Public Key Cryptography

Kostas Papagiannopoulos
University of Amsterdam
kostaspap88@gmail.com – kpcrypto.net

Contents

Introduction

Introduction

- Symmetric key cryptography requires that the sender and receiver somehow know a shared secret key K in advance
- The same key K is used for encryption and decryption

Introduction

- Symmetric key cryptography requires that the sender and receiver somehow know a shared secret key K in advance
- The same key K is used for encryption and decryption
- How can Alice and Bob agree on a key in first place?
- We have a chicken-egg situation regarding the secure key distribution

Introduction

- **Public key cryptography** by Diffie and Hellman in 1976 and by Rivest, Shamir and Adleman in 1977
- Alice and Bob no longer need a shared secret key to communicate

Introduction

- **Public key cryptography** by Diffie and Hellman in 1976 and by Rivest, Shamir and Adleman in 1977
- Alice and Bob no longer need a shared secret key to communicate
- Alice now has a **public** encryption key K_A^{pub} known to all, including Bob
- Everyone can use it to encrypt a message M for Alice

$$C = enc(M, K_A^{pub})$$

Introduction

- **Public key cryptography** by Diffie and Hellman in 1976 and by Rivest, Shamir and Adleman in 1977
- Alice and Bob no longer need a shared secret key to communicate
- Alice now has a **public** encryption key K_A^{pub} known to all, including Bob
- Everyone can use it to encrypt a message M for Alice

$$C = enc(M, K_A^{pub})$$

- Alice also has a **private** decryption key K_A^{pri} known only to her
- She can use it to decrypt a ciphertext C

$$M = dec(C, K_A^{pri})$$

Introduction

- **Public key cryptography** by Diffie and Hellman in 1976 and by Rivest, Shamir and Adleman in 1977
- Alice and Bob no longer need a shared secret key to communicate
- Alice now has a **public** encryption key K_A^{pub} known to all, including Bob
- Everyone can use it to encrypt a message M for Alice

$$C = enc(M, K_A^{pub})$$

- Alice also has a **private** decryption key K_A^{pri} known only to her
- She can use it to decrypt a ciphertext C

$$M = dec(C, K_A^{pri})$$

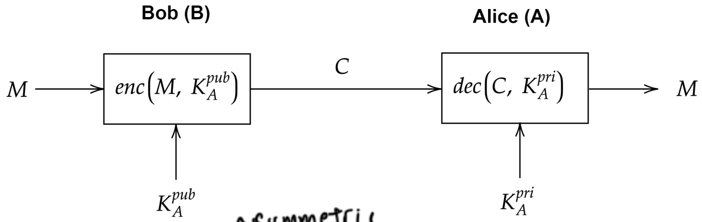
- Bob can establish his own public/private key pair (K_B^{pub}, K_B^{pri}) so that everyone, including Alice, can encrypt messages for him

Introduction

Confidentiality using PK cryptography

- Anyone can communicate with Alice and be certain that only Alice can decrypt their messages
- In general, public key cryptography is slower than symmetric cryptography
- **Hybrid cryptosystem.** We will first exchange keys using the slow public key algorithm and then communicate using the fast symmetric key algorithm

fix chicken-and-egg

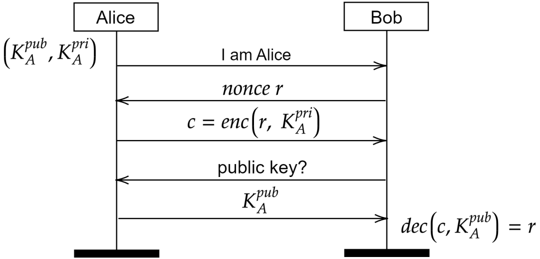


asymmetric

Introduction

Authentication using PK cryptography

1. Alice initiates contact with Bob and claims that she is Alice
2. Bob sends nonce r to Alice
3. Alice encrypts r with her private key K_A^{pri} and sends $c = enc(r, K_A^{pri})$ to Bob
4. Bob acquires the public key K_A^{pub} of Alice
5. Bob decrypts c and verifies that $dec(c, K_A^{pub})$ is equal to his nonce r

