IT-Security

Chapter 4: Asymmetric Cryptography

Public key encryption, Digital Signatures, Diffie-Hellman Key Agreement

Prof. Dr.-Ing. Ulrike Meyer

Overall Lecture Context

- **In the security mechanisms we covered so far**
  - Alice and Bob needed to share the same secret key
- **In this chapter we learn how asymmetric cryptosystems work**
  - Alice can share a single public key with multiple other parties and keeps a private key to herself
  - In an asymmetric encryption scheme,
    - anyone in possession of Alice's public key can encrypt messages for Alice
    - but only Alice can (with the private key) decrypt messages
  - In a digital signature scheme
    - only Alice can sign a messages
    - anyone in possession of the public key can verify a signature on a message

Overview

- **Basic Number Theory**
  - Finite Fields, greatest common divisor, Fermat's theorem
  - Factorization
  - Discrete Logarithms
- **Digital Signatures**
  - Intuition on integrity protection with digital signatures
  - RSA as signature scheme
  - Digital signature standard
- **Public Key Encryption Schemes**
  - Intuition
  - RSA as encryption scheme
- **Diffie-Helman Key Agreement**
  - Basic idea
  - Man-in-the-middle attack
- **Quantum Computers**

Modular Arithmetic and Residue Class Rings

Let $\mathbb{Z}_{\boldsymbol{n}} = \{\bar{\mathbf{0}}, \bar{\mathbf{1}}, \bar{\mathbf{2}}, \ldots, \overline{\boldsymbol{n}-\mathbf{1}}\}$ with $\bar{k} = \{ x \in \mathbb{Z} \mid x \mod n = k\}$

Addition: $\mathbb{Z}_n \times \mathbb{Z}_n \to \mathbb{Z}_n$, $\bar{a} + \bar{b} := \overline{a + b}$

Then, for all $\bar{a}, \bar{b} \in \mathbb{Z}_n$ it holds that

$\bar{a} + \bar{b} = \bar{b} + \bar{a}$

$(\bar{a} + \bar{b}) + \bar{c} = \bar{a} + (\bar{b} + \bar{c})$

$\bar{0} + \bar{a} = \bar{a}$

$\bar{a} + \overline{n - a} = \bar{n} = \bar{0}$

Multiplication: $\mathbb{Z}_n \times \mathbb{Z}_n \to \mathbb{Z}_n$, $\bar{a} \bullet \bar{b} := \overline{ab}$

Then, for all $\bar{a}, \bar{b} \in \mathbb{Z}_n$ it holds that

$\bar{a} \bullet \bar{b} = \bar{b} \bullet \bar{a}$

$(\bar{a} \bullet \bar{b}) \bullet \bar{c} = \bar{a} \bullet (\bar{b} \bullet \bar{c})$

$\bar{1} \bullet \bar{a} = \bar{a}$

$\bar{a}$ is called invertible mod n if there is an $\bar{x} \in \mathbb{Z}_n$ such that $\bar{a} \bullet \bar{x} = \bar{1}$

Thus, $(\mathbb{Z}_{\boldsymbol{n}}, +, \bullet)$ forms a commutative ring with 1

For ease of reading, we will denote $\bar{k}$ as k mod n in the rest of this lecture

Example: Addition and Multiplication in $\mathbb{Z}_6$

+	0	1	2	3	4	5
0	0	1	2	3	4	5
1	1	2	3	4	5	0
2	2	3	4	5	0	1
3	3	4	5	0	1	2
4	4	5	0	1	2	3
5	5	0	1	2	3	4

•	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	1	2	3	4	5
2	0	2	4	0	2	4
3	0	3	0	3	0	3
4	0	4	2	0	4	2
5	0	5	4	3	2	1

- **Invertible in $\mathbb{Z}_6$:**
  - 1, 5
- **Not invertible in $\mathbb{Z}_6$:**
  - 0, 2, 3, 4
- **Not all elements of $\mathbb{Z}_6 \setminus \{0\}$ are invertible**

$\Rightarrow (\mathbb{Z}_6, +, \bullet)$ is a ring but not a finite field

Example: Addition and Multiplication in $\mathbb{Z}_5$

+	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

•	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

- **Invertible in $\mathbb{Z}_5$:**
  - 1, 2, 3, 4
- **Not invertible in $\mathbb{Z}_5$:**
  - 0
- **All elements of $\mathbb{Z}_5 \setminus \{0\}$ are invertible**

$\Rightarrow (\mathbb{Z}_5, +, \bullet)$ is a finite field

Extended Euclidian Algorithm

Let $\mathbf{gcd}(\boldsymbol{n}, \boldsymbol{k})$ denote the greatest common divisor of $\boldsymbol{n}$ and $\boldsymbol{k}$

Then there are integers $\boldsymbol{x}, \boldsymbol{y}$ such that $\boldsymbol{xn} + \boldsymbol{yk} = \mathbf{gcd}(\boldsymbol{n}, \boldsymbol{k})$

Euclidian algorithm computes $\gcd(n, k)$

Input: integers k, n **with** $\boldsymbol{n} > \boldsymbol{k} > 1$

Set $r_0 = n, r_1 = k$

WHILE $r_{i+2} > 0$

Compute q_{i+1}, r_{i+2} with $r_i = q_{i+1} \cdot r_{i+1} + r_{i+2}$

END(WHILE)

RETURN $\gcd(n, k) = \boldsymbol{r_{i+1}}$

Extended Euclidian algorithm

Additionally computes x, y

Set $u_0 = v_1 = 1, u_1 = v_0 = 0$

WHILE $r_{i+2} > 0$

Compute $u_{i+2} = u_i - q_{i+1} \cdot u_{i+1}$

Compute $v_{i+2} = v_i - q_{i+1} \cdot v_{i+1}$

END(WHILE)

RETURN $\boldsymbol{x} = \boldsymbol{u_{i+1}}$ and $\mathbf{y} = \boldsymbol{v_{i+1}}$

Example

Euclidian algorithm to compute $\gcd(595, 408)$

Set $r_0 = 595, r_1 = 408$

$$r_0 = q_1 \cdot r_1 + r_2$$

$$595 = 1 \cdot 408 + 187$$

$$r_1 = q_2 \cdot r_2 + r_3$$

$$408 = 2 \cdot 187 + 34$$

$$r_2 = q_3 \cdot r_3 + r_4$$

$$187 = 5 \cdot 34 + \mathbf{17}$$

$$r_2 = q_3 \cdot r_3 + r_5$$

$$34 = 2 \cdot 17 + 0$$

$\Rightarrow \mathbf{gcd(408, 595) = 17}$

Extended Euclidian algorithm additionally computes x, y

Set $u_0 = v_1 = 1, u_1 = v_0 = 0$

$u_2 = u_0 - q_1 u_1$ $\qquad$ $v_2 = v_0 - q_1 v_1$

$u_2 = 1 - 1 \cdot 0 = 1$ $\qquad$ $v_2 = 0 - 1 \cdot 1 = -1$

$u_3 = u_1 - q_2 u_2$ $\qquad$ $v_3 = v_1 - q_2 v_2$

$u_3 = 0 - 2 \cdot 1 = -2$ $\qquad$ $v_3 = 1 - 2 \cdot (-1) = 3$

$u_4 = u_2 - q_3 u_3$ $\qquad$ $v_4 = v_2 - q_3 v_3$

$u_4 = 1 - 5 \cdot (-2) = 11$ $\qquad$ $v_4 = -1 - 5 \cdot (3) = -16$

$\Rightarrow \mathbf{11 \cdot 595 + (-16) \cdot 408 = 17}$

Correctness of the Euclidian Algorithm

Observation: $\gcd(n, k) = \gcd(n - k, k)$

Proof:

