CS458: Introduction to Information Security

Notes 4: Public-Key Cryptography

Yousef M. Elmehdwi

Department of Computer Science

Illinois Institute of Technology

yelmehdwi@iit.edu

February 7th 2024

Slides: Modified from Computer Security: Principles and Practice, 4th Edition, Christof Paar and Jan Pelzl, Stephen R. Tate UNC Greensboro & Ewa Syta

Padding

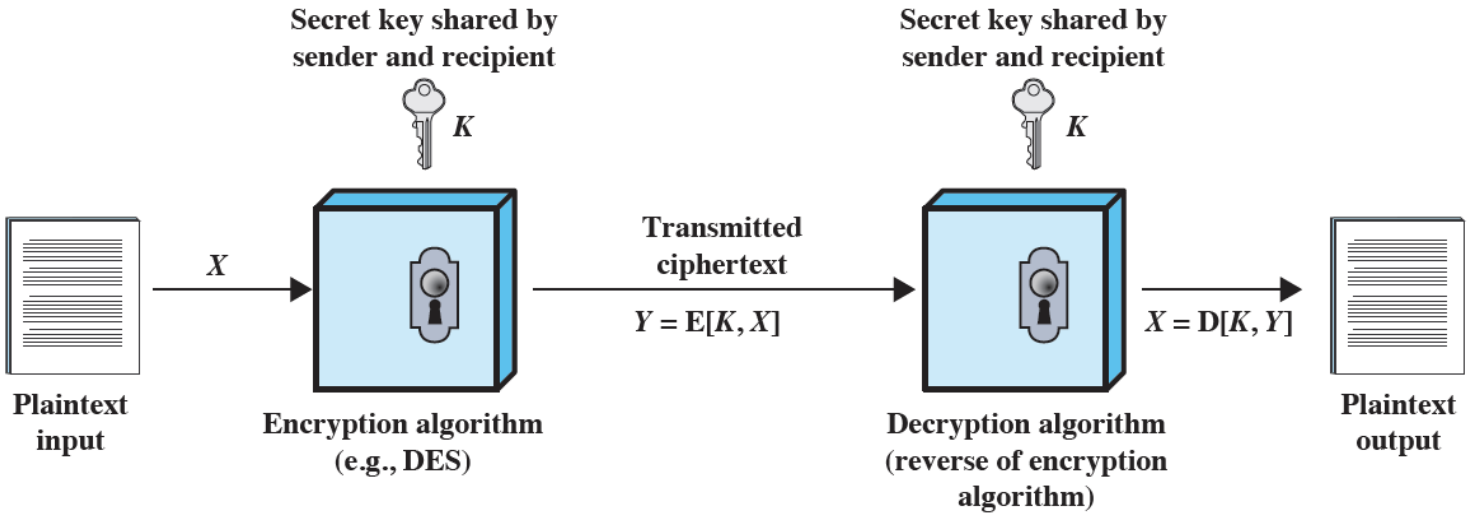
- *How do we encrypt data that is less than one block using block cipher?*
- ECB and CBC modes require encrypting full blocks of plaintext!
- What if you have `192` bits of plaintext with AES/CBC? You'd need `128` bits plus `64` bits.
- Padding is used in block ciphers to fill up blocks with padding bytes.
- AES uses `128`-bit (`16` bytes) blocks, and DES uses `64`-bit (`8` bytes) blocks.
- One common padding technique is `PKCS#7/PKCS#5`[1], which is a standardized padding scheme defined by the PKCS organization.
  - Calculate the number of padding bytes needed, denoted as c.
  - Add c bytes, each with a value equal to c.
  - For example, if we have `32`-bit blocks in hexadecimal: *42 1a 49 c3 | 21* becomes *42 1a 49 c3 | 21 03 03 03*.
  - Note: This padding technique only works for padding full bytes.
  - `PKCS#5` padding is identical to `PKCS#7` padding, but it's specifically defined for block ciphers with a `64`-bit block size.
  - In practice, the two can be used interchangeably.

[1] *Public Key Cryptography Standards*

Outline

- Principles of Asymmetric Cryptography
  - key concept, key differences from symmetric cryptography and main functionalities
- Practical Aspects of Public-Key Cryptography
  - Key management, key size and applications.
- Important Public-Key Algorithms

Symmetric Cryptography revisited



- Two properties of symmetric (secret-key) crypto-systems:
  - The same *secret key K* is used for encryption and decryption
  - Encryption and Decryption are very similar (or even identical) functions

- *There are two basic principles of any cryptosystem: confidentiality and authenticity. Symmetric cryptosystem has a problem associated with these two principles*

Symmetric Cryptography: Shortcomings

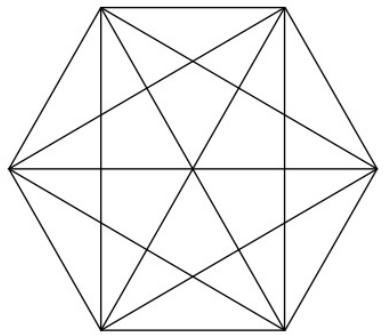
- Symmetric algorithms, e.g., AES or 3DES, are very secure, fast & widespread **but**:

1. Key distribution problem: The secret key must be transported securely
   - i.e., when Alice and Bob communicate using a symmetric system, they need to securely exchange their shared key k_{ab}
2. Key management: In a network, each pair of users requires an individual key

   $\rightarrow$ n users in the network require $\frac{n \times (n-1)}{2}$ keys, each user stores $(n-1)$ keys
   - If Alice wants to talk to Bob, Carol and Dave, she needs to exchange and maintain k_{ab}, k_{ac}, and k_{ad}

- Example: 6 users (nodes)

  $\frac{6 \times 5}{2} = 15$ keys (edges)



Symmetric Cryptography: Shortcomings

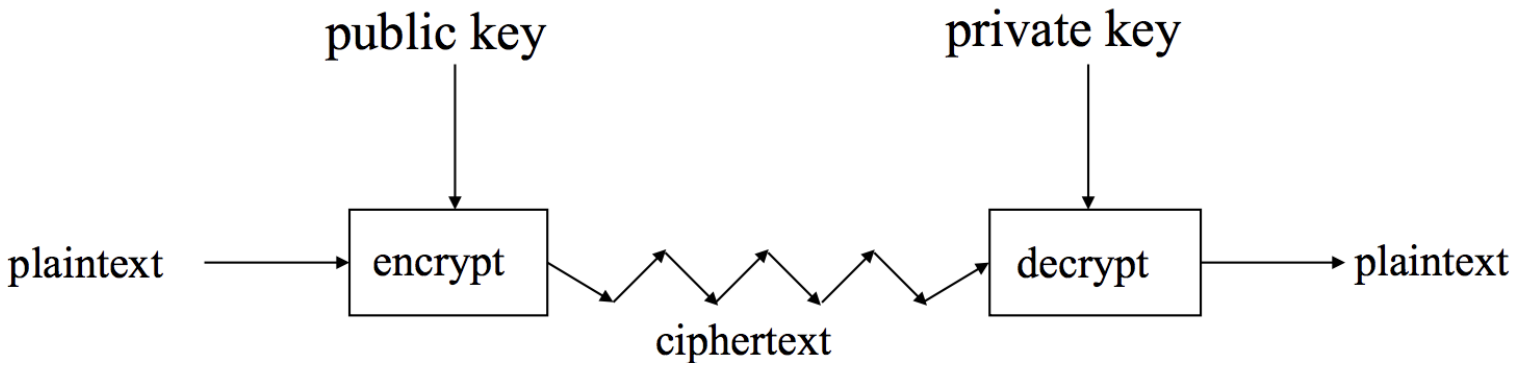
3. No Protection Against Cheating by Alice or Bob: Alice or Bob can **cheat each other**, because they have identical keys.
    - Who is the author of a message encrypted with k_{ab}, a key Alice and Bob share?
        - Example: Alice can claim that she never ordered a TV on-line from Bob (he could have fabricated her order). To prevent this: "non-repudiation"

