Cybersecurity Fundamentals

Lecture 7

Asymmetric Cryptography

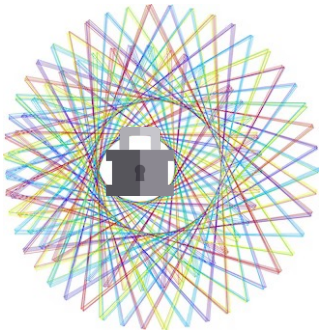
Thomas Zacharias

University of Glasgow

Symmetric Cryptography

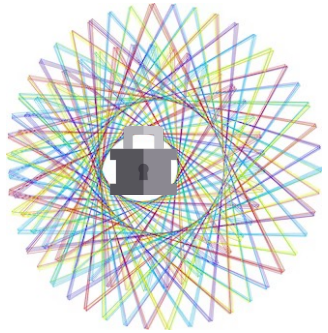
- The parties interact securely via the same cryptographic key.
- The key must be securely shared before interaction.
- Symmetric-key algorithms are fast.

Asymmetric Cryptography



- Secure interaction is via a public key and a corresponding private key.
- The private key never needs to be shared.
- Asymmetric-key algorithms are (currently) significantly slower.

In this lecture



- Mathematical background.
- Diffie-Hellman key exchange.
- Definition of public-key encryption.
- The RSA encryption algorithm.
- Hybrid encryption.
- Digital signatures.

Modular arithmetic

- **Euclid's division lemma:** Let x be an integer and n be a positive integer. There exist unique integers q and r such that $x = q \cdot n + r$ and $0 \leq r < n$.
- The integers q and r are called the quotient and the remainder, respectively.
- We say that x modulo n is equal to r and write

$$x \bmod n = r$$

- **Examples**:
  - $29 = 2 \cdot 12 + 5$, so: $29 \bmod 12 = 5$
  - $132 = 18 \cdot 7 + 6$, so: $132 \bmod 7 = 6$
  - $99 = 9 \cdot 11$, so: $99 \bmod 11 = 0$ (11 divides 99)
  - $-27 = -5 \cdot 6 + 3$, so: $-27 \bmod 6 = 3$

Modular arithmetic

- Consider the sequence of integers modulo 7:

-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10	11
3	4	5	6	0	1	2	3	4	5	6	0	1	2	3	4	5	6	0	1	2	3	4

mod 7

- The possible remainders $0, 1, 2, 3, 4, 5, 6$ appear periodically.
- We can first reduce modulo 7 and then add/subtract/multiply. E.g.,
  - $(3 + 8) \bmod 7 = (3 \bmod 7 + 8 \bmod 7) \bmod 7 = (3 + 1) \bmod 7 = 4 \bmod 7 = 4$
  - $(4 - 10) \bmod 7 = (4 \bmod 7 - 10 \bmod 7) \bmod 7 = (4 - 3) \bmod 7 = 1 \bmod 7 = 1$
  - $((-2) \cdot 5) \bmod 7 = ((-2) \bmod 7 \cdot 5 \bmod 7) \bmod 7 = (5 \cdot 5) \bmod 7 = 25 \bmod 7 = 4$

Modular arithmetic

- In general, for any positive integer n, we make the following observations:
  - By reducing (consecutive) integers modulo n, the remainders $0, 1, \ldots, n-1$ appear periodically.
  - The following rules hold for addition, subtraction, and multiplication:
    - $(x + y) \bmod n = (x \bmod n + y \bmod n) \bmod n$
    - $(x - y) \bmod n = (x \bmod n - y \bmod n) \bmod n$
    - $(x \cdot y) \bmod n = (x \bmod n \cdot y \bmod n) \bmod n$
- **Example**: compute $(1093028 \cdot 190301) \bmod 100$.
  - Since $1093028 \bmod 100 = 28$ and $190301 \bmod 100 = 1$, we have that

$$(1093028 \cdot 190301) \bmod 100 = (28 \cdot 1) \bmod 100 = 28$$

Modular arithmetic: exponentiation

- For integers x, e, n, where $n > 0$, we want to compute:

$$x^e \bmod n$$

- If we attempt to do the computation directly, then x^e becomes very large even for reasonable values of e.

Modular arithmetic: exponentiation

- For integers x, e, n, where $n > 0$, we want to compute:

$$x^e \bmod n$$

- We can perform modular exponentiations efficiently, by
  - Using the binary representation of $e = (e_{\kappa-1} \cdots e_1 e_0)_2 = \sum_{i=0}^{\kappa-1} e_i \cdot 2^i$
  - Applying the modular multiplication rule for the powers of 2.

$$\begin{aligned} x^2 \bmod n &= (x \cdot x) \bmod n = (x \bmod n \cdot x \bmod n) \bmod n \\ x^4 \bmod n &= (x^2 \cdot x^2) \bmod n = (x^2 \bmod n \cdot x^2 \bmod n) \bmod n \\ x^8 \bmod n &= (x^4 \cdot x^4) \bmod n = (x^4 \bmod n \cdot x^4 \bmod n) \bmod n \\ &\vdots \end{aligned}$$