- If d divides n and k , then there are r, s with $n = rd$ and $k = sd$
- Thus $n - k = (r - s)d$, so that d also divides $n - k$
- Thus, any divisor of n and k also divides $n - k$
- Vice verse if $d|k$ and $d|\ n - k$, then there are w, t with $n - k = wd$ and $k = td$
- Thus $n = n - k + k = (w + t)d$ and any divisor of $n - k$ and k also divides n

Consequence: $\gcd(n, k) = \gcd(n \bmod k\ , k) \Rightarrow \gcd(n, k) = \gcd(r_2, k) = \gcd(r_2, r_3) \ldots$

Applying this repeatedly until the remainder $r_{i+2} = 0$ **gives us** $r_{i+1} = \gcd(r_{i-1}, r_i) = \gcd(n, k)$

Existence of Multiplicative Inverses

$a \in \mathbb{Z}_n$ **is invertible** $\mathrm{mod}\, n$ $\Leftrightarrow$ a **and** n **are relatively prim** $\Leftrightarrow$ $\gcd(n, a)$ **= 1**

Proof of "⇒" : Assume a is invertible

⇒ there is an integer x such that $xa = 1 \bmod n$

⇒ there is an integer k such that $xa = 1 + kn$

⇒ there is an integer k such that $xa + (-k)n = 1$

Now if there was an integer d s.t. $d|a$ and $d|k$

⇒ $d|\, xa + (-k)n$ and thus: $d|\,1$

⇒ $d = 1$ and thus a and n are relatively prime

Proof of "⇐": Assume a and n are relatively prime.

Then $\gcd(a, n) = 1$

⇒ there are integers x, y such that $xa + yn = 1$

⇒ $xa = 1 - yn = 1 \bmod n$

⇒ x is the inverse of $a \bmod n$

$\mathbb{Z}_n^*$ ≔ Set of invertible elements in $\mathbb{Z}_n$

For p prime, $\mathbb{Z}_p^* = \mathbb{Z}_p \setminus \{0\}$ and $(\mathbb{Z}_p, +, \bullet)$ is a field

Euler's φ function

- **The number $|\mathbb{Z}_n^*|$ of invertible elements of $\mathbb{Z}_n$ is called $\varphi(\boldsymbol{n})$**
- **For a prime number p it holds that $\varphi(\boldsymbol{p}) = \boldsymbol{p} - 1$**
  - ▶ All elements of $\mathbb{Z}_p \setminus \{0\}$ are invertible $\bmod \boldsymbol{p}$

  $\Rightarrow \varphi(\boldsymbol{p}) = \boldsymbol{p} - 1$
- **If $n = pq$ where p and q are two different prime numbers, then**

$$\varphi(\boldsymbol{n}) = (\boldsymbol{p} - 1)(\boldsymbol{q} - 1)$$

  - ▶ Not invertible: $p, 2p, 3p, \dots, (q-1)p, qp \rightarrow q$ elements
  - ▶ Not invertible: q, $2q, \dots, (p-1)q \rightarrow$ another $p - 1$ elements
  - ▶ The other $n - q - (p-1) = n - q - p + 1$ elements are invertible

  $\Rightarrow \varphi(\boldsymbol{n}) = (\boldsymbol{p} - 1)(\boldsymbol{q} - 1)$

Examples:

$\mathbb{Z}_5^*$ = {1, 2, 3, 4},

$\mathbb{Z}_4^*$ = {1, 3},

$\mathbb{Z}_{10}^*$ = {1, 3, 7, 9}

Euler's Theorem

Euler's theorem:

For any $a \in \mathbb{Z}_n^*$: $a^{\varphi(n)} = 1 \bmod n$

Proof:

- If $a, b \in \mathbb{Z}_n^*$, then $a \cdot b \in \mathbb{Z}_n^*$
- Multiplying all elements of $\mathbb{Z}_n^*$ with some $a \in \mathbb{Z}_n^*$ just reorders them:
  - Assume $\boldsymbol{x}$ is the product of all different $\boldsymbol{x}_1, \dots, \boldsymbol{x}_{\varphi(n)} \in \mathbb{Z}_n^*$
  - Then, for any $a \in \mathbb{Z}_n^*$: $ax_1 ax_2 \dots ax_{\varphi(n)} = a^{\varphi(n)} x = x$
    - otherwise $ax_i = ax_j$ for some $i \neq j$
  - Multiplying the above equation with x^{-1} on both sides yields $a^{\varphi(n)} = 1 \bmod n$

Consequence:

For any $a \in \mathbb{Z}_n^*$ and any integer s it holds that $a^{\varphi(n)s+1} = a \bmod n$

Generalization of Euler's Theorem

Generalization:

Let $n = pq$ where p and q are **two different** prime numbers then

for all $a \in \mathbb{Z}_n$ it holds that $a^{\varphi(n)+1} = a \bmod n$

Proof: For $a = 0$ the equation obviously holds

For all invertible $a \in \mathbb{Z}_n$ we already proofed it on the last slide

So, lets assume an $a \in \mathbb{Z}_n$ that is not invertible

- Then it is either divisible by p or by q (or both but then $a = 0$).
- Let's assume a is not divisible by p but divisible by q.
- Then, $a^{p-1} = 1 \bmod p$ and $a^{q-1} = 0 \bmod q$
- Thus, $a^{\varphi(n)+1} = (a^{p-1})^{q-1}a = a \bmod p$ and $a^{\varphi(n)+1} = (a^{q-1})^{p-1}a = a \bmod q$
- Thus, there are integers r and s with $a^{\varphi(n)+1} = a + rp$ and $a^{\varphi(n)+1} = a + sq$

- Consequently, $rp = sq$ such that $q \mid r$
- So, there is an integer l with $r = lq$
- Thus, $a^{\varphi(n)+1} = a + rp = a + lqp = \mathrm{a} + ln$

$\Rightarrow a^{\varphi(n)+1} = a \bmod n$

The Factorization Problem

Definition

Given a composite integer n, find a non-trivial factor of n

Hardness of Factorization

- No known polynomial time algorithms for factorization on classical computers
- Best current algorithms for classical computers have sub-exponential run-time
  - Pollard’s Rho Method
  - Quadratic Sieve
  - Number Sieve
  - …

The Discrete Logarithm Problem

Definition DL Problem

Given a cyclic group G, a generator $g \in G$, and g^x but not x, find the **discrete logarithm** x.

