Lecture Outline

- Introduction to digital signatures
  - definitions
  - security goals

- Digital signature algorithms
  - RSA signatures
  - Digital Signature Algorithm (DSA)
Digital Signatures

- A digital signature scheme is a method of signing messages stored in electronic form.

- Digital signatures can be used in very similar ways conventional signatures are used:
  - paying by a credit card and signing the bill
  - signing a contract
  - signing a letter

- Unlike conventional signatures, we have that:
  - digital signatures are not physically attached to messages
  - we cannot compare a digital signature to the original signature
Digital Signatures

- A digital signature scheme consists of the following algorithms
  
  - **key generation**
    - produces a private signing key $sk$ and a public verification key $pk$
  
  - **message signing**
    - given a message $m$ and a private key $sk$, produces a signature $\sigma(m)$ on $m$
  
  - **signature verification**
    - given a message $m$, a public key $pk$, and a signature $\sigma(m)$ on $m$
      under the corresponding secret key $sk$
    - the algorithm uses $pk$ to verify whether $\sigma(m)$ is a valid signature on $m$
Digital Signatures

- Digital signatures allows us to achieve the following security objectives:
  - authentication
  - integrity
  - non-repudiation
    - note that this is the main difference between signatures and MACs
    - a MAC cannot be associated with a unique sender since a symmetric shared key is used

- Are there other conceptual differences from MACs?
  -
  -
• **Attack models:**
  
  – **key-only attack:** adversary knows only the verification key
  
  – **known message attack:** adversary has a list of messages and corresponding signatures
    
    $$(m_1, \sigma(m_1)), (m_2, \sigma(m_2)), \ldots$$
    
  – **chosen message attack:** adversary can request signatures on messages of its choice $m_1, m_2, \ldots$
Digital Signatures

• **Adversarial goals:**
  
  – **total break:** adversary is able to obtain the private key and can forge a signature on any message
  
  – **selective forgery:** adversary is able to create a valid signature on a message chosen by someone else with a significant probability
  
  – **existential forgery:** adversary is able to create a valid signature on at least one message

• **Signature schemes are only computationally secure**
  
  – this holds for all public-key cryptosystems
  
  – remember why?
A signature scheme is defined by three PPT algorithms \((\text{Gen}, \text{Sign}, \text{Vrfy})\) such that:

1. **key generation algorithm** \(\text{Gen}\), on input a security parameter \(1^k\), outputs a key pair \((pk, sk)\), where \(pk\) is the public key and \(sk\) is the private key.

2. **signing algorithm** \(\text{Sign}\), on input a private key \(sk\) and message \(m \in \{0, 1\}^*\), outputs a signature \(\sigma\), i.e., \(\sigma \leftarrow \text{Sign}_{sk}(m)\)

3. **verification algorithm** \(\text{Vrfy}\), on input a public key \(pk\), a message \(m\), and a signature \(\sigma\), outputs a bit \(b\), where \(b = 1\) means the signature is valid and \(b = 0\) means it is invalid, i.e., \(b \equiv \text{Vrfy}_{pk}(m, \sigma)\)
• We’ll want to achieve the same level of security as in case of MACs: existential unforgeability under an adaptive chosen-message attack

• Let $\Pi = (\text{Gen}, \text{Sign}, \text{Vrfy})$ be a signature scheme

• The signature experiment $\text{Sig-forg}_{A,\Pi}(k)$:
  1. generate $(pk, sk) \leftarrow \text{Gen}(1^k)$
  2. adversary $A$ is given $pk$ and oracle access to $\text{Sign}_{sk}(\cdot)$; let $Q$ denote the set of queries $A$ makes to the oracle
  3. $A$ eventually outputs a pair $(m, \sigma)$
  4. output 1 ($A$ wins) iff (a) $\text{Vrfy}_{sk}(m, \sigma) = 1$ and (b) $m \not\in Q$
• **Definition:** A signature scheme \( \Pi = (Gen, Sign, Vrfy) \) is existentially unforgeable under an adaptive chosen-message attack if any PPT adversary \( A \) cannot win the experiment with more than negligible probability

\[
Pr[\text{Sig-forg}_A,\Pi(k) = 1] \leq \text{negl}(k)
\]

• Another essential part of signature schemes is **reliable key distribution**
  - what can happen?
  - what are consequences?
  - is this unique to signature schemes?
Plain RSA Signature Scheme

• **Key generation:**
  
  – choose large prime $p$ and $q$, set $n = pq$
  
  – compute $ed \equiv 1 \pmod{\phi(n)}$
  
  – set the public key to $(n, e)$ and the private key to $d$

• **Signing:**
  
  – given message $m$ and the key pair $pk = (n, e)$ and $sk = d$, produce the signature $\sigma(m)$ as $\sigma(m) = m^d \mod n$