  - Computing $x^e \bmod n$ as

$$\begin{aligned} x^e \bmod n &= x^{\sum_{i=0}^{\kappa-1} e_i \cdot 2^i} \bmod n = \\ &= \left(\prod_{i=0}^{\kappa-1} x^{e_i \cdot 2^i}\right) \bmod n = \\ &= \left(\prod_{i=0}^{\kappa-1} x^{e_i \cdot 2^i}\right) \bmod n = \left(\prod_{i:e_i=1} (x^{2^i} \bmod n)\right) \bmod n. \end{aligned}$$

Multiplicative inverses modulo n

- **Theorem**: Let x, n be integers, where $n > 0$. Let $\gcd(x, n)$ denote the greatest common divisor of x and n. There exist integers A, B such that

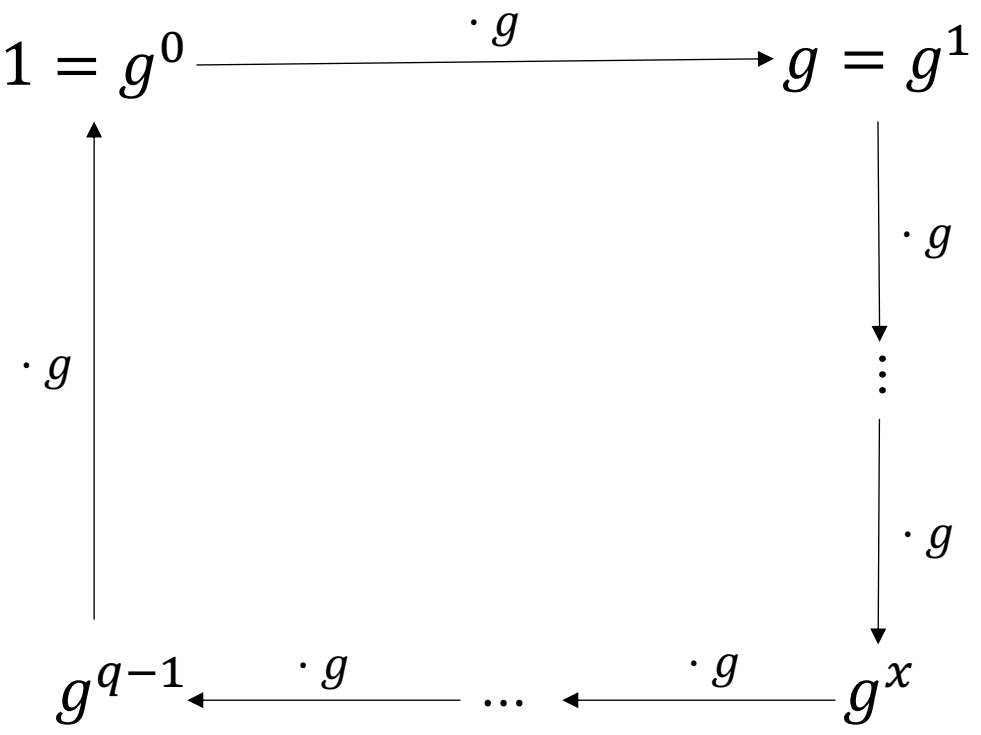
$$x \cdot A + n \cdot B = \gcd(x, n)$$

- The integers A, B can be computed efficiently (via the extended Euclidean algorithm).
- **Theorem**: Let x, n be integers, where $n > 0$ such that $\gcd(x, n) = 1$. Then, there exists an (efficiently computable) integer A, called the multiplicative inverse of x modulo n, such that

$$(x \cdot A) \bmod n = 1$$

Cyclic group of order q

- Let $\boldsymbol{G}$ be a finite set of q elements denoted by
$$\boldsymbol{G} = \{1 = g^0, g = g^1, g^2, \dots, g^{q-1}\}$$
- The "base" element g is called the generator of $\boldsymbol{G}$.
- Every element in $\boldsymbol{G}$ is generated by repeatedly applying the following operation (multiplication by g)



Cyclic group of order q

- Intuitively,
  - The element g^x is reached by "walking" x steps on the cycle, starting from g^0.
  - multiplying g^x by g repeatedly y times implies a "walk" of y steps on the cycle, starting from g^x.
  - every q steps, we are back to where we started!
- Thus, we can define the following arithmetic operations in $\boldsymbol{G}$:
  - Multiplication: $g^x \cdot g^y = g^{(x+y) \bmod q}$
  - Division: $g^x / g^y = g^{(x-y) \bmod q}$
  - Exponentiation: $(g^x)^y = g^{(x \cdot y) \bmod q}$

An example of a cyclic group in Cryptography

- Let p, q be primes such that $p = 2 \cdot q + 1$.
- Let a be an integer such that