$\Rightarrow$

Definition Decisional Diffie-Hellman Problem

Given Given a cyclic group G, a generator $g \in G$, and g^x , g^y , g^z but not x, y, z, decide if $g^{x\,y} = g^z$

- **The security of many asymmetric cryptosystems is based on the hardness of the discrete logarithm problem or the decisional Diffie-Hellman problem**
- **Relation between the two problems**
  - If in a group the discrete logarithm problem can be solved, the DDH problem can also be solved

Overview

- **Basic Number Theory**
  - Finite Fields, greatest common divisor, Fermat's theorem
  - Factorization
  - Discrete Logarithms

- **Digital signature schemes**
  - Intuition
  - RSA as signature scheme
  - Digital signature standard

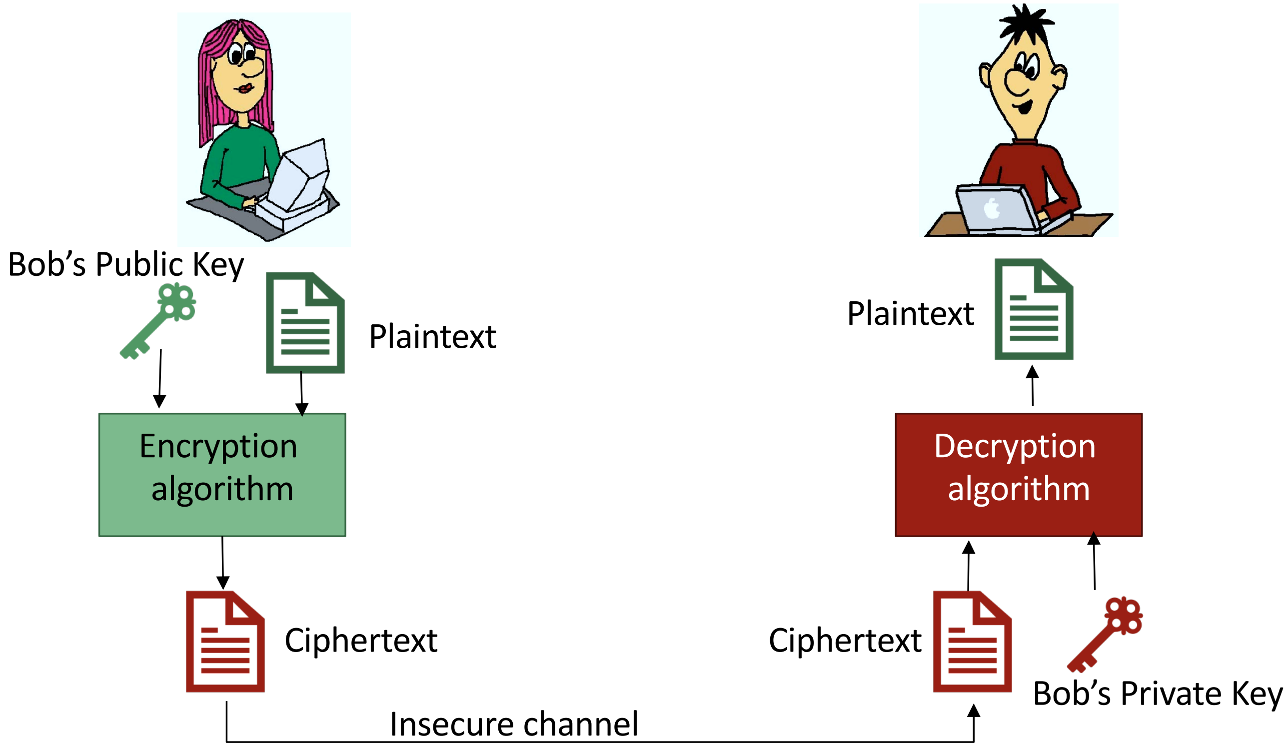
- **Public Key Encryption Schemes**
  - Intuition
  - RSA as encryption scheme

- **Diffie-Helman Key Agreement**
  - Basic idea
  - Man-in-the-middle attack

- **Quantum Computers**

Intuition Public Key Encryption

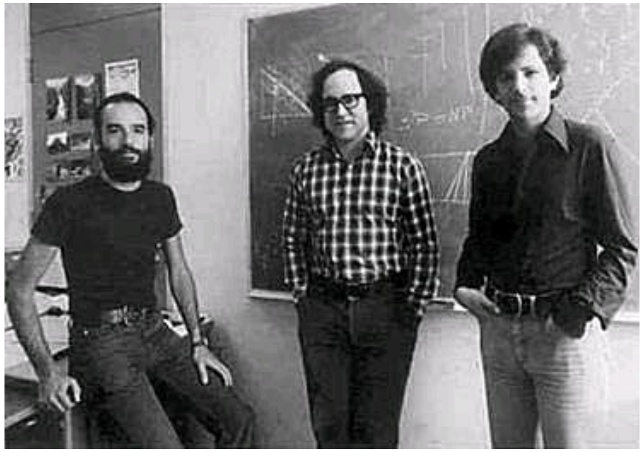
- Alice wants to send a **confidential** plaintext to Bob
- Alice has an authentic copy of Bob's **public key**
- Alice uses Bob's public key to **encrypt** plaintext to ciphertext
- Bob uses his **private key** to **decrypt** ciphertext to plaintext
- Decryption is **"difficult"** without the **private key**

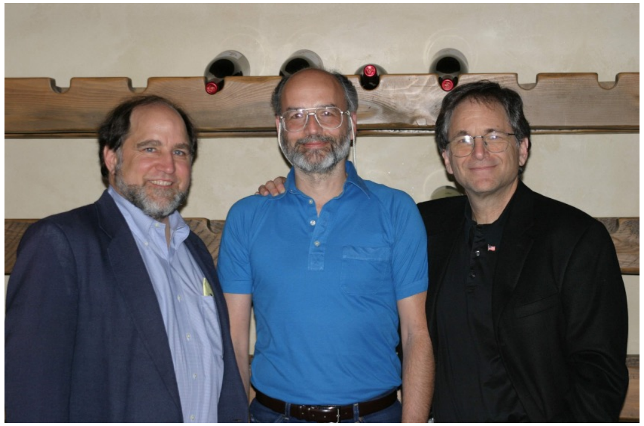


Note: The definition of an encryption scheme presented in Chapter 2 also holds for asymmetric encryption!