• **Signature verification:**
  
  – given message $m$, a signature on it $\sigma(m)$ and the public key $pk = (n, e)$, verify the signature as $m \equiv \sigma(m)^e \mod n$
Plain or “textbook” RSA signature scheme is easily insecure

- it is easy to forge a signature
  - first choose $\sigma(m)$
  - then compute $m$ as $\sigma^e \mod n$
  - this is an existential forgery through a key-only attack
- producing a signature on a meaningful message using this attack is difficult
- forgery of meaningful messages is still easy using adversary’s ability to request signatures
• Insecurity of plain RSA signatures
  – forging a signature on an arbitrary message
    • say, adversary has \((m_1, \sigma(m_1))\) and \((m_2, \sigma(m_2))\)
    • it forges a signature on \(m_3 = m_1 \cdot m_2 \mod n\) as
      \[
      \sigma(m_3) = \sigma(m_1) \cdot \sigma(m_2) \mod n
      \]
    • this is an existential forgery using a known message attack
    • to obtain a signature on a message \(m\) of adversary’s choice:
      – \(A\) requests a signature on some \(m_1\) and \(m_2 = m/m_1 \mod n\)
      – \(\sigma(m) = \sigma(m_1) \cdot \sigma(m_2) \mod n\)
• Many modifications to plain RSA exist, but often without security proofs

• One general idea is to hash messages prior to signing
  – signing a short digest is faster than long messages
  – usage of proper cryptographic hash functions prevents forgeries
  – now a signature on $m$ is produced as $\sigma(h(m))$
  – for RSA:
    • let $h : \{0, 1\}^* \rightarrow \mathbb{Z}_n^*$ be a cryptographic hash function
    • given message $m \in \{0, 1\}^*$, sign as $\sigma = (h(m))^d \mod n$
    • verification checks whether $h(m) = \sigma^e \mod n$
• It is crucial to use strong cryptographic hash functions
  – all security properties of hash functions are required to hold to prevent different types of attacks
    • preimage resistance
    • second preimage resistance
    • collision resistance

• Let’s go back to public-key only attack
  – choose arbitrary $\sigma$ and compute $\hat{m} = \sigma^e \mod n$
  – then $\hat{m} = h(m)$ and $\sigma$ is a signature on $m$
  – what property do we need to make this forgery hard?
• **Other attacks against hashed RSA**
  
  – the need for second **preimage resistance**
    
    • assume an attacker has a valid signature \( \sigma(h(m)) \) on message \( m \)
    
    • if the second preimage property of \( h \) doesn’t hold, the attacker can find \( m' \neq m \) with \( h(m) = h(m') \)
    
    • now \( \sigma(h(m)) \) is a valid signature on \( m' \)
  
  – **collision resistance** property is similarly needed
    
    • recall the contract signing example
    
    • we construct many versions of a legitimate contract \( m \) and a bogus contract \( m' \) until a collision \( h(m) = h(m') \) is found
Security of RSA Signatures

- **Security of hashed RSA** is proven in an idealized model where $h$ is modeled as a truly random function
  - a hash function is used in practice
  - hashed RSA is widely used

- Both RSA encryption and signatures look similar, but a signature scheme cannot be built from the “reverse” of an encryption scheme
  - why?
  - it is true that RSA is both?
• How about combining encryption with signing?

• To encrypt a message $m$ and produce a signature on it, we can:
  1. sign and encrypt separately: send $E(m), \sigma(m)$
  2. sign and then encrypt: transmit $E(m||\sigma(m))$
  3. encrypt and then sign: transmit $E(m), \sigma(E(m))$

• Which one is the best?
  – what do you think about the first type?
The third type is prone to tampering

- suppose Alice sends a message to Bob using the third type $E_B(m), \sigma_A(E_B(m))$ is used
- Mallory can capture this transmission, substitute her own signature, and resend $E_B(m), \sigma_M(E_B(m))$
- Bob will think that the message came from Mallory even though the message might contain information Mallory did not possess
Signature Algorithms

- Other signature algorithms
  - **ElGamal signature scheme**
    - was published in 1985 and works in groups where the discrete logarithm problem is hard
  - **Schnorr signature scheme**
    - modifies ElGamal signature scheme to sign a digest of a message in a subgroup of $\mathbb{Z}_p^*$
  - **Digital Signature Algorithm (DSA)**
    - a signature standard adopted by NIST
    - incorporates ideas from ElGamal and Schnorr signature schemes

- All of the above schemes are probabilistic
• Long-term security for an encryption key might not be required

• Signatures, however, can be used to sign legal documents and may need to be verified many years later after signing
  – security of a signature scheme must be evaluated more carefully