- *The public key cryptosystem is successful in achieving both these principles; i.e., confidentiality and authenticity.*

Public-Key Encryption Structure

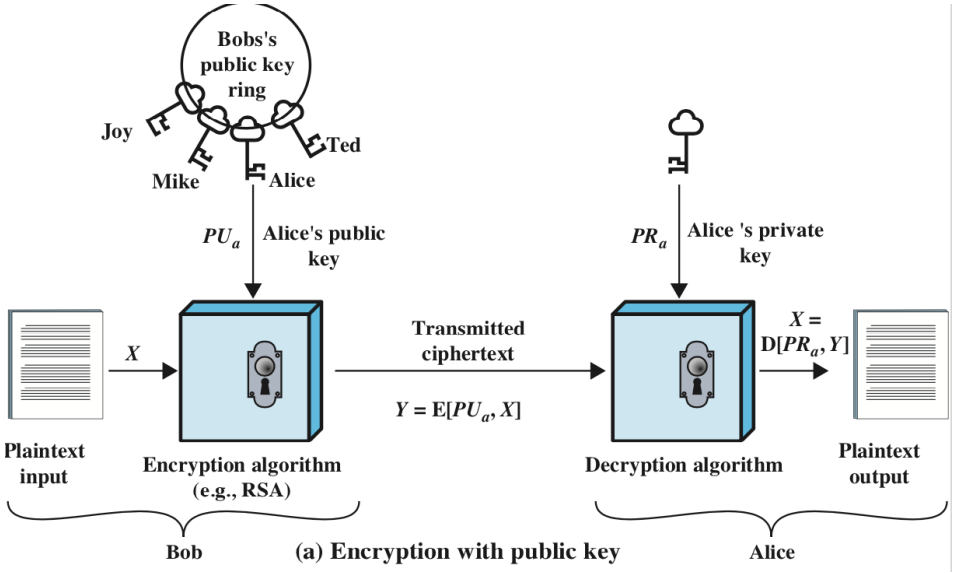
- Publicly proposed by Diffie and Hellman in 1976
- Based on mathematical functions rather than on simple operations on bit patterns
- Asymmetric
  - Uses two separate keys
  - Public key and private key
  - Public key is made public for others to use

Public Key Crypto



- Two keys:
  - Private key known only to owner
  - Public key available to anyone
  - One key pair per person
    - $O(N)$ keys

Public Key Crypto



- Plaintext: Readable message or data that is fed into the algorithm as input
- Encryption algorithm: Performs transformations on the plaintext
- Public and private key: Pair of keys, one for encryption, one for decryption
- Ciphertext: Scrambled message produced as output
- Decryption algorithm: Produces the original plaintext

Public Key Crypto

- In public key cryptography, an encryption key (which could be the public or private key) is used to encrypt a plaintext message and convert it into an encoded format known as ciphertext.
- Then the other key is used as a decryption key to decrypt this ciphertext so that the recipient can read the original message.
- *In short, the main difference between a public key vs private key is that one encrypts while the other decrypts.*

Uses of Public Key Crypto

- *Q: What is the Usage of Public Key Cryptography?*
  - One of the unique advantages of asymmetric encryption using public key pairs is the ability to ensure both the security of encrypted messages and the identity of the sender.
- Encryption
  - Suppose we encrypt m with Bob's public key.
  - Bob's private key can decrypt c to recover m.
  - *Why public key for encryption?*

- *To ensure the identity of the sender, the order of using the key pairs can be flipped.*
- Digital Signatures[1]
  - Bob signs by "encrypting" with his private key.
  - Anyone can use Bob's public key to verify the signature (by decrypting with public key).
  - Like a handwritten signature, but way better...
  - *Why private key for digital signatures?*

[1] *Digital signatures to assure all parties that a particular message has been sent from a particular person*

Security of the Keys

- In asymmetric encryption, two keys are involved:
  - Private key: Known only to the individual.
  - Public key: Available to anyone.
  - The fundamental principle behind asymmetric encryption is that while one key is public, the other remains private and cannot be easily computed.
- This security mechanism relies on the concept of a "trapdoor one-way function":
  - **"One-way"** means that it's easy to compute in one direction but challenging to compute in the reverse direction.
    - It's easy to calculate $f(x)$ from x.
    - However, it's hard to invert: finding x from $f(x)$.
    - For example, given p and q, computing the product $N = pq$ is straightforward, but finding p and q from N is challenging.
  - A trapdoor one-way function has an additional property: with certain knowledge, it becomes easy to invert, allowing the calculation of x from $f(x)$.
    - This "trapdoor" is utilized when creating key pairs. Possessing it enables the reversal of the encryption process.
    - An analogy: Think of a "trapdoor" as falling through effortlessly, but climbing back out is exceedingly difficult without the specific means, akin to needing a ladder.

Non-repudiation and Digital Signatures

- Nonrepudiation refers to the property of a cryptographic system that ensures that the sender of a message cannot deny having sent the message.
- Digital signatures are a method of achieving nonrepudiation by verifying the authenticity of the sender of a message.
  - A digital signature is a mathematical value that is generated based on the message and the sender's private key.
  - When the message is received, the digital signature can be verified using the sender's public key.
  - If the digital signature is valid, it means that the message has not been tampered with and was indeed sent by the claimed sender.

Applications for Public-Key Cryptosystems

- We can classify the use of public-key cryptosystems into three categories:
  - Symmetric Key Distribution (e.g., Diffie-Hellman key exchange, RSA) without a pre-shared secret (key)
  - Nonrepudiation and Digital Signatures[1] (e.g., RSA, DSA or ECDSA) to provide message integrity
    - i.e., Integrity/authentication[2]: encipher using private key, decipher using public one
  - Encryption (e.g., RSA / Elgamal)
    - Confidentiality: encipher using public key, decipher using private key
    - Disadvantage: Computationally very intensive (much slower than symmetric Algorithms!)

Major Public Key Algorithms

Algorithm	Digital Signature	Symmetric Key Distribution	Encryption of Secret Keys
RSA	Yes	Yes	Yes
Diffie-Hellman	No	Yes	No
DSS	Yes	No	No
Elliptic Curve	Yes	Yes	Yes

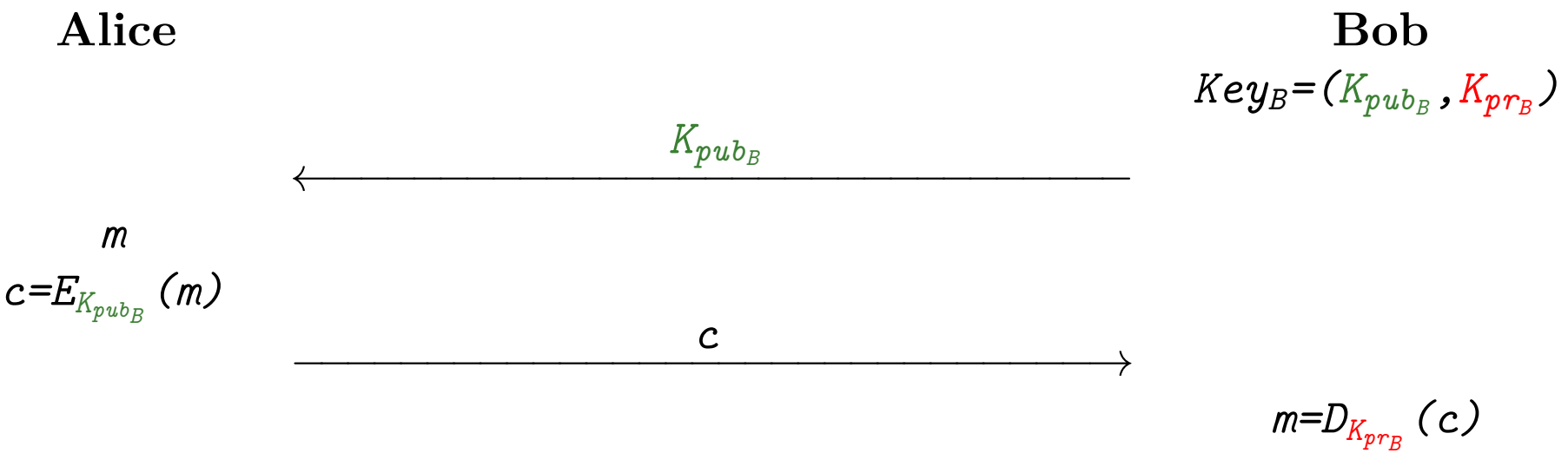
[1] *Digital signatures are used to detect unauthorized modifications to data and to authenticate the identity of the signatory.*

[2] *Message integrity means that a message has not been tampered with or altered.*
The concept of Authentication serves to provide proof that the other side of a communication is indeed who they claim to be, and who you intend for them to be.