RSA

- **First asymmetric encryption scheme invented in 1977**
  - By Ron Rivest, Adi Shamir, and Leonard Adleman at MIT
  - Original idea of asymmetric encryption goes back to Diffie and Hellman, though
- **Patented from 1983 to 2000**
- **Supports different key lengths and variable block sizes**
  - Currently, 2048 bit keys are considered sufficient
  - Implies a block length of 2048 bit
- **Requires plaintext blocks to be represented as integers**
  - Requires a coding scheme that converts bit strings in integers





RSA Key Generation

Public Key

- Randomly select two different large prime numbers p, q
- Set $n := pq$
- Chose $e \in \mathbb{Z}_n$ such that e is invertible $\mod \varphi(n)$
- Set public key to (n, e)

Private Key

- Compute $d \in \mathbb{Z}_n$ such that $ed = 1 \mod \varphi(n)$
  - ∃ integer k such that $ed = 1 + k\,\varphi(n)$
- Set private key to d

Side Notes

- Large prime numbers can be found by
  - Choosing random numbers of appropriate size
  - Testing for primality with probabilistic primality tests
- If the desired bit length of the modulus is k than p and q should be $k/2$-bit prime numbers
- Choose $e \in \mathbb{Z}_n$ randomly; check if $\gcd(e, n) = 1$
- Compute d from e with the Extended Euclidian Algorithm

RSA Operation

Encryption

For a public RSA key $pk = (e, n)$,

$E_{pk} : \mathbb{Z}_n \to \mathbb{Z}_n$

$E_{pk}(m) = c = m^e \, mod \, n$

Decryption

For the corresponding private RSA key $sk = d$

$D_{sk} : \mathbb{Z}_n \to \mathbb{Z}_n$

$D_{sk}(c) = c^d = m \, mod \, n$

Correctness of RSA

For any ciphertext $c \in \mathbb{Z}_n$:

$$c^d = m^{ed} \bmod n = m^{\varphi(n)k+1} \bmod n = m \bmod n$$

Small Example:

Key generation:

Let $p = 3$, $q = 5$, then $n = pq = 15$

$\varphi(n) = 2 \cdot 4 = 8$

Chose $e = 3$, then e is invertible mod $\varphi(n)$ as 8 and 3 are relatively prime

Setting $d = 3$ we get $ed = 9 = 1 \bmod 8$

Encryption of $m = 7$:

$m^e \bmod n = 7^3 \bmod 15 = 343 \bmod 15 = 13$

Decryption of $c = 13$:

$c^d \bmod n = 13^3 \bmod 15 = 2197 \bmod 15 = 7$

Efficient Modular Exponentiation

- **RSA Encryption and Decryption: $\boldsymbol{x}^k \bmod \boldsymbol{n}$**
- **“Naïve” modular exponentiation**
  - Requires k modular multiplications
  - Problem: the size of the exponent is of the same order as the size of the modulus $\boldsymbol{n}$
  - Naïve modular exponentiation is not efficient
- **More efficient modular exponentiation**
- **Idea: Use the binary representation of k**
  - $k = \sum k_i 2^i = k_0 + 2(k_1 + 2(k_2 + \cdots) \cdots)$ where $k_i \in \{0,1\}$
  - Then we get $x^k = \prod x^{k_i 2^i}$
  - So, all we need to do is square and multiply

Example

- $k = 37 = 1 + 2^2 + 2^5$
- So $x^{37} = x \cdot x^{2^2} \cdot x^{2^5} = ((((x^2)^2)^2 x)^2)^2 x$
- Two multiplications by x and 5 squares

RSA Security (1)

Theorem:

Let p, q be prime numbers and $n = p \cdot q$

Then n can be efficiently factorized iff $\varphi(n)$ can be computed efficiently

Proof:

"$\Longrightarrow$": If n can be efficiently factorized then p and q can efficiently be computed from n and therefore $\varphi(n) = (p-1) \cdot (q-1)$ is efficiently computable

"$\Longleftarrow$": If $\varphi(n)$ is known, then one can compute p and q from the two equations $n = p \cdot q$ and $\varphi(n) = (p-1) \cdot (q-1)$

⚠ **Factorizing n is equivalent to computing $\varphi(n)$**

RSA Security (2)

Theorem:

Let p, q be prime numbers and $n = p \cdot q$ and (e, n) a public RSA key and d the corresponding private key. Then d can be efficiently computed from (e, n) iff n can be factorized efficiently.

Proof:

"$\Longrightarrow$ ": There is a probabilistic polynomial-time algorithm that computes p and q from d, e, and n

"$\Longleftarrow$": clear: if we can factorize n we have p and q and can compute $\varphi(n)$ and can thus compute d as the inverse of e mod $\varphi(n)$

⚠ **Computing d is equivalent to factorizing n**

RSA Security (3)

Summary:

- Compute a private RSA key d from public key (e, n) is equivalent to factorizing n
- Factorizing n is equivalent to computing $\varphi(n)$

⚠ **It is still unclear if there is a way to decrypt RSA-encrypted messages without knowledge of the private key d**

Recall Hardness of Factorization:

- For classical computers, there is currently no polynomial-time algorithm for factorization

Chosen Plaintext Attack Against RSA

Recall from Chapter 2: chosen plaintext attack against a cipher

- Attacker can obtain ciphertext for plaintexts of its choice

Example: RSA can always be attacked in a chosen plaintext setting

- Any attacker with access to the public key (e, n) can generate ciphertexts for plaintexts of its choice
  - Attacker choses m and computes $c = m^e \mod n$

⚠ **For deterministic asymmetric ciphers we always need to consider a chosen plaintext setting as realistic**

