$ and a challenger:

$\mathsf{SigForge}_{\mathcal{A},\Pi}(n)$

Defining Digital Signature Security

Let Π be a digital signature scheme. Consider the following game between an adversary $\mathcal{A}$ and a challenger:

$\text{SigForge}_{\mathcal{A},\Pi}(n)$

- Challenger runs $(pk, sk) \leftarrow \text{Gen}(1^n)$ and gives pk to $\mathcal{A}$

Defining Digital Signature Security

Let Π be a digital signature scheme. Consider the following game between an adversary $\mathcal{A}$ and a challenger:

$\text{SigForge}_{\mathcal{A},\Pi}(n)$

- Challenger runs $(pk, sk) \leftarrow \text{Gen}(1^n)$ and gives pk to $\mathcal{A}$
- $\mathcal{A}$ gets pk and oracle access to $\text{Sign}_{sk}(\cdot)$ and outputs (m, σ)
  - Let Q denote the set of $\text{Sign}_{sk}(\cdot)$ queries made by $\mathcal{A}$

Defining Digital Signature Security

Let Π be a digital signature scheme. Consider the following game between an adversary $\mathcal{A}$ and a challenger:

$\text{SigForge}_{\mathcal{A},\Pi}(n)$

- Challenger runs $(pk, sk) \leftarrow \text{Gen}(1^n)$ and gives pk to $\mathcal{A}$
- $\mathcal{A}$ gets pk and oracle access to $\text{Sign}_{sk}(\cdot)$ and outputs (m, σ)
  - Let Q denote the set of $\text{Sign}_{sk}(\cdot)$ queries made by $\mathcal{A}$
- We say that $\text{SigForge}_{\mathcal{A},\Pi}(n) = 1$ (i.e., $\mathcal{A}$ wins) if $\text{Verify}_{pk}(m, \sigma) = 1$ and $m \notin Q$.

Defining Digital Signature Security

Let Π be a digital signature scheme. Consider the following game between an adversary $\mathcal{A}$ and a challenger:

$\text{SigForge}_{\mathcal{A},\Pi}(n)$

- Challenger runs $(pk, sk) \leftarrow \text{Gen}(1^n)$ and gives pk to $\mathcal{A}$
- $\mathcal{A}$ gets pk and oracle access to $\text{Sign}_{sk}(\cdot)$ and outputs (m, σ)
  - Let Q denote the set of $\text{Sign}_{sk}(\cdot)$ queries made by $\mathcal{A}$
- We say that $\text{SigForge}_{\mathcal{A},\Pi}(n) = 1$ (i.e., $\mathcal{A}$ wins) if $\text{Verify}_{pk}(m, \sigma) = 1$ and $m \notin Q$.

Definition: A digital signature $\Pi = (\text{Gen}, \text{Sign}, \text{Verify})$ is *existentially unforgeable under an adaptive chosen-message attack* if for all PPT $\mathcal{A}$,

$$\Pr[\text{SigForge}_{\mathcal{A},\Pi}(n) = 1] \leq \text{negl}(n)$$

Outline

1. Lecture 22 Review
2. Defining Digital Signatures
3. Digital Signatures from RSA

Plain RSA Signature

Plain RSA

Plain RSA Signature

Plain RSA

- Gen: $(N, e, d) \leftarrow \text{GenRSA}(1^n)$, $pk = N, e$, $sk = d$

Plain RSA Signature

Plain RSA

- Gen: $(N, e, d) \leftarrow \text{GenRSA}(1^n)$, $pk = N, e$, $sk = d$
- $\text{Sign}_{sk}(m \in \mathbb{Z}_N^*)$: $\sigma = [m^d \bmod N]$

Plain RSA Signature

Plain RSA

- Gen: $(N, e, d) \leftarrow \text{GenRSA}(1^n)$, $pk = N, e$, $sk = d$
- $\text{Sign}_{sk}(m \in \mathbb{Z}_N^*)$: $\sigma = [m^d \bmod N]$
- $\text{Verify}_{pk}(m \in \mathbb{Z}_N^*, \sigma \in \mathbb{Z}_N^*)$: Output 1 if and only if $m = [\sigma^e \bmod N]$