Requirements for Public-Key Cryptosystems

- Computationally easy
  - for a party to generate a key pair
  - for sender knowing public key to encrypt messages
  - for receiver knowing private key to decrypt ciphertext
- Computationally infeasible
  - for an opponent knowing only the public key to determine the private key
  - for an opponent knowing the public key and a ciphertext to recover the original message
- Useful if either key can be used for each role (not necessary for all public key applications)

Basic Protocol for Public-Key Encryption



- *Key Distribution Problem solved (at least for now; public keys need to be authenticated)*

Basic Key Transport Protocol

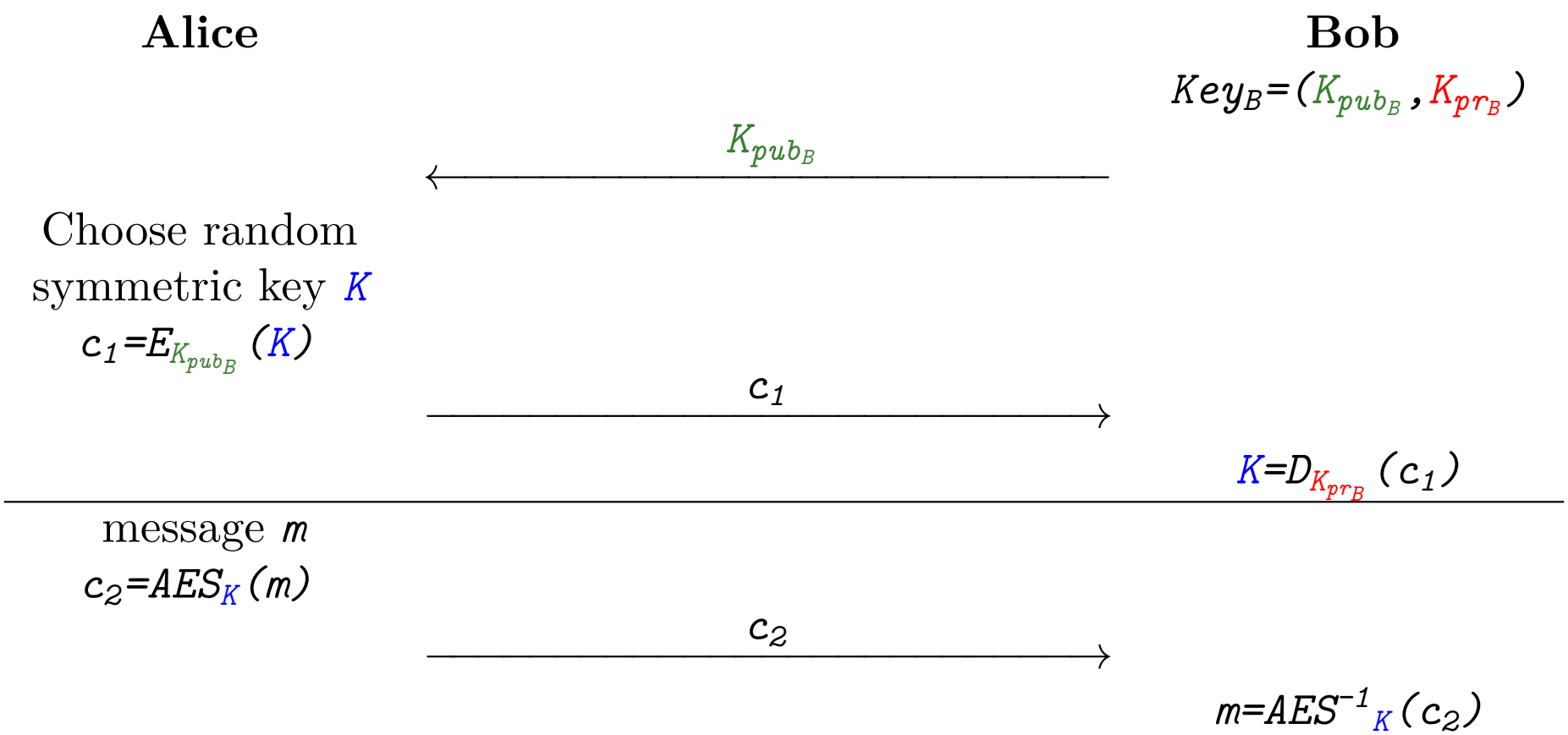
- In practice, key transport protocols often use a hybrid approach, combining both asymmetric and symmetric algorithms.
  - Examples: SSL/TLS protocol[1] for secure Web connections, or IPsec, the security part of the Internet communication protocol.

1. Key exchange (for symmetric schemes) and digital signatures are performed with (slow) asymmetric algorithms
   - *Asymmetric algorithms are used to securely exchange a shared symmetric key between the parties involved. This shared key will be used for efficient symmetric encryption.*
2. Encryption of data is carried out using (fast) symmetric ciphers.
   - *Symmetric encryption algorithms, such as block ciphers or stream ciphers, are employed to encrypt the actual data being transmitted. Symmetric ciphers are efficient for encrypting large volumes of data quickly.*

[1] *SSL/TLS (Secure Sockets Layer /Transport Layer Security) is a cryptographic protocol designed to provide communications security over a computer network. The protocol is widely used in applications such as email, instant messaging, and voice over IP, but its use as the Security layer in HTTPS remains the most publicly visible.*

Basic Key Transport Protocol

- Example: Hybrid protocol with AES as the symmetric cipher



How to build Public-Key Algorithms

- Asymmetric schemes are based on a one-way function $f()$:
  - Computing $y = f(x)$ is computationally easy
  - Computing $x = f^{-1}(y)$ is computationally infeasible
- One way functions are based on **mathematically hard problems**. Three main families:
  - Integer-Factorization Schemes:
    - Several public-key schemes are based on the fact that it is difficult to factor large integers, e.g., RSA
    - Given a composite integer n, find its prime factors
    - (Multiply two primes: easy)
  - Discrete Logarithm Schemes
    - Several algorithms, such as Diffie-Hellman key exchange, Elgamal, Digital Signature Algorithm (DSA)
    - Given a, y and n, find x such that $a^x \equiv y \bmod n$
    - (Exponentiation a^x: easy)
  - Elliptic Curves (EC) Schemes
    - Generalization of discrete logarithm and are widely used in modern cryptography for their efficiency and security.
    - e.g., Elliptic Curve Diffie-Hellman key exchange (ECDH) and the Elliptic Curve Digital Signature Algorithm (ECDSA)

Key Lengths and Security Levels

- Key size, key length, or key space in cryptography refers to the number of bits in a cryptographic key used by an algorithm.
- An algorithm is said to have a security level of n *bit* if the best known attack against the algorithm would require 2^n computational steps to break it

Key Lengths and Security Levels

- Symmetric Algorithms:
  - In symmetric cryptography, the security level directly corresponds to the key length.
  - Achieving an n-bit security level requires a key of length n bits.
  - For example, to achieve 128-bit security, a 128-bit key is used for symmetric encryption.
- Asymmetric Algorithms:
  - In asymmetric cryptography, the relationship between key size and security is more complex.
  - The strength of public key cryptosystems relies on the difficulty of specific mathematical problems, like factoring large numbers or solving discrete logarithm problems.
  - While these problems are time-consuming to solve, they are typically faster than trying all possible keys by brute force.
  - Consequently, asymmetric keys must be longer (contain more bits) compared to symmetric keys to achieve equivalent resistance to attack.