Semantic Security

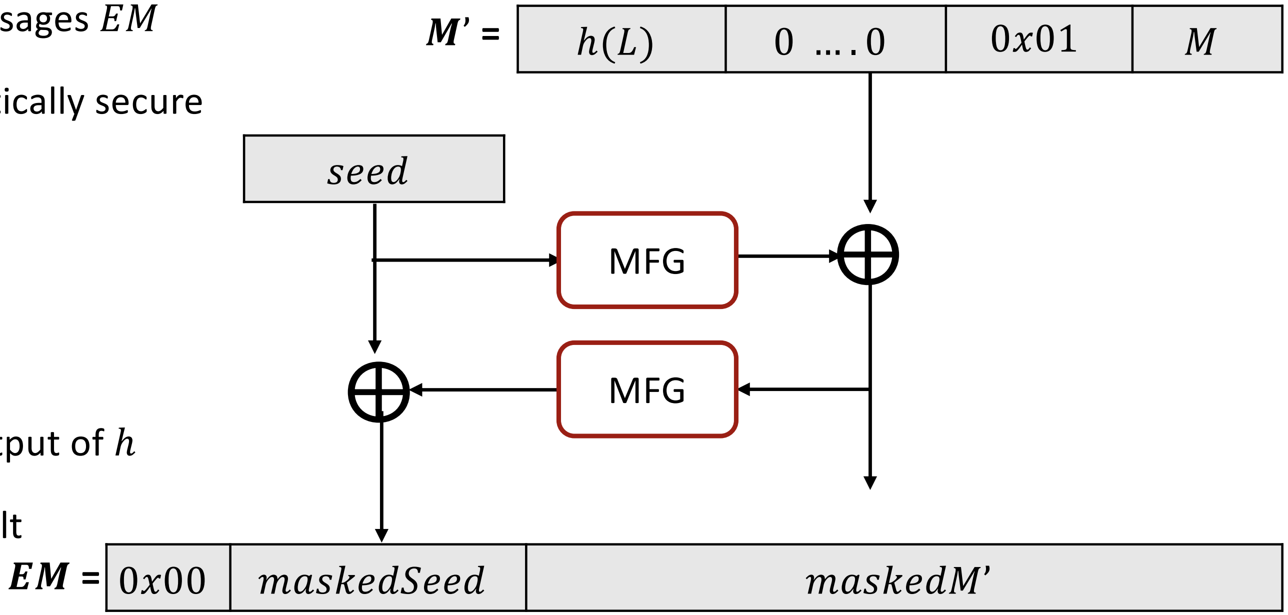
Definition: Semantic Security

- Assume a challenger choses two plaintexts m_1 and m_2
- He encrypts the plaintexts with a public key pk $c_1 = E_{pk}(m_1)$ and $c_2 = E_{pk}(m_2)$
- He then provides m_1, m_2, c_1, c_2 and pk to an adversary
- Then the public key encryption schemes is said to be **semantically secure**
  - if the adversary cannot guess with a probability larger than ½ which ciphertext encrypts which plaintext

⚠ **Deterministic asymmetric ciphers like (textbook) RSA are not semantically secure**

Turning RSA into a Semantically Secure Cipher with OAEP

- **The Optimal Asymmetric Encryption Padding OAEP**
  - Converts message M into encoded messages EM
  - Uses random seed to make RSA semantically secure
- **Notations**
  - M: bit-string message to encrypt
  - h: hash function
  - $seed$: random seed, same length as output of h
  - L: optional label, empty string by default
  - MGF: mask generation function
  - Padding with zeros:
    - let n be a k-byte modulus, then $k - |M| - 2|h(L)| - 2$ bytes of zero bytes are used as padding



Backdoors in Key Generation

- **Idea**
  - Whenever RSA is used,
    - keys must be generated
  - Whoever implements these key generation
    - can manipulate the code such that keys generated with it include a backdoor
  - This backdoor allows him to
    - retrieve the private key corresponding to a public key generated with his implementation

- **Underlining Model**
  - Manufacturer (Attacker)
    - Designer of the backdoor
    - Integrates the backdoor in the key generation code
  - User (Victim)
    - In possession of a device or piece of code for key generation, e.g. for RSA, manipulated by the manufacturer
    - Can observe public and private keys generated by his device
  - External attacker
    - Can observe public keys used by the user

Backdoor for RSA Key Generation

Naïve RSA Backdoor

- Key generation code with backdoor
  - Fix a prime number $\boldsymbol{p}$
  - Choose a second prime number $\boldsymbol{q}$ at random
  - Set $\boldsymbol{n} = \boldsymbol{qp}$
  - Select $\boldsymbol{e}$ relatively prime to $\boldsymbol{\varphi}(\boldsymbol{n})$ and $\boldsymbol{d}$ such that $\boldsymbol{ed} = \mathbf{1} \bmod \boldsymbol{\varphi}(\boldsymbol{n})$

Exploiting the backdoor

- If manufacturer sees that user uses $(\boldsymbol{e}, \boldsymbol{n})$
  - compute $\boldsymbol{q}$ by $\boldsymbol{n}/\boldsymbol{p}$, from $\boldsymbol{q}, \boldsymbol{p}, \boldsymbol{e}$ compute $\boldsymbol{d}$

Unfortunately

- External attacker that observes two public keys $(\boldsymbol{e}, \boldsymbol{n})$ and $(\boldsymbol{e}', \boldsymbol{n}')$ can compute $\boldsymbol{p} = \gcd(\boldsymbol{n}, \boldsymbol{n}')$
  - Thus, any external attacker that suspects this backdoor can check for it
- User can check if the code/devices has this backdoor in the same way

Backdoor for RSA Key Generation

Better RSA Backdoor

- Manufacturer's RSA key pair $(\boldsymbol{E}, \boldsymbol{N})$ and $\boldsymbol{D}$
- Key generation code with backdoor
  - Pick random prime numbers $\boldsymbol{p}$ and $\boldsymbol{q}$ and set $\boldsymbol{n} = \boldsymbol{pq}$
  - Compute $\boldsymbol{e} = \boldsymbol{p}^{\boldsymbol{E}} \, mod \, \boldsymbol{N}$
  - Check if $\boldsymbol{e}$ is invertible mod$\varphi(\boldsymbol{n})$
  - If yes, compute the inverse $\boldsymbol{d}$ and output $(\boldsymbol{e}, \boldsymbol{n})$, $\boldsymbol{d}$
  - If no, pick a new prime number $\boldsymbol{p}$ and start again

Exploiting the backdoor

- If manufacturer sees that client uses $(\boldsymbol{e}, \boldsymbol{n})$
- Compute $\boldsymbol{e}^{\boldsymbol{D}} \, mod \, \boldsymbol{N} = \boldsymbol{p}$ and can use this to compute $\boldsymbol{q}$ and then $\boldsymbol{d}$

External attacker and user

- Cannot check for this backdoor as they do not have the private key $\boldsymbol{D}$
- To the user $\boldsymbol{e}$ looks as if it was randomly picked

⚠ **Backdoors like this exist for the key generation operations of many public key cryptosystems**

Overview

- **Basic Number Theory**
  - Finite Fields, greatest common divisor, Fermat's theorem
  - Factorization
  - Discrete Logarithms

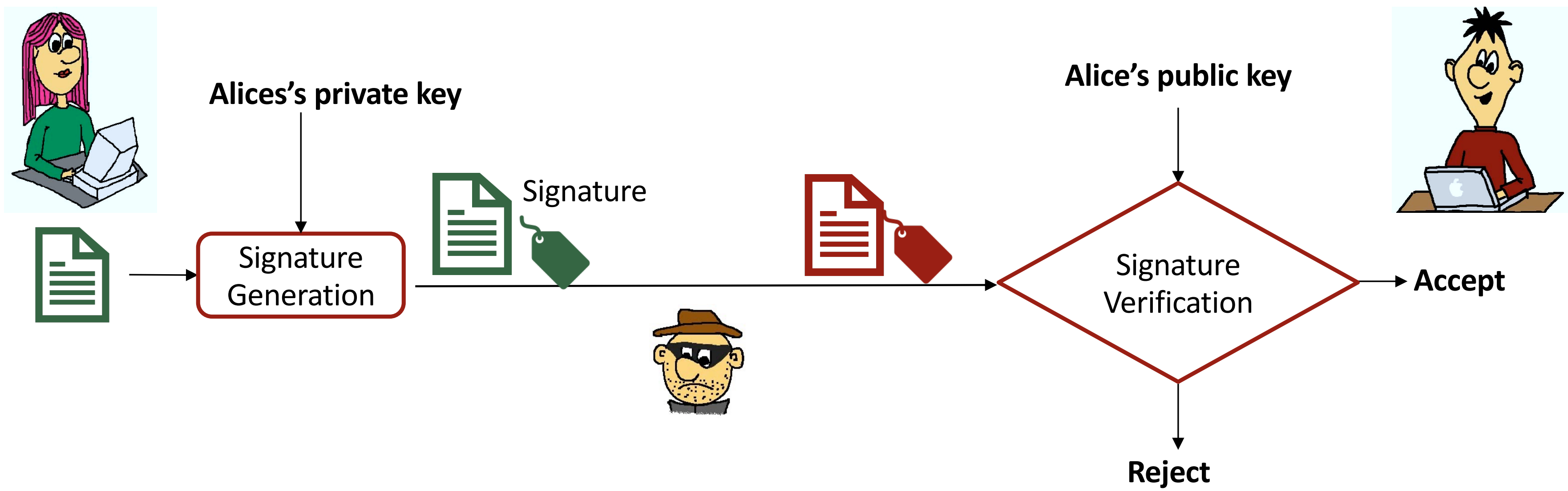
- **Digital signature schemes**
  - Intuition
  - RSA as signature scheme
  - Digital signature standard