$$\{1,2, \dots, p-1\} = \{a \bmod p, a^2 \bmod p, \dots, a^{p-1} \bmod p\}$$

- Then, the group

$$\boldsymbol{G} = \{a^{2i} \bmod p \mid i = 0, \dots, q-1\}$$

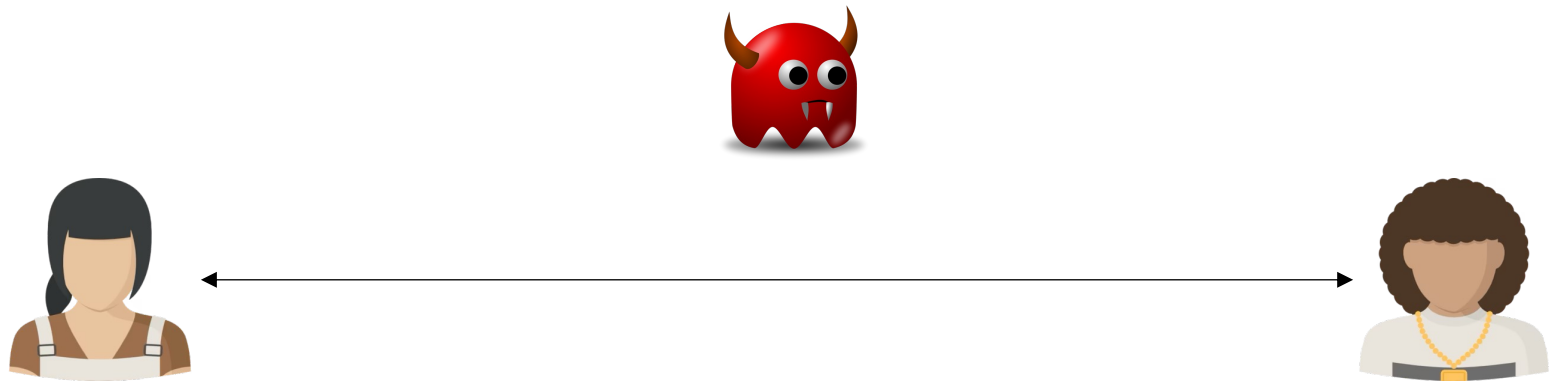
is a cyclic group of order q with generator $g := a^2 \bmod p$, under multiplication modulo p.

A fundamental question in Cryptography

How can two parties share a secret key without ever having a moment of privacy?

The status before 1976

- It was generally believed that secure communication could not be achieved without first sharing some secret information using a private channel.
- Secure key exchange over a public untrusted channel seemed infeasible.

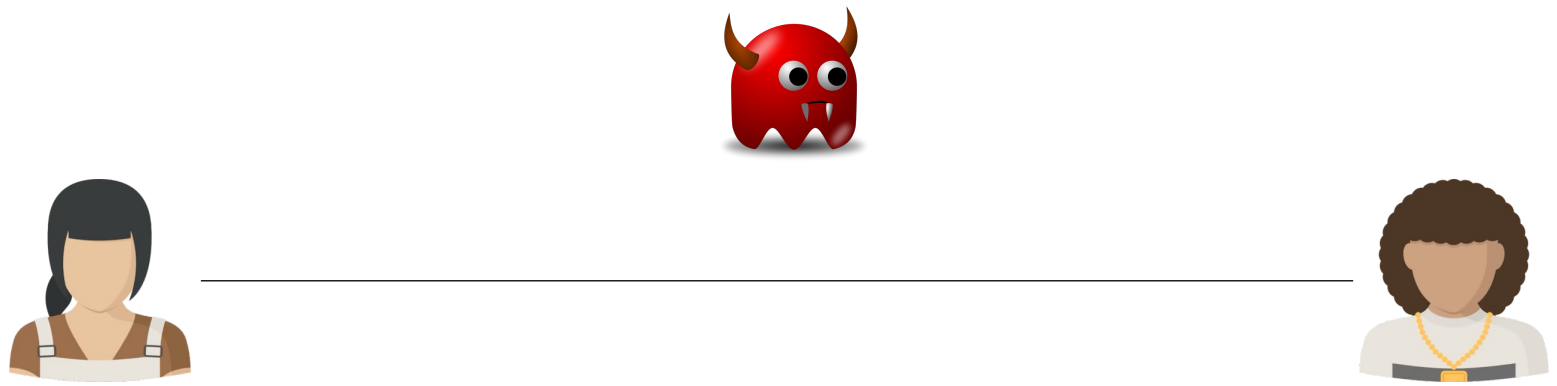


New Directions in Cryptography [Diffie and Hellman, 1976]

- There is often "asymmetry" in the world: certain actions can be easily performed but not easily reversed.
- Such kind of asymmetry can be used to achieve secure key exchange over a public channel in the presence of eavesdroppers.
- Introduction of the notion of Public-key Cryptography.

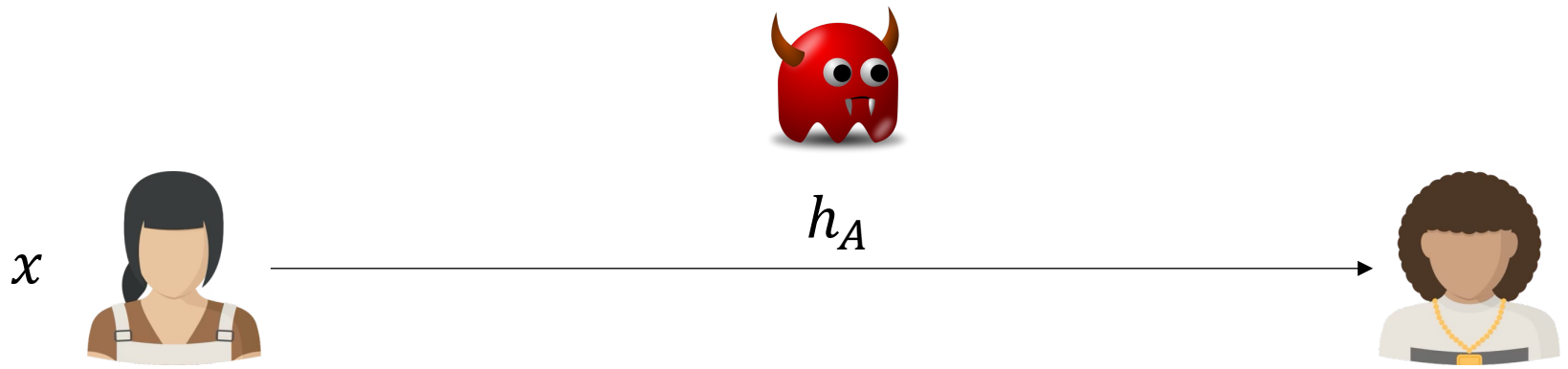
The Diffie-Hellman key exchange protocol