Key Lengths and Security Levels

- In symmetric schemes
  - Keys are randomly selected,
  - Key lengths varying between 128 and 256 bits, depending on the required level of security
    - *Since 2015, NIST guidance: "the use of keys that provide less than 112 bits of security strength for key agreement is now disallowed."*
- In asymmetric schemes
  - Mathematical relationship between the public and private keys (there is a mathematical pattern between the two keys).
  - This pattern can potentially be exploited by attackers to crack the encryption. Therefore, asymmetric keys need to be much longer to present an equivalent level of security.
    - 80-bit symmetric key and a 1024-bit asymmetric key (RSA) are equivalent in strength.
    - 112-bit symmetric key and a 2048-bit asymmetric key (RSA) offer roughly similar levels of security.
    - 128-bit symmetric key and a 3072-bit asymmetric key (RSA) offer roughly similar levels of security.
  - Key lengths varying between 1,024 and 4,096 bits depending on the required level of security
    - *Since 2015, NIST recommends a minimum of 2048-bit keys for RSA (2048-bit keys are sufficient until 2030)*

General Facts about Public Key Systems

- Public Key Systems are much slower than Symmetric Key Systems
  - Public-key cryptography is computationally intensive compared to symmetric algorithms because of the mathematical operations involved in the encryption and decryption process.
  - Generally used in conjunction with a symmetric system for bulk encryption
- Public Key Systems are based on “hard” problems
  - Factoring large composites of primes, discrete logarithms, elliptic curves
- Only a handful of public key systems perform both encryption and signatures

Asymmetric Encryption Algorithms

- RSA (Rivest, Shamir, Adleman)
  - Developed in 1977
  - Most widely accepted and implemented approach to public-key encryption
  - Block cipher in which the plaintext and ciphertext are integers between `0` and `n-1` for some `n`.
- Diffie-Hellman key exchange algorithm
  - Enables two users to securely reach agreement about a shared secret that can be used as a secret key for subsequent symmetric encryption of messages
  - Limited to the exchange of the keys

Asymmetric Encryption Algorithms

- Digital Signature Standard (DSS)
  - Originally proposed in 1991, revised in 1993 due to security concerns, and another minor revision in 1996 and 2013
  - Cannot be used for encryption or key exchange
  - Uses an algorithm that is designed to provide only the digital signature function
  - Digital signatures are used to detect unauthorized modifications to data and to authenticate the identity of the signatory.
- Elliptic curve cryptography (ECC)
  - Equal security for smaller bit size than RSA
  - Seen in standards such as IEEE P1363
  - Based on a mathematical construct known as the elliptic curve
  - ECC is widely used in cryptocurrencies, such as Bitcoin, due to its lightweight.

Prime Numbers

- Factors are whole numbers that can be divided evenly into another number.
- Example
  - $1, 3, 5$ and 15 are factors of 15
- Prime number p
  - p is an integer
  - $p \geq 2$
  - The only divisors of p are 1 and p.
  - i.,e, are numbers with exactly 2 factors.
- Composite number n
  - n is an integer
  - $n > 1$
  - The divisors of n are 1, n and at least one other number.
  - i.e., have more than 2 factors.
- Example
  - $2, 5, 11, 19$ are primes and $4, 6, 9$ are composite numbers.
  - Composite numbers that are a product of two prime numbers.

GCD

- The greatest common divisor *(GCD)* of two positive integers *a* and *b*, denoted *gcd(a,b)*, is the largest positive integer that divides both *a* and *b*.
  - *gcd(12, 20) = 4*
  - *gcd(14, 36) = 2*
- Two integers *a* and *b* are said to be relatively prime[1] or coprime if *gcd(a,b) = 1*
  - *12* and *7*

[1] *In this case, a does have a multiplicative inverse modulo b, which means there exists an integer x such that:* $a \cdot x \equiv 1 \pmod{b}$

Modular Arithmetic

- "Wrap around" arithmetic
  - Numbers "wrap around" upon reaching a certain value called the modulus.
  - Example: *12-hour* clock
- Modulo operator for a positive integer n
  - $a \bmod n$ denotes the remainder when a is divided by n.
  - $r \equiv a \bmod n$, that is, $a = r + qn$, where q *is quotient*
  - $5 \equiv 32 \bmod 9$, that is, $32 = 5 + 3 \times 9$
  - $\equiv$ congruence relation (equivalence relation)
- Associativity, Commutativity, and Distributivity hold in Modular Arithmetic
- Inverses also exist in modular arithmetic
  - Additive Inverse: $a + (-a) \bmod n \equiv 0$
  - Multiplicative Inverse: $a * a^{-1} \bmod n \equiv 1$

Euler's Totient Function $\phi(N)$

- Counts the positive integers up to a given integer N that are relatively prime to N
  - Relatively prime means with no factors in common with N
- Example: $\phi(10) = ?$
  - 4 because $\{1,3,7,9\}$ are relatively prime to 10
- Example: $\phi(p) = ?$, where p is a prime
  - $p-1$ because all lower numbers are relatively prime

RSA Public-Key Encryption

- By Rivest, Shamir & Adleman of MIT in 1977[1]
- It stands as one of the most commonly used asymmetric cryptosystems today, although elliptic curve cryptography (ECC) is becoming increasingly popular.
- Unlike the symmetric systems, RSA is not based on substitution and transposition.
- Instead, it relies on exponentiation of integers modulo a prime.
  - This cryptographic method is built on arithmetic involving very large integers, often hundreds or even thousands of bits long.
- RSA is mainly used for two applications
  - Transport of (i.e., symmetric) keys
  - Digital signatures

[1] A Method for Obtaining Digital Signatures and Public-Key Cryptosystems

RSA Key Generation

- Let p and q be two large prime numbers. Let $N = pq$ be the modulus.
  - p and q very large, chosen at random
  - Important to discard p and q once done
- Choose e relatively prime to $\phi(N) = \phi(p)\phi(q) = (p-1)(q-1)$.
  - ϕ is Euler's totient function: counts the positive integers up to a given integer N that are relatively prime to N
  - i.e., select the public exponent $e \in [1, \phi(N)-1]$ such that $gcd(e, \phi(N)) = 1$
- Find $d \in [1, \phi(N)-1]$ s.t. $e.d \equiv 1 \mod (p-1)(q-1)$.
  - e and d are multiplicative inverses $\phi(N)$
- *Public key* is (e, N)
- *Private key* is (d, N)
- Remark: $gcd(e, \phi(N)) = 1$ ensures that e has an inverse and, thus, that there is always a private key d

RSA Encryption and Decryption

- Message M is treated as a number.
  - Represent the message as an integer between 0 and $N-1$, where N is the modulus.
  - To ensure that the message falls within this range, it may be necessary to break a long message into a series of blocks and represent each block as such an integer.
  - Any standard representation can be used for this purpose.
    - The purpose here is not to encrypt the message but only to get it into the numeric form necessary for encryption
- To encrypt message M compute $C = M^e \mod N$
- To decrypt C compute $M = C^d \mod N$

RSA Keys

- Recall that e and N are public.
- If attacker can factor N, she can use e to easily find d since $ed \equiv 1 \bmod (p-1)(q-1)$.
- Factoring the modulus breaks RSA.
- It is not known whether factoring is the only way to break RSA.

Does RSA Really Work?