• For adequate security ElGamal and RSA signature schemes leads to signatures of a thousand or more bits
  – it is possible to construct a scheme that produces shorter signatures
  – Schnorr signature scheme has significantly shorter signatures
  – this influenced development of the signature standard
• ElGamal and Schnorr signature schemes then led to another scheme called **Digital Signature Algorithm (DSA)**
  – the DSA was adopted as a standard in 1994
  – published as FIPS PUB 186
  – current revision is FIPS PUB 186-4 (released July 2013)

• Both Schnorr signature scheme and DSA
  – use a subgroup of $\mathbb{Z}_p^*$ of prime order $q$
  – have a key of the same form

• The DSA is specified to hash the message before signing
Digital Signature Algorithm

• The original DSA
  – the modulus \( p \) is required to have length \( 512 \leq |p| \leq 1024 \) such that \( |p| \) is a multiple of 64
  – the size of \( q \) is 160 bits
  – SHA-1 is used as the hash function
  – signature on a 160-bit message digest is 320 bits (2 elements in \( \mathbb{Z}_q \))

• DSA today
  – modulus \( p \) is 1024, 2048, or 3072 bits long
  – \( q \) is 160, 224, or 256 bits long
  – any hash function from FIPS 180 can be used
Digital Signature Algorithm

- Recall a common setup for groups where discrete logarithm problem is hard
  - choose prime $p$, such that $|p| \geq 1024$
  - there is a sufficiently large prime $q$ such that $q|(p - 1)$
  - $g$ is a generator of subgroup of $\mathbb{Z}_p^*$ having order $q$
  - we obtain setup for the group $(p, q, g)$
• Key generation
  – let \((p, q, g)\) be a group setup for the discrete log problem to be hard
  • we also want \(|p|\) and \(|q|\) from one of the predefined size pairs
  – let \(H : \{0, 1\}^* \rightarrow \mathbb{Z}_q\) be a hash function
  – choose secret \(x \in \mathbb{Z}_q\)
  – compute \(h \equiv g^x \pmod{p}\)
  – the public key is \(pk = (H, p, q, g, h)\)
  – the private key is \(sk = x\)
Digital Signature Algorithm

- **Signing**
  - given a message $m \in \{0, 1\}^*$, public key $pk = (H, p, q, g)$, and secret key $sk = x$
  - choose $y \in \mathbb{Z}_q^*$ uniformly at random
  - compute the signature $\sigma(m) = (\sigma_1, \sigma_2)$, where
    \[
    \sigma_1 = (g^y \mod p) \mod q \quad \text{and} \\
    \sigma_2 = (H(m) + x\sigma_1)y^{-1} \mod q
    \]
  - if $\sigma_1 = 0$ or $\sigma_2 = 0$, a new value of $y$ should be chosen
Digital Signature Algorithm

- **Signature verification**
  - given a message \( m \in \{0, 1\}^* \), signature \( \sigma(m) = (\sigma_1, \sigma_2) \) and \( pk = (H, p, q, g, h) \)
  - verification involves computing
    - \( e_1 = H(m)\sigma_2^{-1} \mod q \)
    - \( e_2 = \sigma_1\sigma_2^{-1} \mod q \)
  - then test \( (g^{e_1}h^{e_2} \mod p) \mod q \overset{?}{=} \sigma_1 \)
  - output 1 (valid) iff verification succeeds
Digital Signature Algorithm

- **Correctness property**
  - the signature $\sigma(m) = (\sigma_1, \sigma_2)$ is
    
    $$\sigma_1 = (g^y \mod p) \mod q \text{ and } \sigma_2 = (H(m) + x\sigma_1)y^{-1} \mod q$$
  - verification involves
    
    $$e_1 = H(m)\sigma_2^{-1} \mod q \text{ and } e_2 = \sigma_1\sigma_2^{-1} \mod q$$
  - the test computes
    
    $$(g^{e_1}h^{e_2} \mod p) \mod q =$$
Digital Signature Algorithm

- **Security of DSA**
  - no proof of security under the discrete logarithm problem exists
  - no proof of security even in the idealized model when $H$ is completely random

- No serious attacks have been found
  - the use of a good hash function is important

- DSS is rather popular in practice

- The standard also specifies elliptic curve version ECDSA
Beyond the Traditional Signatures

- Besides the traditional signature schemes, many other types of signature schemes with special properties exist.

- Based on their goals, we divide them into the following categories:
  
  - stronger security properties
    
    - fail-stop signatures
    - undeniable signatures
    - forward secure signatures
    - key-insulated signatures
• Signature types (cont.)
  – achieving anonymity or repudiation
    • blind signatures
    • ring signatures
    • group signatures
    • designated verifier signatures
  – constrained environments
    • aggregate signatures
  – delegation of signing rights
    • proxy signatures