- Alice and Bob use a publicly known cyclic group $\boldsymbol{G}$ of order a prime q Let g be the generator of $\boldsymbol{G} = \{g^0, g, g^2, \ldots, g^{q-1}\}$.



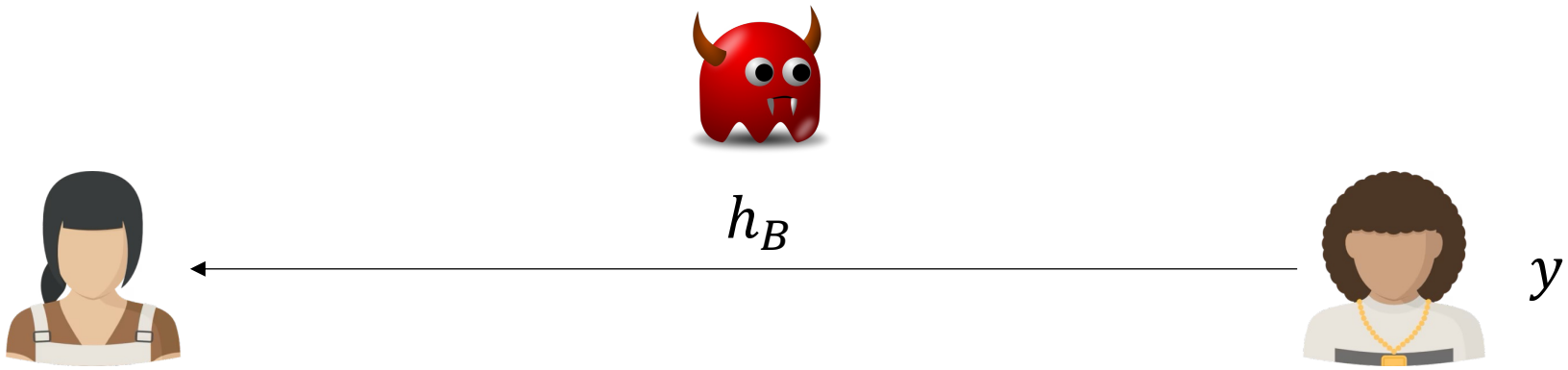
The Diffie-Hellman key exchange protocol

- Alice executes the following steps:
  1. She chooses a random integer x from $\{0, \ldots, q-1\}$.
  2. She computes the group element $h_A \leftarrow g^x$.
  3. She sends h_A to Bob through the public channel.



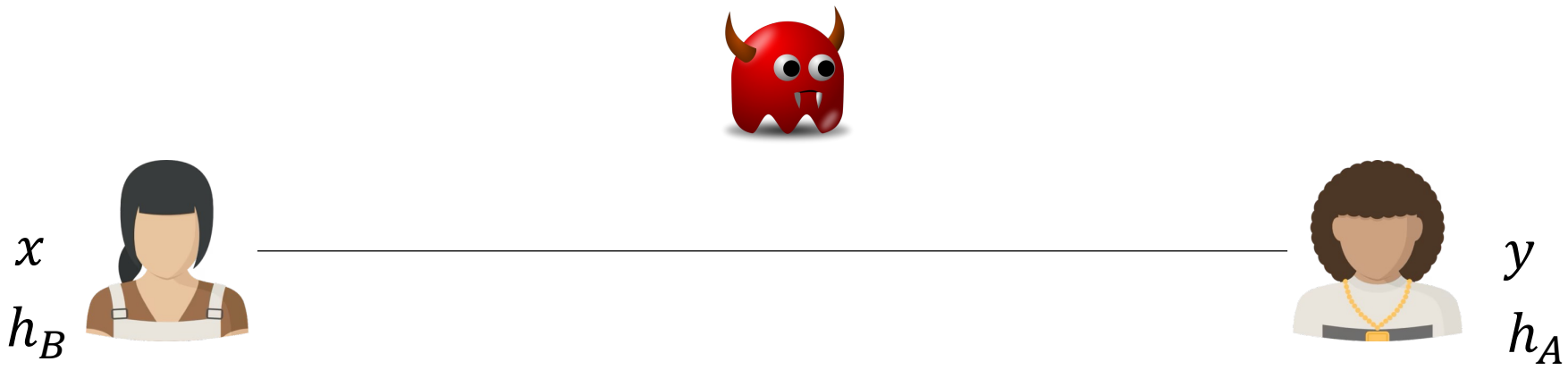
The Diffie-Hellman key exchange protocol

- Bob executes the following steps:
  1. He chooses a random integer y from $\{0, \ldots, q-1\}$.
  2. He computes the group element $h_B \leftarrow g^y$.
  3. He sends h_B to Alice through the public channel.