- Given $C \equiv M^e \bmod N$, show that $C^d \bmod N \equiv M^{ed} \equiv M \bmod N$
- We'll need Euler's Theorem:
  - If x is relatively prime to n then $x^{\phi(n)} \equiv 1 \bmod n$.
- Facts:
  - $ed \equiv 1 \bmod (p-1)(q-1)$
  - By definition of "mod", $ed = k(p-1)(q-1) + 1$ for some integer k
  - $\phi(N) = (p-1)(q-1)$
- Then $ed - 1 = k(p-1)(q-1) = k\phi(N)$.
- So, $M^{ed} = M^{(ed-1)+1} = M \cdot M^{ed-1} = M \cdot M^{k\phi(N)} = M \cdot (M^{\phi(N)})^k$

Does RSA Really Work?

- To proof $M \equiv M \cdot (M^{\phi(N)})^k \mod N$, we'll need Euler's Theorem:
  - If $gcd(M,N) = 1$ (M is relatively prime to N) then $M^{\phi(N)} \equiv 1 \mod N$.
  - A minor generalization: $1 \equiv 1^k \equiv (M^{\phi(N)})^k \mod N$
- For the proof we distinguish two cases:
  1. $gcd(M,N) = 1$

     $$M \cdot (M^{\phi(N)})^k \equiv M \cdot 1^k \mod N \equiv M \mod N$$

  2. $gcd(M,N) = gcd(M,p \cdot q) \neq 1$
     - Since p and q are primes, M must have one of them as a factor: $M=r \cdot p$ or $M=s \cdot q$, where r,s are integers s.t. $r<p$ and $s<p$
     - Without loss of generality, assume $M=r \cdot p \Rightarrow gcd(M,q)=1 \Rightarrow$ Euler's Theorem holds $1 \equiv 1^k \equiv (M^{\phi(q)})^k \mod q$
     - $(M^{\phi(N)})^k \equiv (M^{(q-1)(p-1)})^k \equiv ((M^{\phi(q)})^k)^{(p-1)} \equiv 1^{(p-1)} = 1 \mod q$
     - This is equivalent to: $(M^{\phi(N)})^k = 1 + u \cdot q$, where u is some integer.
     - $M \cdot (M^{\phi(N)})^k = M + M \cdot u \cdot q = M + (r \cdot p) \cdot u \cdot q = M + r \cdot u \cdot N$
     - $\Rightarrow d_{k_{pr}} = M \cdot (M^{\phi(N)})^k \equiv M \mod N$

Simple RSA Example

- Select "large" primes $p = 11,\ q = 3$
- Then $N = pq = 33$ and $(p-1)(q-1) = 20$
- Choose $e = 3$ (relatively prime to 20)
- Find d such that $ed \equiv 1 \bmod 20$
  - We find that $d = 7$ works
- *Public key:* $(N,e) = (33,3)$. *Private key:* $d = 7$

Simple RSA Example

- *Public key:* $(N, e) = (33, 3)$
- *Private key:* $d = 7$
- Suppose message to encrypt is $M = 8$
- Ciphertext C is computed as
  - $C \equiv M^e \bmod N \equiv 8^3 = 512 \equiv 17 \bmod 33$
- Decrypt C to recover the message M by
  - $M = C^d \bmod N = 17^7 = 410{,}338{,}673 = 12{,}434{,}505 \cdot 33 + 8 \equiv 8 \bmod 33$

Implementation aspects

- The RSA cryptosystem uses only one arithmetic operation (modular exponentiation) which makes it conceptually a simple asymmetric scheme
- Even though conceptually simple, due to the use of very long numbers, RSA is orders of magnitude slower than symmetric schemes, e.g., DES, AES
- When implementing RSA (esp. on a constrained device such as smartcards or cell phones) close attention has to be paid to the correct choice of arithmetic algorithms

More Efficient RSA

- Modular exponentiation of large numbers with large exponents is an expensive operation.
- To make it more manageable, several tricks are used in practice.
- Modular exponentiation example
  - $5^{20} = 95367431640625 \equiv 25 \mod 35$
  - The naïve approach is to multiply 5 by itself 20 times and then reduce the result $\mod 35$
- When you works with "real" RSA numbers, they get too big to store and would take forever to compute!

More Efficient RSA: Square-and-Multiply

- Compute: $x^a \mod n$
- Instead of performing a multiplications (where a is the exponent), the square-and-multiply algorithm only requires around $\log_2(a)$ multiplications.
- This is a substantial improvement, especially for large exponents.
- Due to fewer multiplications, the algorithm is faster to execute, making it more suitable for devices with limited processing power and memory.

More Efficient RSA: Square-and-Multiply

- Compute: $x^a \mod n$
- Convert the exponent to binary:
  - Represent the exponent a as a binary number (e.g., 10 in decimal becomes 1010 in binary).
- Initialize a variable *result* to 1.
- Process bits from left to right:
  - Iterate through the binary representation of a from left to right.
    - If the bit is 1:
      - Square the current *result* and take the modulo by n.
      - Multiply the squared *result* by x and again take the modulo by n.
    - If the bit is 0: Simply square the current *result* and take the modulo by n.
- After iterating through all bits, the final *result* will be the desired value of $x^a \mod n$.

Efficient RSA: square-and-multiply algorithm

Compute: $x^a \mod n$

```
Require: x in {0, ..., n − 1} and a = (a_{l−1},...,a_0)_2
    r ← 1
    for i from l − 1 downto 0 do
      r ← r^2 mod n
      if a_i = 1 then
          r ← r × x mod n
      end if
    end for
return r
```

Efficient RSA: square-and-multiply algorithm

```
Require: x in {0, ..., n − 1} and a = (a_{l−1},...,a_0)_2
    r ← 1
    for i from l − 1 downto 0 do
     r ← r^2 mod n
      if a_i = 1 then
          r ← r × x mod n
      end if
    end for
return r
```

- Computes 5^{20} without modulo reduction
  - Binary representation of exponent: $20 = (10100)_2$
  - $(1, 10, 101, 1010, 10100) = (1, 2, 5, 10, 20)$
  - 5^{20}: Note that $20 = 2 \times 10$, $10 = 2 \times 5$, $5 = 2 \times 2 + 1$, $2 = 1 \times 2$
  - $5^1 \equiv 5 \bmod 35$
  - $5^2 = (5^1)^2 = 5^2 \equiv 25 \bmod 35$
  - $5^5 = (5^2)^2 \cdot 5^1 = 25^2 \cdot 5 = 3125 \equiv 10 \bmod 35$
  - $5^{10} = (5^5)^2 = 10^2 = 100 \equiv 30 \bmod 35$
  - $5^{20} = (5^{10})^2 = 30^2 = 900 \equiv 25 \bmod 35$
- No huge numbers and it's efficient!

Speed-Up Techniques

- Modular exponentiation is computationally intensive
- Even with the square-and-multiply algorithm, RSA can be slow on resource-constrained devices.
- Some important tricks: (not covered here)
  - Short public exponent e (e is often set to $65537=2^{16}+1$ for faster encryption)
    - $65537=2^{16}+1$ is a good compromise between security and performance.
    - Large enough to avoid the attacks to which small exponents make RSA vulnerable, and
    - Can be computed extremely quickly on binary computers, which often support shift and increment instructions
  - Chinese Remainder Theorem (CRT) (for more efficient decryption)
    - With CRT, the exponentiation $C^d \ mod \ N$ is split into two operations: $C^d \ mod \ p$ and $C^d \ mod \ q$, where p and q are the prime factors of N.
    - These smaller modular exponentiations are faster, and the results are combined using CRT at the end.
    - Since p and q are about half the size of N, this speeds up the exponentiation.
    - This technique is primarily applied for decryption.
  - Exponentiation with pre-computation. Pre-computation involves storing pre-computed values for certain operations used during exponentiation.