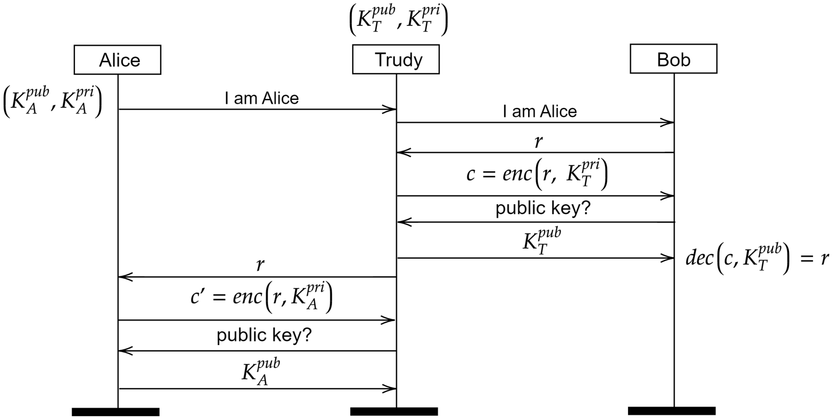


公钥加密,私钥解密 / 私钥解密,公钥加密

Introduction

Authentication using PK cryptography

- Trudy (the attacker) can perform a man-in-the-middle attack to impersonate Alice



man-in-the-middle attack

Introduction

Digital signatures using PK cryptography

- What was the problem with the previous attack?

Introduction

Digital signatures using PK cryptography

- What was the problem with the previous attack?
- Anyone can claim to be Alice and make a public key

Introduction

Digital signatures using PK cryptography

- What was the problem with the previous attack?
- Anyone can claim to be Alice and make a public key

- Public key cryptography provides a solution by “reversing” the public/private key pair in order to **sign** a document

Introduction

Digital signatures using PK cryptography

- What was the problem with the previous attack?
- Anyone can claim to be Alice and make a public key
- Public key cryptography provides a solution by "reversing" the public/private key pair in order to **sign** a document
- Alice has generated the public/private key pair (K_A^{pub}, K_A^{pri}) and wants to sign document m

Introduction

Digital signatures using PK cryptography

- What was the problem with the previous attack?
- Anyone can claim to be Alice and make a public key

- Public key cryptography provides a solution by "reversing" the public/private key pair in order to **sign** a document

- Alice has generated the public/private key pair (K_A^{pub}, K_A^{pri}) and wants to sign document m
- Alice decrypts document m with her **private** key K_A^{pri} producing a signed document d

$$d = dec(m, K_A^{pri})$$

Introduction

Digital signatures using PK cryptography

- What was the problem with the previous attack?
- Anyone can claim to be Alice and make a public key

- Public key cryptography provides a solution by “reversing” the public/private key pair in order to **sign** a document

- Alice has generated the public/private key pair (K_A^{pub}, K_A^{pri}) and wants to sign document m
- Alice decrypts document m with her **private** key K_A^{pri} producing a signed document d

$$d = dec(m, K_A^{pri})$$

- Now everyone can verify that Alice signed the document by encrypting d using the **public** key of Alice K_A^{pub}

$$m = enc(d, K_A^{pub})$$

Introduction

Digital signatures using PK cryptography

- What was the problem with the previous attack?
- Anyone can claim to be Alice and make a public key

- Public key cryptography provides a solution by "reversing" the public/private key pair in order to **sign** a document

- Alice has generated the public/private key pair (K_A^{pub}, K_A^{pri}) and wants to sign document m
- Alice decrypts document m with her **private** key K_A^{pri} producing a signed document d

$$d = dec(m, K_A^{pri})$$

- Now everyone can verify that Alice signed the document by encrypting d using the **public** key of Alice K_A^{pub}

$$m = enc(d, K_A^{pub})$$

- Digital signatures provide the **non-repudiation** property i.e. Alice cannot deny signing the document m