The Diffie-Hellman key exchange protocol

- Alice computes the key $k_A \leftarrow (h_B)^x$.
- Bob computes the key $k_B \leftarrow (h_A)^y$.



Correctness of the Diffie-Hellman protocol

- Alice and Bob agree on the same key!

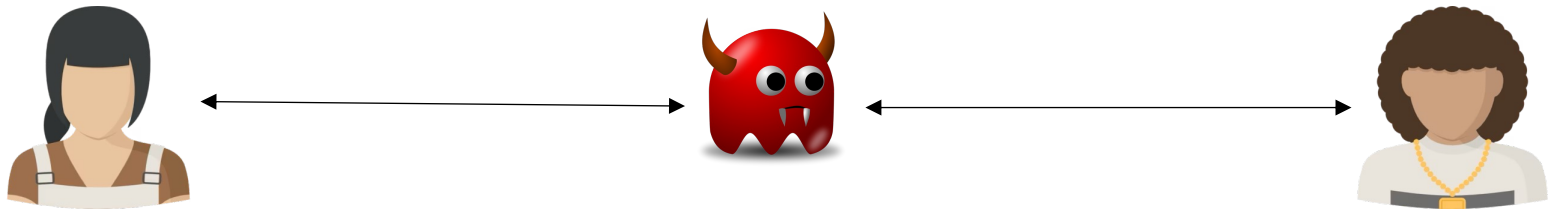
$$k_A = (h_B)^x = (g^y)^x = g^{y \cdot x \bmod q} =$$
$$= g^{x \cdot y \bmod q} = (g^x)^y = (h_A)^y = k_B$$

Security of the Diffie-Hellman protocol against an eavesdropper (passive adversary)

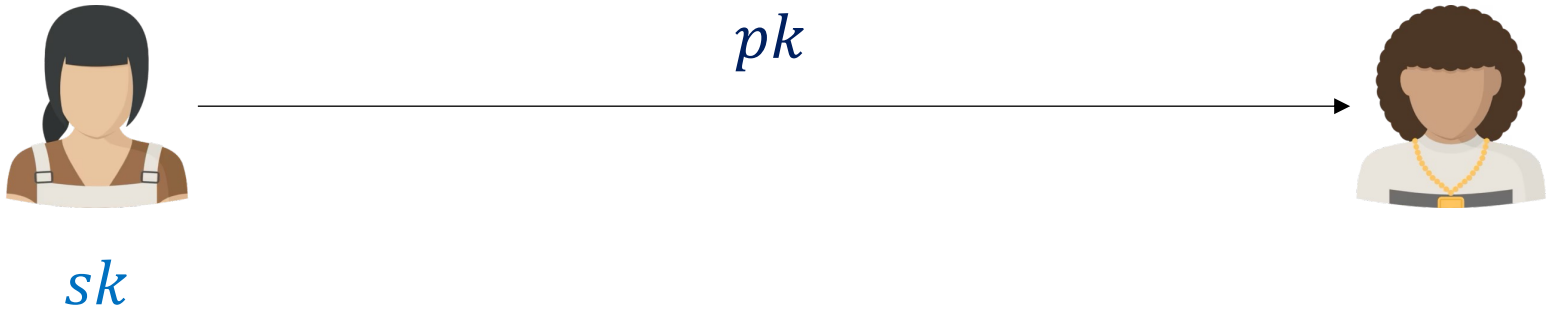
- A passive adversary observes the transmitted values g^x and g^y.
- The adversary cannot learn information about the shared key because in a cryptographic cyclic group, the following problems are considered infeasible when the group order q is sufficiently large:
  - **Discrete Logarithm Problem (DL)**: given g^x (or g^y), it is hard to compute x (or y).
  - **Decisional Diffie-Hellman Problem (DDH)**: given g^x and g^y, it is hard to distinguish $g^{x \cdot y \bmod q}$ from a random element in $\boldsymbol{G}$.

Insecurity of the Diffie-Hellman protocol against active adversary

- **Man-in-the-middle attacks**: the adversary is intercepting and modifying messages sent from one party to the other.
- The Diffie-Hellman protocol is not secure against man-in-the-middle attacks, because it does not provide authentication of the participants:
  - A man-in-the-middle adversary can act in such a way that Alice and Bob terminate the protocol with different keys k_A and k_B, both known to the adversary.
  - Neither Alice nor Bob can detect that any attack was carried out.