Speed-Up Techniques

- Montgomery Modular Multiplication
  - This specific multiplication algorithm avoids expensive division operations often present in standard modular multiplication.
  - It reduces the number of modular reductions needed, leading to faster computations.
- Hardware Acceleration
  - Some devices provide dedicated hardware accelerators for cryptographic operations, including modular exponentiation.
  - Utilizing these accelerators can offer significant performance gains compared to software implementations.
- Choosing Appropriate Libraries
  - Utilize optimized libraries that implement efficient algorithms and leverage hardware acceleration when available.
  - Carefully evaluate different libraries based on your specific needs and constraints.

RSA in the Real World

- Things are never easy. You cannot use the RSA we talked about for real applications.
  - Deterministic encryption
  - Malleability

- *Public key cryptosystems like RSA are often taught by describing how they work at a high level. Even an algorithm textbook written by one of the inventors of RSA focuses more on exponentiation and how the decryption operation inverts ciphertext than on the underlying security details. However, all public key algorithms have various requirements that are not part of the basic primitive but are vital for security.*

(Plain) RSA is deterministic

- Public key: (e, N). Private key: d. Encryption: $E(M) = M^e \bmod N$
- Eve finds matching ciphertexts, she knows the plaintexts match too.
  - Remember ECB?
- Eve can check for potential decryptions.
  - Eve (of course) knows Alice's public key.
  - She sees C. She suspects $D_d(C) = M$.
  - She can check! She computes $E_e(M)$ and compares to C

(Plain) RSA is malleable

- Malleability:
  - Property allows an attacker to manipulate a ciphertext without knowing the private key.
  - In the case of RSA, this means manipulating the encrypted message (ciphertext) to obtain a related message after decryption.
  - Specifically, multiplying two RSA ciphertexts created with the same public key results in the encryption of the product of their plaintexts.
- Public key: (e,N). Private key: d. Encryption: $E(M) = M^e \bmod N$
- Suppose Eve intercepts two ciphertexts: $E(M_1)$ and $E(M_2)$.
  - $E(M_1) \cdot E(M_2) = M_1^e \cdot M_2^e \bmod N = (M_1 \cdot M_2)^e \bmod N = E(M_1 \cdot M_2)$
- Eve doesn't know M_1 or M_2 but she managed to calculate a function of the plaintext
  - (after decryption, Alice will get the product of M_1 and M_2)

(Plain) RSA is malleable: Example

- Suppose Eve knows that Alice is reporting Eve's salary to Bob's payroll service firm.
  - $C = (Eve\_salary)^e \mod N$
- Eve intercepts C and sends $2^e \times C \mod N$ instead.
- Bob decrypts:
  - $(2^e \times C)^d = (2 \times Eve\_salary)^{ed} \mod N = 2 \times Eve\_salary$

Padding

- To avoid these problems, practical RSA implementations typically embed some form of structured, randomized padding into the value M before encrypting it
- Mechanism: Introduces random data before, after, or both sides of the actual message (plaintext).
  - This padding ensures that M does not fall into the range of insecure plaintexts.
  - Once a given message padded, will encrypt to one of a large number of different possible ciphertexts.

- *Padding in RSA: Prevents malleability attacks by obscuring the relationship between ciphertexts and their corresponding plaintexts. Makes RSA semantically secure, meaning an attacker cannot gain any information about the plaintext without the private key, even if they can manipulate the ciphertext.*

Padding

- We can fix a few issues by introducing padding.
  1. Malleability
     - If we have a strict format for messages, i.e. that the first or last bytes contain a specific value, simply multiplying both message and ciphertext will decrease the probability of creating a valid (in terms of padding) message.
  2. Semantical Security
     - Add randomness such that RSA is not deterministic anymore.

Padding: RSA-OAEP

- Optimal Asymmetric Encryption Padding (OAEP)
- Roughly, to encrypt M
  - Choose random r
  - Encode M as $M' = [X = M \oplus H_1(r)\ ,\ Y = r \oplus H_2(X)\ ]$ where H_1 and H_2 are cryptographic hash functions,
- Usage in RSA
  - The encoded message can then be encrypted with RSA: Encrypt $(M')^e \mod n$
  - The deterministic property of RSA is now avoided by using the OAEP encoding.
- To decrypt $M' = [X, Y]$
  - Recover the random string as $r = Y \oplus H_2(X)$
  - Recover the message as $M = X \oplus H_1(r)$
- Unless both X and Y are fully recovered, cannot obtain r, without r, cannot obtain any information of M.

Factoring assumption

- The factoring problem is to find all prime divisors of a composite number N.
- The factoring assumption is that there is no probabilistic polynomial-time algorithm for solving the factoring problem, even for the special case of an integer N that is the product of just two distinct primes.
- The security of RSA is based on the factoring assumption. No feasible factoring algorithm is known, but there is no proof that such an algorithm does not exist.

How big is big enough?

- The security of RSA depends on N, p, q being sufficiently large.
- What is sufficiently large?
  - Nowadays, N is typically chosen to be $2048\ bits$ or $3072\ bits$ long[1].
  - The primes p and q whose product is N are generally chosen be roughly the same length, so each will be about half as long as N.

  - *If one of the primes, say p, were significantly smaller than q, an attacker might have an easier time attempting to factor N by testing smaller primes as potential factors. By ensuring that p and q are similar in size, you increase the complexity of the factoring problem and make it more secure.*

[1] NIST Special Publication (SP) 800-57 Part 3, Rev. 1.

Key Lengths Comparison

Symmetric Key Size (bits)	RSA and Diffie-Hellman Key Size (bits)	Elliptic Curve Key Size (bits)
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	521

Symmetric vs. Asymmetric

- By now you should know that you either get performance or key distribution.
  - Symmetric: fast but need to deal with keys.
  - Asymmetric: slow (orders of magnitude) but resolved key distribution

Security of RSA

- Brute force
  - Involves trying all possible private keys until the correct one is found.
  - However, the large key sizes used in RSA (typically 2048 bits or more) make this computationally infeasible with current technology.
- Mathematical attacks
  - There are several approaches, all equivalent in effort to factoring the product of two primes
- Timing attacks
  - These exploit differences in execution time based on the secret key.
  - While relevant in some early implementations, modern optimizations and careful algorithm design have mitigated the effectiveness of these attacks.

Security of RSA

- Chosen ciphertext attacks
  - These attacks involve sending deliberately crafted ciphertexts to the RSA system and analyzing the responses to gather information about the private key.
  - Various forms of padding, such as OAEP, are crucial to prevent attackers from exploiting such information leaks.
- Fault Injection Attacks
  - These involve intentionally inducing hardware or software faults during the decryption process to extract or manipulate decrypted data.
  - Secure hardware implementations and careful error handling can counter these attempts.

Overview of Mathematical Attacks: Factoring Problem

- Mathematical approaches to attacking RSA
  - Factor N into two prime factors, hence find $\phi(N) = (p-1) \times (q-1)$ and then find $d \equiv e^{-1} \bmod \phi(N)$
  - Determine $\phi(N)$ directly without first determine p and q, then find d
  - Determine d directly without first determine $\phi(N)$
- For a large N with large prime factors, factoring is a hard problem, but not as hard as it used to be.

- *Factoring large numbers is computationally difficult, but advancements in algorithms and computing power necessitate using sufficiently large key sizes to stay ahead of potential threats.*

Progress in Factorization

- For a large N with large prime factors, factoring is a hard problem, but not as hard as it used to be.
- To track the state of factorization technology and encourage research in cryptography, RSA Laboratories issued RSA challenges with varying key sizes.
- These challenges involved attempting to factor large composite numbers with known bit lengths. Initially, challenges with key sizes of 100, 110, and 120 digits (*approximately 332 bits, 365 bits, and 398 bits, respectively*) were posed.
- Over time, advancements in factorization algorithms have been made.

- *The number of bits in a decimal number can be estimated by using the fact that each digit in decimal represents about 3.32 bits as there are 10 possible values for each decimal digit, and* $2^{3.32} \approx 10$.

