Cryptography III

Public-Key Cryptography

Digital Signatures
Public Key Cryptography
Public Key Cryptography

Key pair
• **Public key**: shared with everyone
• **Secret key**: kept secret, hard to derive from the public key

Protocol
• Sender encrypts using recipient's public key
• Recipient decrypts using its secret key
Intuition

• Alice
  – Buys padlock
  – Keeps key
  – Sends open padlock to Bob
• Bob
  – Locks message with Alice’s padlock
  – Sends locked message to Alice
• Alice
  – Opens padlock with key
• No keys are exchanged
• Alice can share identical open padlocks with multiple people
Properties

• Advantages
  – A single public-secret key pair allows receiving confidential messages from multiple parties

• Disadvantages
  – Conceptually complex
  – Slower performance than symmetric cryptography
RSA

• Most widely used public key cryptosystem today
• Developed by Rivest, Shamir, and Adleman (1978)
• RSA patent expired in 2000
• 2048-bit (or longer) keys recommended
• Much slower than AES
• Typically used to encrypt an AES symmetric key
• In 1973, Clifford Cocks and James Ellis developed an equivalent system at GCHQ, declassified in 1997
Signatures: from Ink to Digital

• Signature in the real world
  – Contracts
  – Checks
  – Job offers
  – Affidavits

• Digital signatures are a matter of both computer security and law
• ESIGN Act (2000)
• Technological failures can have legal consequences
What is a Digital Signature?

message

signature algorithm

signature

Alice

Signature key

signed message

Verification algorithm

Verification key

Bob

yes

no
Goals for a Digital Signature

• Authenticity
  – Binds an identity (signer) to a message
  – Provides assurance of the signer

• Unforgeability
  – An attacker cannot forge a signature for a different identity

• Nonrepudiation
  – Signer cannot deny having signed the message

• Integrity
  – An attacker cannot take a signature by Alice for a message and create a signature by Alice for a different message
Digital Signature with Public-Key Encryption

• In a public-key cryptosystem (e.g., RSA), we can often reverse the order of encryption and decryption
  \[ E_{PK} (D_{SK} (M)) = M \]

• Alice “decrypts” plaintext message \( M \) with the secret key and obtains a digital signature on \( M \)
  \[
  \text{sign}(M, SK) \{
  \text{return } S = D_{SK} (M) \}
  \]

• Knowing Alice’s public key, \( PK \), can verify the validity of signature \( S \) on \( M \)

• Bob “encrypts” signature \( S \) with \( PK \), and

• Checks if it the result is message \( M \)
  \[
  \text{verify}(M, S, PK) \{
  \text{return } \left( M == E_{PK} (S) \right) \}
  \]
Digital Signature with Public-Key Encryption

- **Message** M
- **Decryption Algorithm**
- **Signature** $S = D_{SK}(M)$
- **Encryption Algorithm** $E_{PK}(S)$
- **Alice’s Secret Key** SK
- **Alice’s Public Key** PK
- **Signed Message** (M, S)

Alice encrypts a message M with her public key PK, creating a signed message (M, S). Bob decrypts the message M with his private key to verify the signature S using Alice’s public key.
Signing Hashes

• Basic method for public-key digital signatures
  – Signature as long as the message
  – Slow public-key encryption/decryption

• Alternative
  – Sign a cryptographic hash of the message
  – Fast cryptographic hashing

• Sign
  \[ S = D_{SK} (h(M)) \]

• Verify
  \[ h(M) == E_{PK} (S) \]

• Security
  – Security of the digital signature
  – Collision resistance of the hash function
Relying on Public Keys

- The verifier of a signature must be assured that the public key corresponds to the correct party
- The signer should not be able to deny the association with the public key
- A trusted party could keep and publish pairs (identity, public key)
  - Government?
  - Private organizations?
- What if the private key is compromised?
  - Need for key revocation mechanism
Sharing a Symmetric Key

• Alice wants to send Bob a message $M$ containing a symmetric key for subsequent encrypted communication

• Requirements
  – Confidentiality of $M$ and subsequent communication
  – Bob understands he is communicating with Alice

• Message $M$ needs to be encrypted and digitally signed

• Alice and Bob want to be protected against an active adversary, Eve
  – Eve can eavesdrop and modify messages

• Eve had previously shared a symmetric key with Alice
  – Eve has $M'$ with signature $S'$ from Alice
Encrypt and Sign

• Sign and encrypt
  – Alice creates signature $S$ of $M$
  – Alice creates ciphertext $C$ of $M$
  – Alice sends $(C, S)$ to Bob
  – Bob decrypts $C$ to $M$ and verifies $S$ on $M$

• Attack
  – Eve intercepts $(C, S)$ and replaces $S$ with $S'$ and $C$ with $C'=E(M')$
  – Bob receives $(C', S')$, decrypts $C'$ to $M'$ and verifies $S'$ on
  – Bob sends Alice a message encrypted with the key in $M'$
  – Eve eavesdrops the message and decrypts it (the key in $M'$ is known to her)
• Sign then encrypt
  - Alice creates signature $S$ of $M$
  - Alice creates encryption $C = E((M, S))$
  - Alice sends $C$ to Bob
  - Bob decrypts $C$ to $(M, S)$ and verifies $S$ on $M$

• Attack
  - Eve replaces $C$ with $E(M', S')$
  - Bob decrypts $C'$ to $(M', S')$ and verifies $S'$ on $M'$
  - Bob sends Alice a message encrypted with the key in $M'$
  - Eve eavesdrops and decrypts

• Countermeasure
  - Alice signs $(M, PK_{Bob})$
Encrypt then Sign

- Encrypt then sign
  - Alice encrypts M to C
  - Alice creates signature S of C
  - Alice sends (C, S) to Bob
  - Bob verifies S on C
  - Bob decrypts C to M

- Attack
  - Eve replaces S with her signature $S''$ on C and forwards (C, $S''$) to Bob
  - Bob now thinks he is communicating with Eve
  - Eve can then forward Bob’s response (intended for Eve) to Alice

- Countermeasure
  - Alice signs both M and C

- Other issue
  - No signature on M exists, hence repudiation opportunity for Alice
Demo
Command-Line Crypto with OpenSSL
What We Have Learned

• Public-key encryption
  – Benefits and how to use it
• Digital signatures
  – Properties and realization with public-key encryption in reverse
• Transmitting encrypted and signed messages
  – Pitfalls and remedies
• Basic crypto in the shell with OpenSSL
  – Encrypt, hash, and sign