Public-key encryption

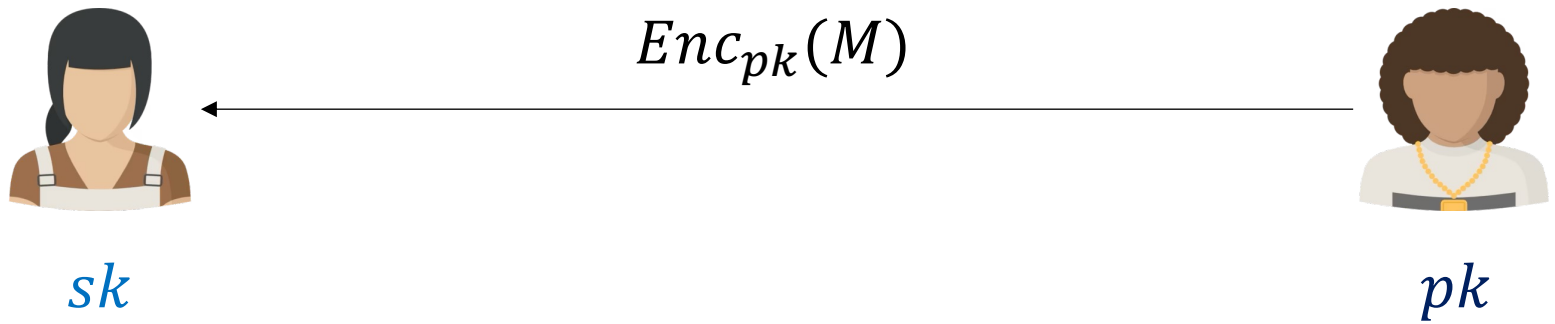


- Alice generates a pair of a private key sk and a public key pk.
- The public key becomes available to Bob (e.g., either via an authenticated public channel, or because Alice publishes it on her website).

Public-key encryption

- Bob wans to send a message M through an insecure channel.

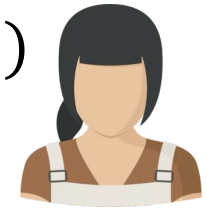
Public-key encryption



Public-key encryption

$Dec_{sk}(Enc_{pk}(M))$

M



sk

pk

Definition of public-key encryption: Syntax and Correctness

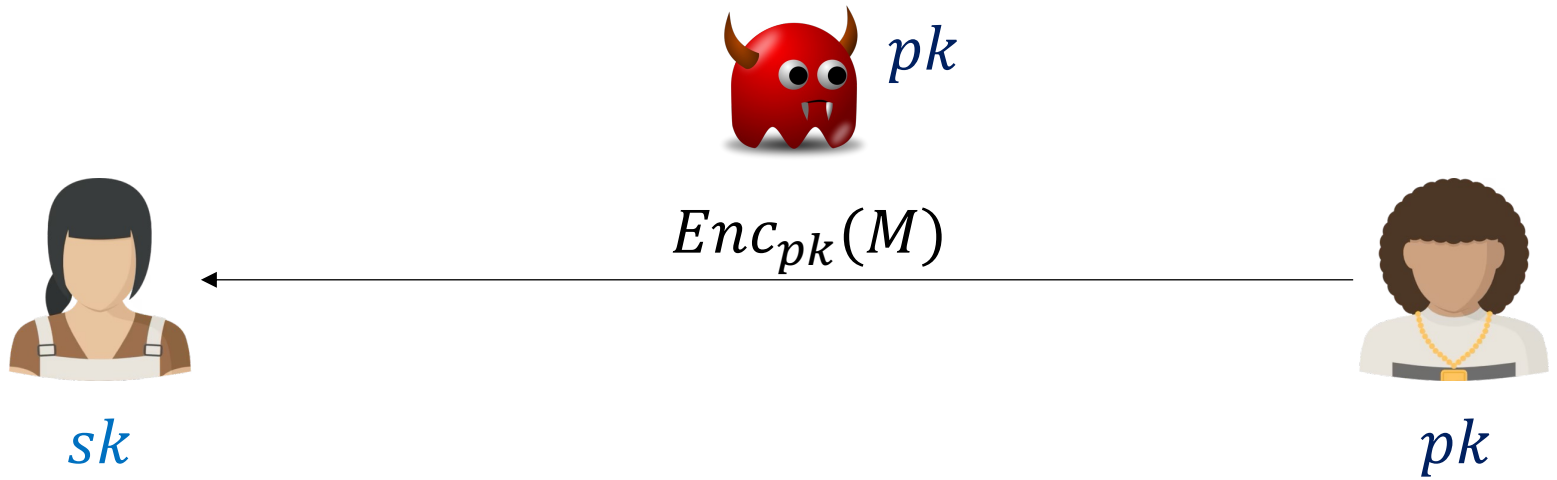
- A public-key encryption scheme is a triple of algorithms as follows:
  - A key-generation algorithm Gen that outputs a private key sk and a public key pk .
  - An encryption algorithm Enc that takes as input a public key pk and a message (plaintext) M and outputs a ciphertext C. We write $C \leftarrow Enc_{pk}(M)$.
  - A decryption algorithm Dec that takes as input a private key sk and a ciphertext C and outputs a message M. We write $M \leftarrow Dec_{sk}(C)$.

- **Correctness**: for every pair of keys (sk, pk) and every message M, it holds that

$$M \leftarrow Dec_{sk}\left(Enc_{pk}(M)\right)$$

Definition of public-key encryption: Security (Indistinguishability under chosen plaintext attack)

A public-key encryption scheme achieves indistinguishability under chosen plaintext attack (IND-CPA security), if it leaks no information about Bob's message M against any adversary that
(1) observes the communication channel and
(2) obtains the public key pk
(so, it can create encryptions for messages of its choice).



The RSA public-key encryption scheme

- Introduced by Rivest, Shamir, and Adleman in 1977.
- One of the oldest public-key cryptosystems, still widely used today.
- Core design idea:

Let p, q be distinct primes. Let $n = p \cdot q$ and $\phi(n) = (p-1) \cdot (q-1)$.
For any integer $e > 0$, define the function
$f_e(x) = x^e \bmod n$
where $M \in \{0,1,\dots,n-1\}$. If $\gcd(e, \phi(n)) = 1$, then the following hold:

1. f_e is a permutation.
2. If d is the multiplicative inverse of e modulo $\phi(n)$,
then f_d is the inverse of f_e.