Plain RSA Signature

Plain RSA

- Gen: $(N, e, d) \leftarrow \text{GenRSA}(1^n)$, $pk = N, e$, $sk = d$
- $\text{Sign}_{sk}(m \in \mathbb{Z}_N^*)$: $\sigma = [m^d \bmod N]$
- $\text{Verify}_{pk}(m \in \mathbb{Z}_N^*, \sigma \in \mathbb{Z}_N^*)$: Output 1 if and only if $m = [\sigma^e \bmod N]$

Correctness: $\sigma^e = (m^d)^e = m^{[ed \bmod \phi(N)]} = m \bmod N$

Plain RSA Signature

Plain RSA

- Gen: $(N, e, d) \leftarrow \text{GenRSA}(1^n)$, $pk = N, e$, $sk = d$
- $\text{Sign}_{sk}(m \in \mathbb{Z}_N^*)$: $\sigma = [m^d \bmod N]$
- $\text{Verify}_{pk}(m \in \mathbb{Z}_N^*, \sigma \in \mathbb{Z}_N^*)$: Output 1 if and only if $m = [\sigma^e \bmod N]$

Correctness: $\sigma^e = (m^d)^e = m^{[ed \bmod \phi(N)]} = m \bmod N$

Question: Is this an unforgeable signature scheme?

Plain RSA Signature

Plain RSA

- Gen: $(N, e, d) \leftarrow \text{GenRSA}(1^n)$, $pk = N, e$, $sk = d$
- $\text{Sign}_{sk}(m \in \mathbb{Z}_N^*)$: $\sigma = [m^d \bmod N]$
- $\text{Verify}_{pk}(m \in \mathbb{Z}_N^*, \sigma \in \mathbb{Z}_N^*)$: Output 1 if and only if $m = [\sigma^e \bmod N]$

Correctness: $\sigma^e = (m^d)^e = m^{[ed \bmod \phi(N)]} = m \bmod N$

Question: Is this an unforgeable signature scheme?
Answer: No, you will work out attacks in today's quiz

Secure RSA-Based Signature

RSA-FDH (Full-Domain Hash) Signature

Secure RSA-Based Signature

RSA-FDH (Full-Domain Hash) Signature

- Gen: $(N, e, d) \leftarrow \text{GenRSA}(1^n)$, $pk = (N, e)$, $sk = d$, $H : \{0,1\}^* \to \mathbb{Z}_N^*$

Secure RSA-Based Signature

RSA-FDH (Full-Domain Hash) Signature

- Gen: $(N, e, d) \leftarrow \text{GenRSA}(1^n)$, $pk = (N, e)$, $sk = d$, $H : \{0,1\}^* \to \mathbb{Z}_N^*$
- $\text{Sign}_{sk}(m \in \{0,1\}^*)$: $\sigma = [H(m)^d \bmod N]$

Secure RSA-Based Signature

RSA-FDH (Full-Domain Hash) Signature

- Gen: $(N, e, d) \leftarrow \text{GenRSA}(1^n)$, $pk = (N, e)$, $sk = d$, $H : \{0,1\}^* \to \mathbb{Z}_N^*$
- $\text{Sign}_{sk}(m \in \{0,1\}^*)$: $\sigma = [H(m)^d \bmod N]$
- $\text{Verify}_{pk}(m, \sigma)$: Output 1 if and only if $\sigma^e = H(m) \bmod N$

Secure RSA-Based Signature

RSA-FDH (Full-Domain Hash) Signature

- Gen: $(N, e, d) \leftarrow \text{GenRSA}(1^n)$, $pk = (N, e)$, $sk = d$, $H : \{0,1\}^* \to \mathbb{Z}_N^*$
- $\text{Sign}_{sk}(m \in \{0,1\}^*)$: $\sigma = [H(m)^d \bmod N]$
- $\text{Verify}_{pk}(m, \sigma)$: Output 1 if and only if $\sigma^e = H(m) \bmod N$

Security: Secure based on RSA in ROM