- **Public Key Encryption Schemes**
  - Intuition
  - RSA as encryption scheme

- **Diffie-Helman Key Agreement**
  - Basic idea
  - Man-in-the-middle attack

- **Quantum Computers**

Intuition Digital Signatures



- Alice uses her private key to generate a signature on the message
- Anyone in possession of Alice's public key can verify the signature
- Difficult to generate a message, signature pair that is accepted by the signature verification
  - Without access to the private key

Definition Digital Signature Scheme

A digital signature scheme consists of

- A key generation algorithm that
  - generates a public key pk for signature verification
  - generates a private key sk for signature generation
- A family of signature generation algorithms sig_{sk} that
  - takes a message M as input and outputs the signature $\text{sig}_{sk}(M)$
- A family of signature verification algorithms ver_{pk} that
  - takes a message M and a signature $\text{sig}_{sk}(M)$ as input and
  - returns success or failure

Naïve RSA Signatures (Insecure!)

Key generation as in RSA Encryption

Public Key

- Randomly select two large prime numbers p, q
- Set $n := pq$
- Chose $e \in \mathbb{Z}_n$ such that e is invertible mod $\varphi(n)$
- Set public key pk = (n, e)

Private Key

- Compute private key $sk = d \in \mathbb{Z}_n$ such that $ed = 1 \text{mod}\ \varphi(n)$

Signature generation

- signature s on message m: $s = m^d$ mod n

Signature verification

- $s^e = m^{de} \stackrel{?}{=} m$

Vulnerable to existential forgery

- Attacker can choose signature s and compute $m = s^e$ and then claim that (m, s) is a valid signature

RSA Signature Scheme

Key generation as in Naïve RSA

Signature generation

- Let h be a publicly known cryptographic hash function
- Signature s on m is $s = h(m)^d$

Signature verification

- On receipt of $(\bar{m}, \bar{s})$ verifier checks if $h(\bar{m}) \stackrel{?}{=} \bar{s}^e \; mod \; n$

Secure against existential forgery

- Attacker cannot find a message m such that $h(m) = s^e$ as h is pre-image resistant

Hashing before signing is also required for security reasons in many other asymmetric signature schemes ⚠

Attacks on Digital Signatures

Attack result

- Total break: (partial) recovery of the signature key
- Universal forgery: forge signatures on any message of the attacker's choice
- Selective forgery: forge a signature on a specific chosen message
- Existential forgery: merely results in some valid message/signature pair not already known to the adversary

Power of attacker

Strength of attacker increases ↓

- Key-Only Attack: Attacker only in possession of the public verification key
- Known-Message Attack: Attacker observes some message/signature pairs; tries to generate another valid pair
- Chosen-Message Attack: Attacker can choose messages and can make the signer sign them; tries to generate another valid pair

Digital Signature Algorithm

- **Adopted as standard by NIST in 1994**
- **Standardized in FIPS 186**
- **Security is based on the DDH assumption**
  - Related to but strong than the Discrete Logarithm problem
- **Can be defined over different cyclic groups for which DDH assumption seems to hold, e.g.**
  - Cyclic sub-groups of order q of $\mathbb{Z}_p^*$, where p and q are prime numbers where q divides $(p-1)$
- **Variants for other cyclic groups exist**
  - E.g. ECDSA on specific elliptic curves over a finite field

Key Generation for DSA

Public parameters

- Two prime number p, q with q|(p-1)
- x ∈ $\mathbb{Z}_p^*$ such that $g := x^{\frac{p-1}{q}}$ mod p ≠ 1
  - The smallest interger i or which $g^i = 1 \, mod \, p$ is $i = q$
  - Thus, g generates a sub group of order q in $\mathbb{Z}_p^*$
- Cryptographic hash function h

Private key

- Chose $a \in \{1, \ldots, q-1\}$ uniformly at random and set $sk = a$

Public key

- Set A = $g^a \bmod p$ as public key pk

Example

Parameters

- $p = 11, \; q = 5$
- Select $x = 2$, then $g = 4$

Private key

- Chose a $= 3$

Public key

- Set A = $g^a \bmod p$ = $4^3 \bmod 11 = 9$

DSA Operation

Signature generation on message m

- Chooses k $\in \{1, \dots, q-1\}$ uniformly at random
- Signer computes

  r = $(g^k \bmod p) \bmod q$

  $s = k^{-1}(h(m) + ar) \bmod q$

- Signature: $\mathrm{sig}_{sk}(m) = (r, s)$

Signature verification

- Upon receipt of m, r, s the verifier
- Checks if r $\in \{1, \dots, q-1\}$ and s $\in \{1, \dots, q-1\}$
- Computes $u_1 = h(m)s^{-1} \bmod q$, $u_2 = rs^{-1} \bmod q$
- Computes v = $g^{u_1} A^{u_2} \bmod p \bmod q$
- Accept signature if $v = r$, reject otherwise

Correctness of Verification

- **Upon receipt of m, r, s the verifier computes**

$$\text{v} = g^{u_1} A^{u_2} \bmod p \bmod q$$

$$= g^{h(m)s^{-1}} A^{rs^{-1}} \bmod p \bmod q$$

$$= g^{h(m)s^{-1} + ars^{-1}} \bmod p \bmod q$$

$$= g^{s^{-1}(h(m)+ar)} \bmod p \bmod q$$

$$= g^{s^{-1}sk} \bmod p \bmod q$$

$$= g^{s^{-1}sk} \bmod p \bmod q$$

$$= r$$

g was selected such that $\boldsymbol{g^q} = 1 \bmod \boldsymbol{p}$, thus
$\boldsymbol{g}^k \bmod \boldsymbol{p} = \boldsymbol{g}^{\boldsymbol{k} \bmod \boldsymbol{q}} \bmod \boldsymbol{p}$