Progress in Factorization

Number of Decimal Digits	Number of Bits	Date Achieved
100	332	April 1991
110	365	April 1992
120	398	June 1993
129	428	April 1994
130	431	April 1996
140	465	February 1999
155	512	August 1999
160	530	April 2003
174	576	December 2003
200	663	May 2005
193	640	November 2005
232	768	December 2009

- Please watch RSA-129 - Numberphile
- https://en.wikipedia.org/wiki/RSA_Factoring_Challenge

Progress in Factorization

- Advances in computer hardware, including the availability of high-performance computing clusters and distributed computing projects, have significantly improved the ability to perform large-scale factorization.
- As a result, the factorization of progressively larger RSA challenge numbers became achievable.
- The progress in factorization and increased computational power has led to the recommendation of longer RSA key sizes for security.
    - For example, what was once considered secure with 1024-bit keys is now considered weak, and longer key sizes (e.g., 2048 bits or more) are recommended for robust security.

Side-Channel Attacks

- Type of cryptographic attack that does not directly target the underlying mathematical properties of a cryptographic algorithm or the encryption key itself.
- Instead, exploit unintended information leakage from various physical implementation aspects of a cryptographic system.
- These physical implementations can include factors such as the time it takes for an operation to execute, the power consumption of a device during cryptographic operations, electromagnetic emissions, and even acoustic signals.

Side-Channel Attacks: Common Types

- Timing Attacks
  - *involve measuring the time it takes for a cryptographic operation to complete. By observing variations in execution time, an attacker may deduce sensitive information about the cryptographic key.*
- Power Analysis Attacks
  - *involve monitoring the power consumption of a device during cryptographic operations. Variations in power usage can reveal information about the operations being performed and potentially expose the encryption key.*
- Electromagnetic (EM) Analysis Attacks
  - *focus on detecting electromagnetic emissions produced by a device during cryptographic operations. The electromagnetic radiation can carry information about the data being processed, revealing key material.*
- Acoustic Attacks
  - *involve using sensitive microphones to capture sound produced by a device during cryptographic operations. The sound may contain information about the device's internal operations, potentially leading to key extraction.*
- Cache Timing Attacks
  - *exploit variations in the time it takes to access data in a CPU's cache memory. These variations can disclose information about memory access patterns and, in turn, cryptographic keys.*

Timing Attack Countermeasures

- Constant exponentiation time
  - Ensure that all exponentiations take the same amount of time before returning a result
  - This is a simple fix but does degrade performance
- Random delay
  - Better performance could be achieved by adding a random delay to the exponentiation algorithm to confuse the timing attack
  - If defenders do not add enough noise, attackers could still succeed by collecting additional measurements to compensate for the random delays
- Blinding
  - Multiply the ciphertext by a random number before performing exponentiation
  - This process prevents the attacker from knowing what ciphertext bits are being processed inside the computer and therefore prevents the bit-by-bit analysis essential to the timing attack

Quantum Computing Implications:

- *The potential development of quantum computers poses a long-term threat to RSA and other public-key encryption schemes. Quantum computers could, in theory, efficiently factor large numbers using algorithms like Shor's algorithm, which would render RSA insecure.*

Key exchange problem

- The key exchange problem is for Alice and Bob to agree on a common random key k.
- One way for this to happen is for Alice to choose k at random and then communicate it to Bob over a secure channel.
  - but same issue as with symmetric crypto.
- A better way is to use public key crypto.

The Discrete Log Problem

- A fundamental problem in number theory and plays a crucial role in modern cryptography.
- The mathematical basis for several algorithms
- For every prime number p, there exists a primitive root[1] (or "generator") α such that

$$\alpha^1,\ \alpha^2,\ \alpha^3,\ \alpha^4,\ldots,\alpha^{p-2},\ \alpha^{p-1} \text{ (all taken } mod\ p\text{)}$$

  are all distinct values (so this is a permutation of $1,2,3,\ldots,p-1$)
- Example: 3 is a primitive root of 17, with powers:

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$3^i \bmod 17$	3	9	10	13	5	15	11	16	14	8	7	4	12	2	6	1

- α and p are global public parameters
- Discrete Logarithm problem
  - Given integers n,α and prime number p, compute i such that $n \equiv \alpha^i \ mod\ p$
  - Solutions known for small p
  - Solutions computationally infeasible as p grows large

[1] *Primitive root: When it raised to any exponent i, the solution distributes uniformly/is equally likely to be an interger between* 0 *and* p.

Diffie-Hellman Key Exchange (DHKE)

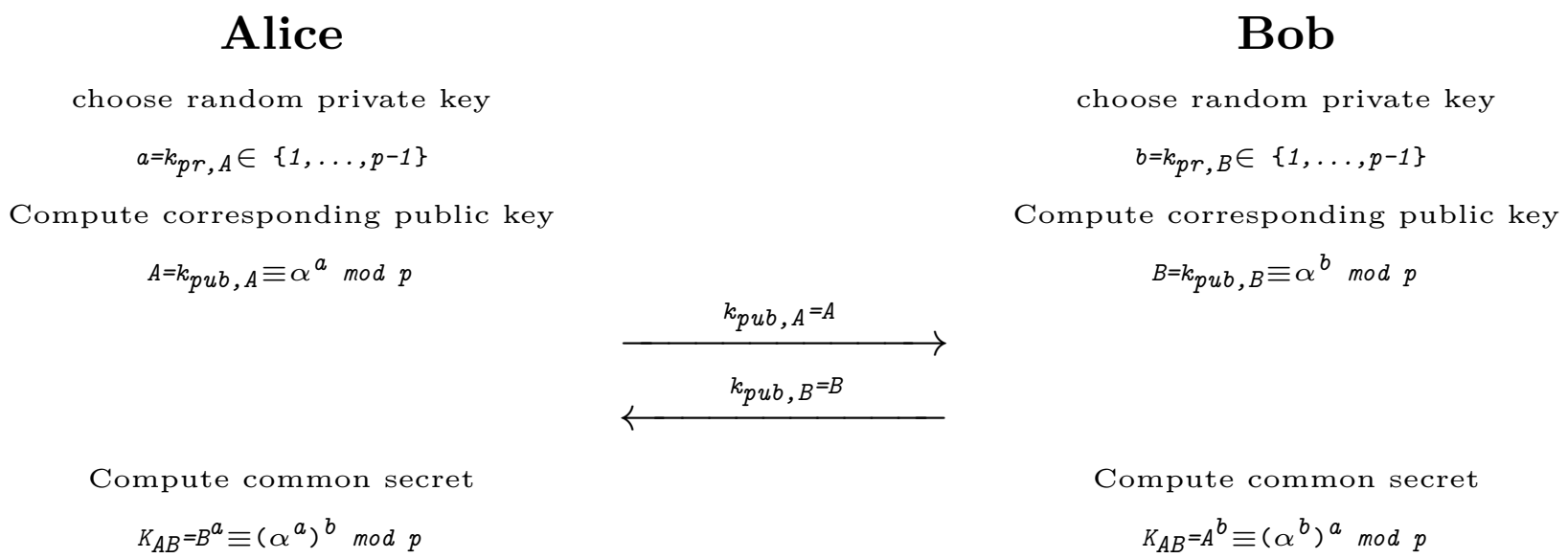
- First published public-key algorithm
- By Diffie and Hellman in 1976 along with the exposition of public key concepts
- Used in a number of commercial products
- Practical method to exchange a secret key securely that can then be used for subsequent encryption of messages
- Security relies on difficulty of computing discrete logarithms
- A "key exchange" algorithm:
  - Used to establish a shared symmetric key.
  - Called a symmetric key exchange protocol
  - Not for encrypting or signing.
- The point is to agree on a key that two parties can use for a symmetric encryption, in such a way that an eavesdropper cannot obtain the key

Diffie-Hellman Key Exchange (DHKE)

- Let p be prime, α be a generator, $\alpha < p$
- Alice selects her private value a
- Bob selects his private value b
- Alice sends $\alpha^a \mod p$ to Bob
- Bob sends $\alpha^b \mod p$ to Alice
- Both compute shared secret, $\alpha^{ab} \mod p$
- Shared secret can be used as symmetric key