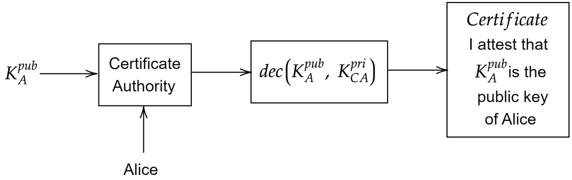
Introduction

Digital signatures using PK cryptography

- How can signatures fix the authentication protocol?
- Alice contacts a Certification Authority (CA)
- The CA can bind the public key of Alice K_A^{pub} to the entity called Alice
- The CA will sign the K_A^{pub} to provide non-repudiation

Introduction

Digital signatures using PK cryptography



1. Alice gets identified by the CA
2. Alice's public key K_A^{pub} gets signed by the (trustworthy) CA using the CA's private key K_{CA}^{pri}
3. The signed public key K_A^{pub} is linked to Alice and the CA provides a **certificate**
4. Anyone that wants to confirm the certificate can check the CA's signature by encrypting the certificate using the CA's public key K_{CA}^{pub}

The RSA Algorithm

RSA

- The RSA cryptosystem is named after its inventors Rivest, Shamir, and Adleman
- It is possible that it was discovered earlier by GCHQ but was classified

RSA

- The RSA cryptosystem is named after its inventors Rivest, Shamir, and Adleman
- It is possible that it was discovered earlier by GCHQ but was classified

- **RSA security.** The security of the algorithm relies on the hardness of factoring the product of two large prime numbers

Given primes p and q, it is easy to compute their product $n = pq$

However given n, it is hard to find the prime factors p and q

RSA

- The RSA cryptosystem is named after its inventors Rivest, Shamir, and Adleman
- It is possible that it was discovered earlier by GCHQ but was classified

- **RSA security.** The security of the algorithm relies on the hardness of factoring the product of two large prime numbers

  Given primes p and q, it is easy to compute their product $n = pq$

  However given n, it is hard to find the prime factors p and q

- Factoring is not known to be NP-complete and attacks like the general number field sieve have damaged the algorithm's security

RSA

- The RSA cryptosystem is named after its inventors Rivest, Shamir, and Adleman
- It is possible that it was discovered earlier by GCHQ but was classified

- **RSA security.** The security of the algorithm relies on the hardness of factoring the product of two large prime numbers

Given primes p and q, it is easy to compute their product $n = pq$

However given n, it is hard to find the prime factors p and q

- Factoring is not known to be NP-complete and attacks like the general number field sieve have damaged the algorithm's security
- Today RSA remains secure as long as large key sizes are used, typically 2048 bits or more

RSA

- The RSA cryptosystem is named after its inventors Rivest, Shamir, and Adleman
- It is possible that it was discovered earlier by GCHQ but was classified

- **RSA security.** The security of the algorithm relies on the hardness of factoring the product of two large prime numbers

Given primes p and q, it is easy to compute their product $n = pq$

However given n, it is hard to find the prime factors p and q

- Factoring is not known to be NP-complete and attacks like the general number field sieve have damaged the algorithm's security
- Today RSA remains secure as long as large key sizes are used, typically 2048 bits or more
- The large key sizes make RSA slow thus public key cryptography has moved to elliptic curve cryptosystems (ECC) that are much faster

RSA

- The RSA cryptosystem is named after its inventors Rivest, Shamir, and Adleman
- It is possible that it was discovered earlier by GCHQ but was classified

- **RSA security.** The security of the algorithm relies on the hardness of factoring the product of two large prime numbers

Given primes p and q, it is easy to compute their product $n = pq$

However given n, it is hard to find the prime factors p and q

- Factoring is not known to be NP-complete and attacks like the general number field sieve have damaged the algorithm's security
- Today RSA remains secure as long as large key sizes are used, typically 2048 bits or more
- The large key sizes make RSA slow thus public key cryptography has moved to elliptic curve cryptosystems (ECC) that are much faster
- Both RSA and ECC will no longer be secure after the arrival of quantum computers due to Shor's algorithm

RSA

- The RSA cryptosystem is named after its inventors Rivest, Shamir, and Adleman
- It is possible that it was discovered earlier by GCHQ but was classified

- **RSA security.** The security of the algorithm relies on the hardness of factoring the product of two large prime numbers