The RSA public-key encryption scheme

- **RSA key-generation algorithm**: on input 1^{κ}
  1. Choose two distinct random κ-bit primes p and q.
  2. Compute $n = p \cdot q$ and $\phi(n) = (p-1) \cdot (q-1)$.
  3. Choose $1 < e < \phi(n)$ such that $\gcd(e, \phi(n)) = 1$.
  4. Compute d as the multiplicative inverse of e modulo $\phi(n)$.
  5. Set the private key as $sk := (d, n)$.
  6. Set the public key as $pk := (e, n)$.
  7. Output (sk, pk).

The RSA public-key encryption scheme

- **RSA encryption algorithm**: on input a public key (e, n) and a message M (encoded as an integer such that $0 \leq M < n$), compute the ciphertext

$$C \leftarrow M^e \bmod n$$

- **RSA decryption algorithm**: on input a private key (d, n) and a ciphertext C (also an integer such that $0 \leq C < n$), compute the message

$$M \leftarrow C^d \bmod n$$

Correctness of RSA encryption

- Since the function $f_d(x) = x^d \bmod n$ is the inverse of the function $f_e(x) = x^e \bmod n$, it holds that

$$Dec_{sk}\left(Enc_{pk}(M)\right) = (M^e \bmod n)^d \bmod n = f_d\left(f_e(M)\right) = M$$

Security of RSA encryption

- Security of RSA depends on the hardness of factoring the composite modulus n into p and q.
  - If the adversary learns p and q, then it can compute $\phi(n)$ and hence, the multiplicative inverse of e modulo $\phi(n)$ (i.e., the private key d).
  - Factoring large integers is believed to be hard (in 2020, factorisation of a 829-bit RSA modulus was completed).
  - Choosing 2048-bit keys is considered a safe choice for the near future.
- Since RSA is deterministic, it cannot be IND-CPA secure!
  - Repeated messages encrypted under the same public key will result in repeated ciphertexts.
  - It can be used for the encryption of random unpredictable messages (e.g., a randomly chosen symmetric key).

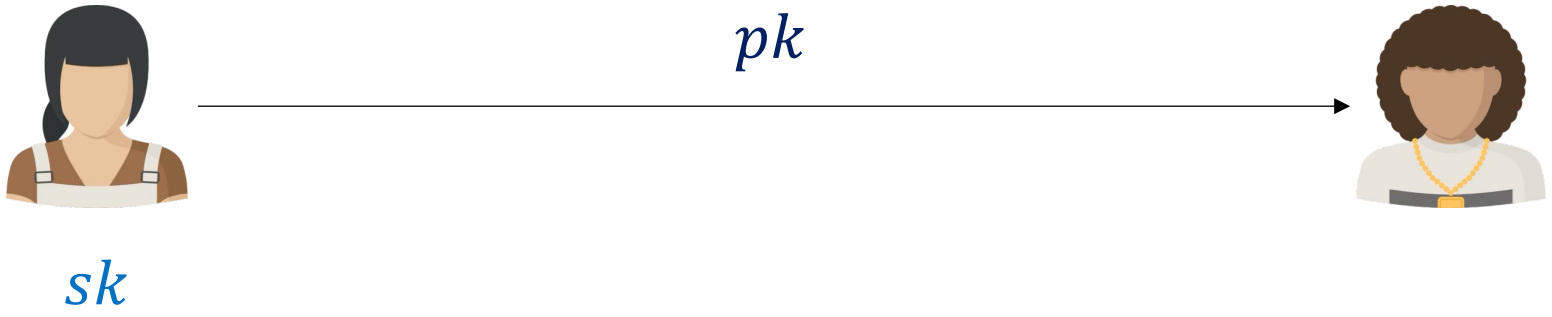
Building secure RSA-based public-key encryption

- **Padded RSA**:
  - Pad the message with a random string before encryption.
  - PKCS #1 v1.5 [1993]:
    - when the (random string length/message length) ratio is sufficiently large, then we can conjecture that it is IND-CPA secure.
    - vulnerable against stronger adversaries (cf. chosen ciphertexts attacks, where the adversary has also black-box access to the decryption algorithm for ciphertexts of its choice).
- **Optimal Asymmetric Encryption Padding (OAEP)**:
  - Standardised as PKCS #1 v2.0 [1998]:
  - Proven to be secure even against stronger adversaries (chosen ciphertext attacks).

Symmetric encryption (SKE) vs Public-key encryption (PKE)

- In SKE, any party that has the secret key can play the role of sender or receiver, but the key must be shared in advance.
- In PKE, the private key never needs to be shared, but the roles of the sender and the receiver are not interchangeable.
- PKE is (roughly) 2 to 3 orders of magnitude slower than SKE.
- SKE can encrypt messages of arbitrary length.

Hybrid encryption



- Alice generates a pair of a private key sk and a public key pk.
- The public key becomes available to Bob (e.g., either via an authenticated public channel, or because Alice publishes it on her website).