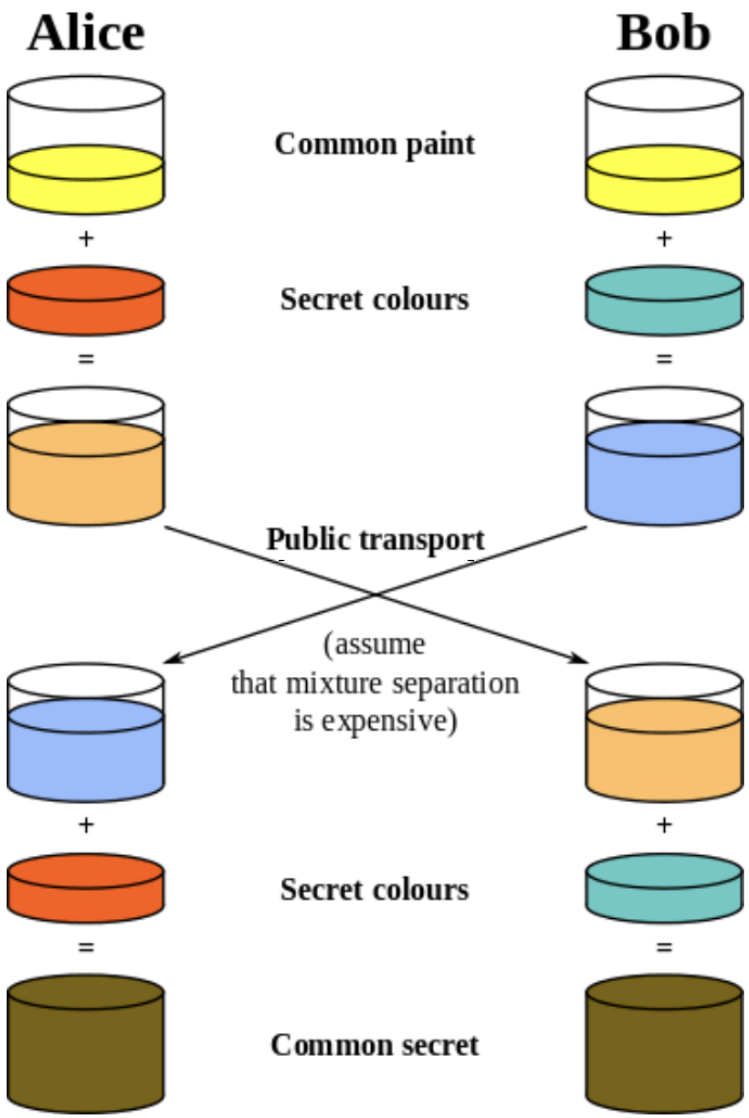
Diffie-Hellman Key Exchange (DHKE)

- Public: p (prime) and $\alpha \mod p$
- Private: Alice's exponent a, Bob's exponent b



- The key $K = K_{AB} = \alpha^{ab} \mod p$ can now be used to establish a secure communication between Alice and Bob
  - e.g., by using K_{AB} as key for a symmetric algorithm like AES or 3DES

Diffie-Hellman Key Exchange (DHKE)



Diffie-Hellman Key Exchange: Example

- Have
  - Prime number $p = 29$
  - Primitive root $\alpha = 2$

Alice		**Bob**
choose $a = k_{pr,A} = 5$		choose $b = k_{pr,B} = 12$
$A = k_{pub,A} \equiv 2^5 \mod 29 \equiv 3 \mod 29$		$B = k_{pub,B} \equiv 2^{12} \mod 29 \equiv 7 \mod 29$
	$A = 3$ →	
	← $B = 7$	
Compute common secret		Compute common secret
$K_{AB} = B^a \equiv 7^5 \mod 29 = 16 \mod 29$		$K_{AB} = A^b \equiv 3^{12} \mod 29 = 16 \mod 29$

Diffie-Hellman: What can Eve do?

- Suppose Bob and Alice use Diffie-Hellman to determine key $K = \alpha^{ab} \mod p$
- Eve can see $\alpha^a \mod p$ and $\alpha^b \mod p$
  - But... $\alpha^a \cdot \alpha^b \mod p = \alpha^{a+b} \mod p \neq \alpha^{ab} \mod p$
- If Eve can find a or b, she gets K

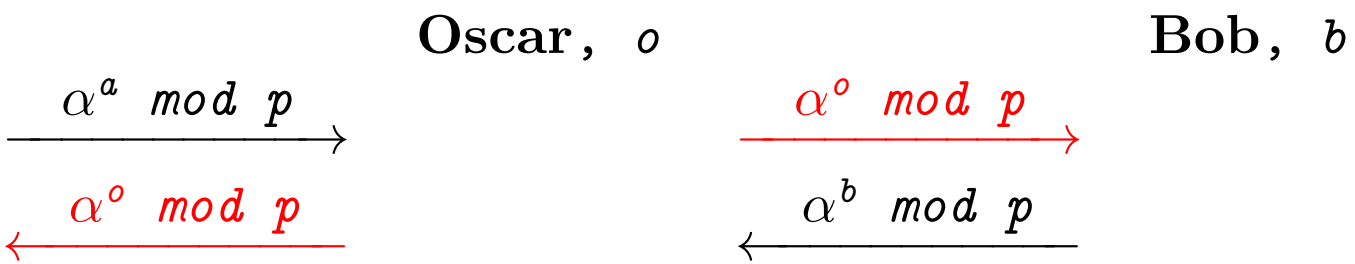
Security of DH key exchange

- The security of DHKE protocol relies on Eve's presumed inability to compute K from A and B and the public information p and α.
- This problem is sometimes referred to as the Diffie-Hellman problem, and, similar to the discrete logarithm problem (DLP), it is believed to be computationally intractable.
  - The challenge in Diffie-Hellman problem is to compute $\alpha^{ab} \mod p$ given $\alpha^a \mod p$ and $\alpha^b \mod p$ with given α and p are known.
  - DHKE is believed to be secure for large enough p
  - However, it is important to note that while the Diffie-Hellman problem is related to the discrete logarithm problem, it is not known to be as hard as DLP.

  - It remains an open question whether one could solve the Diffie-Hellman problem without first solving the discrete logarithm problem.
  - If the only way of solving DHP requires the DLP, one would way that "the DHP is equivalent to the DLP". However, this is not proven (yet)

Diffie-Hellman: Man-in-the-Middle Attack

- Subject to man-in-the-middle attack.
- Oscar sits between Alice and Bob, and replaces all messages on either direction.
  - Neither Alice and Bob will be able to detect it!



- Oscar shares secret α^{ao} with Alice.
- Oscar shares secret α^{bo} with Bob.
- Alice and Bob don't know Oscar exists (Man-in-the-Middle)

Diffie-Hellman: Man-in-the-Middle

- How to prevent Man-in-the-Middle attack?
  - Encrypt DH exchange with symmetric key.
  - Encrypt DH exchange with public key.
  - Sign DH values with private key.
  - Other?
- At this point, DH may look pointless
  - but it's not (more on this later).

Summary

- Public-key algorithms have capabilities that symmetric ciphers don't have, in particular digital signature and key establishment functions.
- Public-key algorithms are computationally intensive (a nice way of saying that they are slow), and hence are poorly suited for bulk data encryption.
- Only three families of public-key schemes are widely used. This is considerably fewer than in the case of symmetric algorithms.
- The Diffie-Hellman protocol is a widely used method for key exchange.
- The discrete logarithm problem is one of the most important one-way functions in modern asymmetric cryptography. Many public-key algorithms are based on it.
- For a better long-term security, at least, a prime of length *2048 bits* should be chosen.

Reading

- *Computer Security: Principles and Practice*
  - Chapter 2 and Chapter 21
- Optional: *Understanding Cryptography: A Textbook for Students and Practitioners*
  - Chapters 6,7, and 8