Given primes p and q, it is easy to compute their product $n = pq$

However given n, it is hard to find the prime factors p and q

- Factoring is not known to be NP-complete and attacks like the general number field sieve have damaged the algorithm's security
- Today RSA remains secure as long as large key sizes are used, typically 2048 bits or more
- The large key sizes make RSA slow thus public key cryptography has moved to elliptic curve cryptosystems (ECC) that are much faster
- Both RSA and ECC will no longer be secure after the arrival of quantum computers due to Shor's algorithm
- Thus public key cryptography is currently moving to lattice-based cryptosystems that are (hopefully) **post-quantum** secure

RSA

RSA Key Generation

- Alice wants to communicate with Bob and she selects the two “large” primes denoted as p and q

  e.g. Select $p = 11$ and $q = 3$

RSA

RSA Key Generation

- Alice wants to communicate with Bob and she selects the two "large" primes denoted as p and q

  e.g. Select $p = 11$ and $q = 3$

- Form the product $n = pq$ and the product $\phi(n) = (p-1)(q-1)$

  e.g. Compute $n = 11 * 3 = 33$ and $\phi(n) = 10 * 2 = 20$

RSA

RSA Key Generation

- Alice wants to communicate with Bob and she selects the two "large" primes denoted as p and q

  e.g. Select $p = 11$ and $q = 3$

- Form the product $n = pq$ and the product $\phi(n) = (p-1)(q-1)$

  e.g. Compute $n = 11 * 3 = 33$ and $\phi(n) = 10 * 2 = 20$

- Choose some e relatively prime to $\phi(n)$

  e.g. A choice is $e = 3$ which is relatively prime to $\phi(n) = 20$

RSA

RSA Key Generation

- Alice wants to communicate with Bob and she selects the two "large" primes denoted as p and q

  e.g. Select $p = 11$ and $q = 3$

- Form the product $n = pq$ and the product $\phi(n) = (p-1)(q-1)$

  e.g. Compute $n = 11 * 3 = 33$ and $\phi(n) = 10 * 2 = 20$

- Choose some e relatively prime to $\phi(n)$

  e.g. A choice is $e = 3$ which is relatively prime to $\phi(n) = 20$

- Find the multiplicative inverse of e modulo $\phi(n)$, denoted as d

$$ed \equiv 1 \bmod \phi(n)$$

  e.g. using Euclid's algorithm we can find $d = 7$ since $7 * 3 \equiv 1 \bmod 20$

RSA

RSA Key Generation

- Alice wants to communicate with Bob and she selects the two “large” primes denoted as p and q

  e.g. Select $p = 11$ and $q = 3$

- Form the product $n = pq$ and the product $\phi(n) = (p-1)(q-1)$

  e.g. Compute $n = 11 * 3 = 33$ and $\phi(n) = 10 * 2 = 20$

- Choose some e relatively prime to $\phi(n)$

  e.g. A choice is $e = 3$ which is relatively prime to $\phi(n) = 20$

- Find the multiplicative inverse of e modulo $\phi(n)$, denoted as d

  $$ed \equiv 1 \bmod \phi(n)$$

  e.g. using Euclid's algorithm we can find $d = 7$ since $7 * 3 \equiv 1 \bmod 20$

- Alice has generated a public key and a private key

  $$\text{Public key: } (n, e) = (33, 3) \qquad \text{Private key: } d = 7$$

- We call n the **modulus**, e the **encryption exponent** and d the **decryption exponent**

RSA

RSA Encryption and Decryption

- To encrypt message m, Bob must use the public key of Alice i.e. he must raise the message to the encryption exponent e modulo the value n

$$c = m^e \bmod n$$

e.g. If the message $m = 15$ we have:

$$c = 15^3 \bmod 33 = 9$$

RSA

RSA Encryption and Decryption

- To encrypt message m, Bob must use the public key of Alice i.e. he must raise the message to the encryption exponent e modulo the value n

$$c = m^e \bmod n$$

e.g. If the message $m = 15$ we have:

$$c = 15^3 \bmod 33 = 9$$

- To decrypt the ciphertext c, Alice must raise it to the decryption exponent d modulo the value n

$$m = c^d \bmod n$$

e.g. For ciphertext $c = 9$ we have:

$$m = 9^3 \bmod 33 = 15$$

Diffie-Hellman Key Exchange

Diffie-Hellman

- The DH cryptosystem is named after its inventors Whitfield Diffie and Martin Hellman
- It is (again!) possible that it was discovered earlier by GCHQ but was classified
- DH is not used for encrypting or signing, but it allows users to establish a shared secret