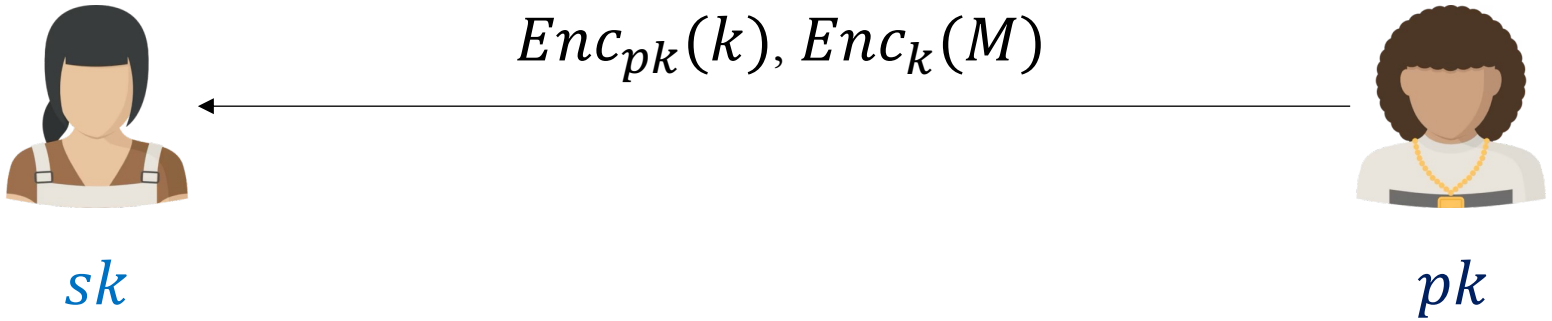
Public-key encryption

- Bob wans to send a message M of arbitrary length, so that encryption/decryption of M is fast.

Hybrid encryption

- Bob chooses a random symmetric encryption key k.

Hybrid encryption



- Bob sends to Alice:
  (i) an encryption of k under pk, and
  (ii) an encryption of M under k.

Hybrid encryption

- Alice first obtains k by decrypting $Enc_{pk}(k)$ with her private key sk.

Hybrid encryption

$Dec_k(Enc_k(M))$

sk

pk

M

- Alice then obtains M by decrypting $Enc_k(M)$ with her private key k.

Digital signatures

- The asymmetric cryptographic primitive used for validating the authenticity (integrity) of a message.
- Alice can sign a message using her private key so that anyone who knows the corresponding public key can verify that (i) the message originated from Alice and (ii) was not modified during transmission.
- Digital signatures provide non-repudiation, i.e., the signer cannot cannot successfully claim they did not sign a message.
- Digital signatures do not provide confidentiality.

Definition of digital signatures: Syntax, Correctness, and Security

- A digital signature scheme is a triple of algorithms as follows:
  - A key-generation algorithm Gen that outputs a private key sk and a public key pk .
  - A signing algorithm $Sign$ that takes as input a private key sk and a message (plaintext) M and outputs a signature σ. We write $\sigma \leftarrow Sign_{sk}(M)$.
  - A verification algorithm $Vrfy$ that takes as input a public key pk and, a message M, and a signature σ and outputs a bit b. We write $b := Vrfy_{pk}(M, \sigma)$.

- **Correctness**: for every pair of keys (sk, pk) and every message M, it holds that

$$Vrfy_{pk}\big(M, Sign_{sk}(M)\big) = 1$$

- **Security (Existential Unforgeability)**: no adversary that has pk can generate a valid signature on a new message that was not previously signed by the signer using sk.

RSA Full Domain Hash (RSA-FDH) signatures

- **RSA key-generation algorithm**: on input 1^{κ}
  1. Choose two distinct random κ-bit primes p and q.
  2. Compute $n = p \cdot q$ and $\phi(n) = (p-1) \cdot (q-1)$.
  3. Choose $1 < e < \phi(n)$ such that $\gcd(e, \phi(n)) = 1$.
  4. Compute d as the multiplicative inverse of e modulo $\phi(n)$.
  5. Set the private key as $sk := (d, n)$.
  6. Set the public key as $pk := (e, n)$.
  7. Output (sk, pk).

RSA Full Domain Hash (RSA-FDH) signatures

- **RSA signing algorithm**: On input a private key (d, n) and a message M, compute the signature

$$\sigma \leftarrow H(M)^d \bmod n$$

where $H(\cdot)$ is a publicly known hash function in the range $\{0,1, \ldots, n-1\}$.

- **RSA verification algorithm**: the algorithm considers the same hash function $H(\cdot)$. On input a public key (e, n), a message M, and a signature σ, output 1 if and only if

$$\sigma^e \bmod n \stackrel{?}{=} H(M)$$

RSA Full Domain Hash (RSA-FDH) signatures

- **Correctness**: Since the function $f_d(x) = x^d \bmod n$ is the inverse of the function $f_e(x) = x^e \bmod n$, it holds that

$$(Sign_{sk}(M))^e \bmod n = (H(M)^d \bmod n)^e \bmod n = f_e\left(f_d\left(H(M)\right)\right) = H(M)$$

$$\Rightarrow Vrfy_{pk}\left(M, Sign_{sk}(M)\right) = 1.$$

- **Security**: RSA-FDH achieves existential unforgeability under similar assumptions as RSA encryption, along with the assumption that $H(\cdot)$ "behaves" as a random function.

End of Lecture 7

The slides content is related to Sections 2.3, 21.3, and 21.4 of "Computer Security Principles and Practice (3rd Edition)" by Stallings and Brown