Reusing k leads to a total break of DSA

Assume $\boldsymbol{k}$ is used to sign two known messages $\boldsymbol{m_1}$ and once for $\boldsymbol{m_2}$, then

$\boldsymbol{r} = (\boldsymbol{g}^{\boldsymbol{k}}\ mod\ \boldsymbol{p})\ mod\ \boldsymbol{q}$ (same for both messages)

$$\boldsymbol{s_1} = \left(\boldsymbol{k}^{-1}(\ h(\ \boldsymbol{m_1}) + \ \boldsymbol{ar})\right) mod\ \boldsymbol{q}$$

$$\boldsymbol{s_2} = (\boldsymbol{k}^{-1}(\ h(\ \boldsymbol{m_2}) + \ \boldsymbol{ar}))\ mod\ \boldsymbol{q}$$

Thus, $\boldsymbol{s_1} - \boldsymbol{s_2} = \boldsymbol{k}^{-1}(\ h(\boldsymbol{m_1}) - h(\boldsymbol{m_2})\ )\ mod\ \boldsymbol{q}$

and therefore: $\boldsymbol{k} = (\boldsymbol{s_1} - \boldsymbol{s_2})^{-1}(\ h(\boldsymbol{m_1}) - h(\boldsymbol{m_2})\ )\ mod\ \boldsymbol{q}$

And thus, $\boldsymbol{a} = \boldsymbol{r}^{-1}\left(\ \boldsymbol{s_1 k} - h(\boldsymbol{m_1})\right) mod\ \boldsymbol{q}$

I.e., private key $\boldsymbol{a}$ can be computed by anyone observing the messages

and signatures if the same k is used twice

MACs versus Digital Signatures

- **MACs can provide**
  - Message integrity
  - Origin authentication
- **Require verifier to share a secret key with MAC producer**

- **Signature Schemes can provide**
  - Message integrity
  - Origin authentication
  - Broadcast authentication
  - Non-repudiation
- **Require verifier to obtain an authentic copy of public key of signer**

Overview

- **Basic Number Theory**
  - Finite Fields, greatest common divisor, Fermat's theorem
  - Factorization
  - Discrete Logarithms

- **Digital signature schemes**
  - Intuition
  - RSA as signature scheme
  - Digital signature standard

- **Public Key Encryption Schemes**
  - Intuition
  - RSA as encryption scheme

- **Diffie-Helman Key Agreement**
  - Basic idea
  - Man-in-the-middle attack

- **Quantum Computers**

Diffie-Hellman (DH) Key Agreement

- **Oldest public key mechanism**
  - Invented in 1976
- **Is a key establishment protocol by which two parties can**
  - Establish a symmetric secret key K
  - Based on publicly exchanged values
- **Security based on hardness of discrete logarithm problem**
  - Any polynomial-time algorithm that solves the DL problem also solves the computational DH-problem:
    - Given a prime number p, a generator g of $\mathbb{Z}_p^*$, g^a, g^b find $K = g^{ab}$
  - It is unknown if the computational DH-problem can be solved without solving the DL problem

Diffie-Hellman Key Agreement

Public parameters

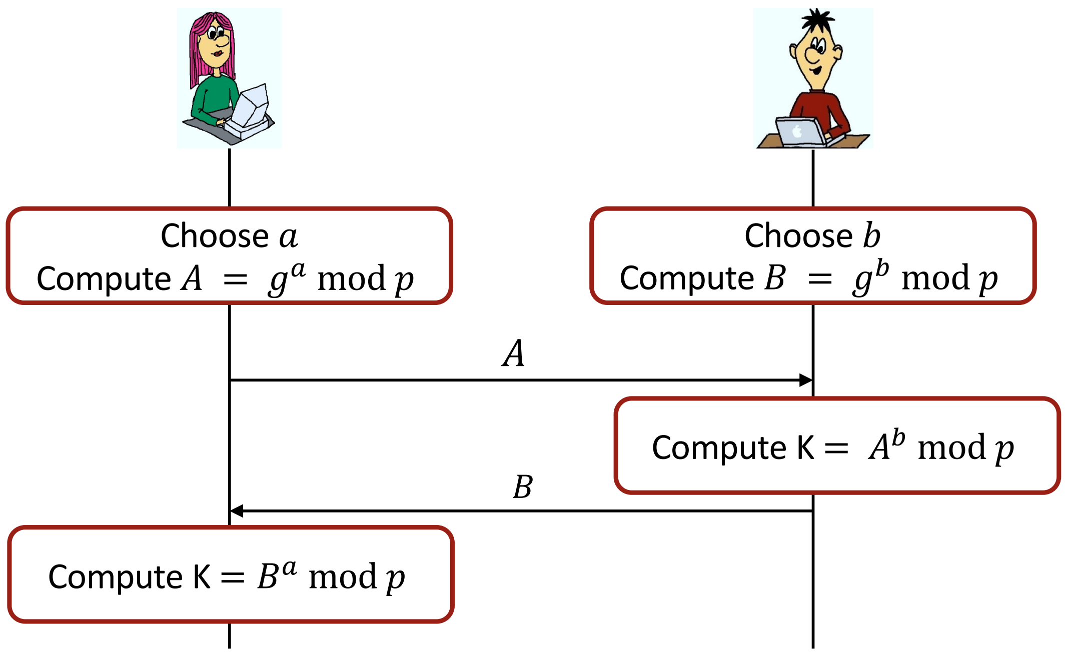
- Prime number p, generator g of $\mathbb{Z}_p^*$

Private values

- Private DH-value of Alice
  - $a \in \{2, \dots, p-2\}$ chosen uniformly at random
- Private DH-value of Bob
  - $b \in \{2, \dots, p-2\}$ chosen uniformly at random

Public values

- Public DH-value of Alice $A = g^a \bmod p$
- Public DH-value of Bob $B = g^b \bmod p$