Diffie-Hellman

- The DH cryptosystem is named after its inventors Whitfield Diffie and Martin Hellman
- It is (again!) possible that it was discovered earlier by GCHQ but was classified
- DH is not used for encrypting or signing, but it allows users to establish a shared secret

- **DH security.** DH relies on the computational hardness of the discrete log problem

$$\text{Given } g, p \text{ and } g^k \bmod p, \text{ find the secret } k$$

- The discrete log is not known to be NP-complete
- DH can be broken with a quantum computer using Shor's algorithm

Diffie-Hellman

DH Key Exchange

- Alice and Bob want to exchange keys so they agree on a prime p and a generator g
- **Generator.** For any $x \in \{1, 2, \ldots, p-1\}$ we can find exponent n such that $x = g^n \bmod p$

  e.g. Let $p = 23$ and $g = 5$

Diffie-Hellman

DH Key Exchange

- Alice and Bob want to exchange keys so they agree on a prime p and a generator g
- **Generator.** For any $x \in \{1, 2, \ldots, p-1\}$ we can find exponent n such that $x = g^n \bmod p$

  e.g. Let $p = 23$ and $g = 5$

- Alice chooses a secret exponent a and Bob chooses a secret exponent b

  e.g. Let $a = 4$ and $b = 3$

Diffie-Hellman

DH Key Exchange

- Alice and Bob want to exchange keys so they agree on a prime p and a generator g
- **Generator.** For any $x \in \{1, 2, \dots, p-1\}$ we can find exponent n such that $x = g^n \mod p$

  e.g. Let $p = 23$ and $g = 5$

- Alice chooses a secret exponent a and Bob chooses a secret exponent b

  e.g. Let $a = 4$ and $b = 3$

- Alice computes $g^a \mod p$ and sends it to Bob

  e.g. Let $g^a \mod p = 5^4 \mod 23 = 4$

Diffie-Hellman

DH Key Exchange

- Alice and Bob want to exchange keys so they agree on a prime p and a generator g
- **Generator.** For any $x \in \{1, 2, \ldots, p-1\}$ we can find exponent n such that $x = g^n \bmod p$

  e.g. Let $p = 23$ and $g = 5$

- Alice chooses a secret exponent a and Bob chooses a secret exponent b

  e.g. Let $a = 4$ and $b = 3$

- Alice computes $g^a \bmod p$ and sends it to Bob

  e.g. Let $g^a \bmod p = 5^4 \bmod 23 = 4$

- Bob computes $g^b \bmod p$ and sends it to Alice

  e.g. Let $g^b \bmod p = 5^3 \bmod 23 = 10$

Diffie-Hellman

DH Key Exchange

- Alice and Bob want to exchange keys so they agree on a prime p and a generator g
- **Generator.** For any $x \in \{1, 2, \ldots, p-1\}$ we can find exponent n such that $x = g^n \mod p$

  e.g. Let $p = 23$ and $g = 5$

- Alice chooses a secret exponent a and Bob chooses a secret exponent b

  e.g. Let $a = 4$ and $b = 3$

- Alice computes $g^a \mod p$ and sends it to Bob

  e.g. Let $g^a \mod p = 5^4 \mod 23 = 4$

- Bob computes $g^b \mod p$ and sends it to Alice

  e.g. Let $g^b \mod p = 5^3 \mod 23 = 10$

- Now both Alice and Bob can compute the shared secret s

$$s = (g^b)^a \mod p = g^{ab} \mod p = 10^4 \mod 23 = 18$$

$$s = (g^a)^b \mod p = g^{ab} \mod p = 4^3 \mod 23 = 18$$