As $A^b \bmod p = g^{ab} = g^{ba} = B^a \bmod p$

Alice and Bob now share the secret key K = g^{ab}

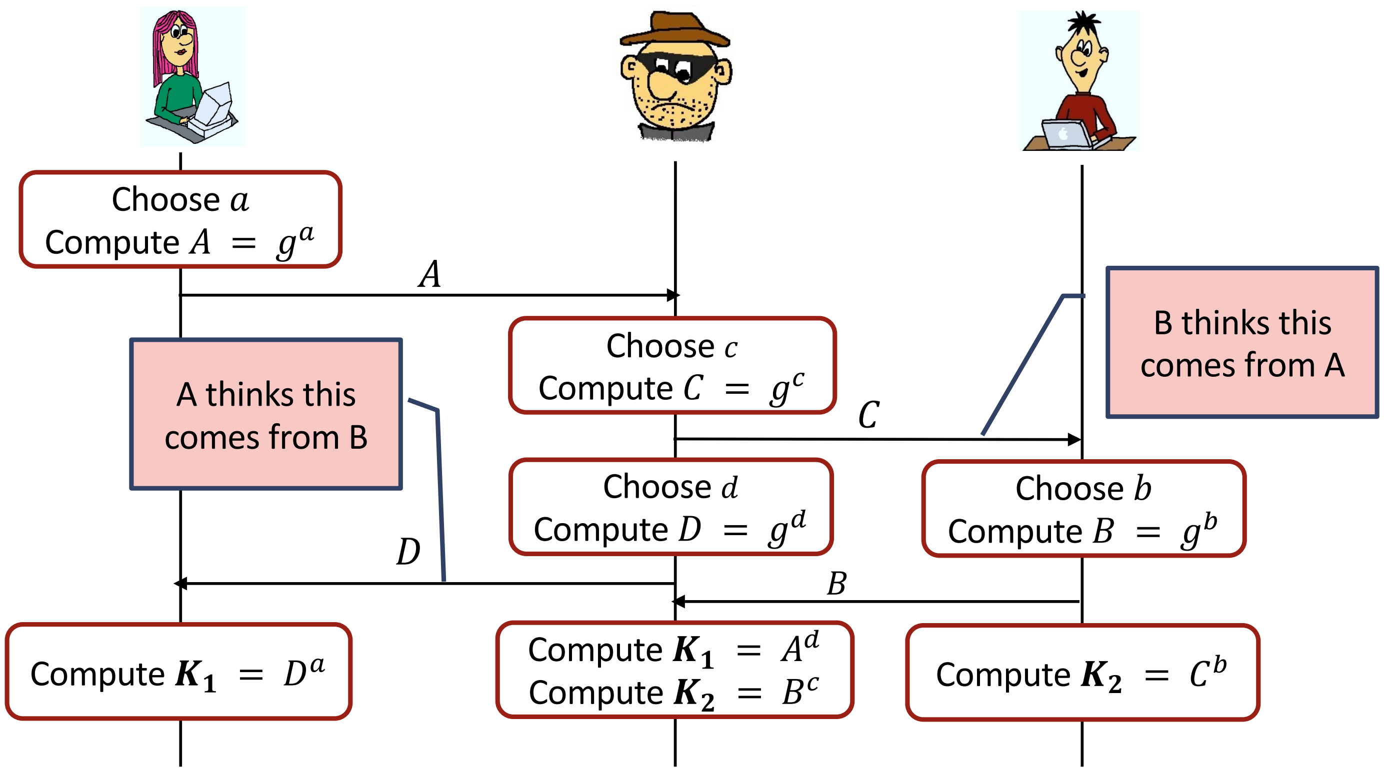
Man-in-the-Middle Attack

All computations are done mod p and a, b, c, d are chosen from $\{2, \ldots, p-2\}$

Result

- A shares $\boldsymbol{K_1}$ with attacker
  - but thinks she shares it with B
- B shares $\boldsymbol{K_2}$ with attacker
  - but thinks he shares it with A
- A and B do not share key
  - but they think they do

⇒ Attacker can eavesdrop!



Symmetric vs. Asymmetric Cryptography

Symmetric Cryptography

- More efficient
  - Often used to encrypt large amounts of data
- Higher number of secret keys required
  - $n(n-1)/2$ keys required to enable pairwise confidential communication between n parties
- Secret keys need to be distributed
  - Need to ensure confidentiality and authenticity

Asymmetric Cryptography

- Less efficient
  - Rarely used to encrypt longer messages
- Lower number of private keys required
  - n keys required in order to enable pairwise confidential communication between n parties
- Only public keys need to be distributed
  - Need to ensure authenticity of public keys but not confidentiality

⚠ **In practice, the best of both worlds is often combined: asymmetric cryptography is used to establish secret keys which are then used for symmetric encryption and integrity protection**

Overview

- **Basic Number Theory**
  - Finite Fields, greatest common divisor, Fermat's theorem
  - Factorization
  - Discrete Logarithms

- **Digital signature schemes**
  - Intuition
  - RSA as signature scheme
  - Digital signature standard

- **Public Key Encryption Schemes**
  - Intuition
  - RSA as encryption scheme

- **Diffie-Helman Key Agreement**
  - Basic idea
  - Man-in-the-middle attack

- **Quantum Computers**

Quantum Computers and Traditional Asymmetric Schemes

- **1994 Peter Shor developed two polynomial time quantum algorithms**
  - A **factorization algorithm** that can factorize large compound numbers
  - A **discrete logarithm algorithm** that can compute the discrete logarithm x of $g^x \, mod \, p$ for a given prime number p and generator g
- **All classical asymmetric schemes can be broken with a large enough quantum computer, e.g.**
  - RSA signature scheme and RSA encryption scheme
  - DSA
  - Diffie-Helman Key Agreement
  - Elliptic Curve Cryptosystems lice ECDSA, ECDH
- **Lead to NIST calls for quantum secure encryption, signature, and key agreement schemes**
  - New post quantum algorithms selected in 2022

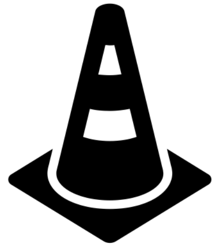
Quantum Computers and Traditional Symmetric Schemes

- **Grover's algorithm (1996) enables breaking symmetric encryption schemes like AES in $O(2^{n/2})$ where n is the bit length of the key**
  - Thus, it is currently believed that doubling the key size for symmetric encryption suffices

- **No known algorithm to find collisions for hash functions faster than on classical computers yet**
  - Cryptographic hash functions are currently believed not to be affected by quantum computers

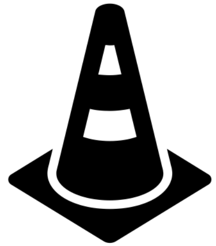
Summary

- **Asymmetric encryption schemes: confidentiality**
  - Most prominent example: RSA
    - Security depends on hardness of factorization
- **Digital signature schemes: integrity protection**
  - Most prominent examples: RSA, DSS
    - Security of DSS depends hardness of computing discrete logarithms
  - All signature schemes require hashing before signing
  - Provide non-repudiation and broadcast integrity protection
    - which cannot be provided by symmetric integrity protection via MACs



Summary

- **Diffie-Helman Key Agreement: establish secret key**
  - Can be used to establish a shared secret key for a symmetric scheme
  - Is itself an asymmetric scheme
  - Security depends on hardness of discrete logarithm
  - Is in its basic version vulnerable to a man-in-the-middle attack



- **All asymmetric schemes require authentic public keys**
  - Need to be able to obtain authentic copy of the public keys of other entities

- **All classical asymmetric schemes can be broken by large enough quantum computers**

References

- **Johannes Buchmann, Einführung in die Kryptographie, Springer Verlag 2016**
  - Chapter 8
- **W. Stallings, Cryptography and Network Security: Principles and Practice, 8th edition, Pearson 2022**
  - Chapter 9: Public Key Encryption and RSA
  - Chapter 10: Other Public Key Cryptosystems
    - Diffie Hellman
  - Chapter 13: Digital Signatures