CS458: Introduction to Information Security

Notes 5: Public-Key Cryptography

Yousef M. Elmehdwi

Department of Computer Science

Illinois Institute of Technology

yelmehdwi@iit.edu

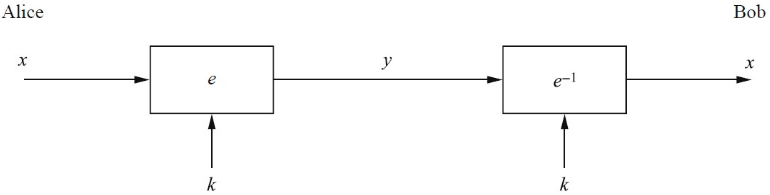
September 25, 2018

Slides: Modified from Christof Paar and Jan Pelzl & Ewa Syta

Outline

- Principles of Asymmetric Cryptography
- Practical Aspects of Public-Key Cryptography
- 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

Symmetric Cryptography: Analogy

- Safe with a strong lock, only Alice and Bob have a copy of the key
  - Alice encrypts $\rightarrow$ locks message in the safe with her key
  - Bob decrypts $\rightarrow$ uses his copy of the key to open the safe

Symmetric Cryptography: Shortcomings

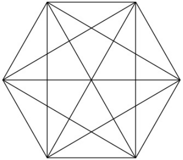
- Symmetric algorithms, e.g., AES or 3DES, are very secure, fast & widespread **but**:

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)

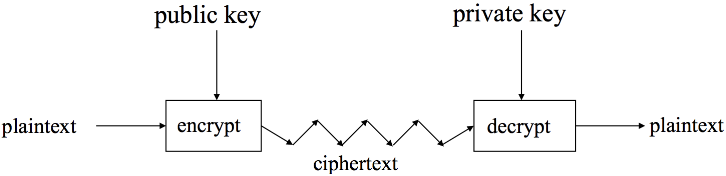


Symmetric Cryptography: Shortcomings

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"

Public Key Crypto



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

Uses of Public Key Crypto

- Encryption
  - Suppose we encrypt m with Bob's public key.
  - Bob's private key can decrypt c to recover m.
  - Q: Why public key for encryption?
- Digital Signatures
  - Bob signs by "encrypting" with his private key.
  - Anyone can use Bob's public key to verify the signature.
  - Like a handwritten signature, but way better...
  - Q: Why private key for digital signatures?

Security of the keys

- Two keys, public and private
  - Given that one key is public, the other one cannot be (easily) computable.
- Based on "trapdoor one-way function"
  - "One-way" means easy to compute in one direction, but hard to compute in other direction (reverse)
    - Easy to calculate $f(x)$ from x
    - Hard to invert: to calculate x from $f(x)$
    - Example:
    - Given p and q, product $N = pq$ easy to compute, but hard to find p and q from N.
  - A trapdoor one-way function has one more property, that with certain knowledge it is easy to invert, to calculate x from $f(x)$
    - i.e., "Trapdoor" is used when creating key pairs. If you have it, you can reverse the process

Security Mechanisms of Public-Key Cryptography

- Here are main mechanisms that can be realized with asymmetric cryptography:
  - Symmetric Key Distribution (e.g., Diffie-Hellman key exchange, RSA) without a pre-shared secret (key)
  - Nonrepudiation and Digital Signatures (e.g., RSA, DSA or ECDSA) to provide message integrity
    - i.e., Integrity/authentication: 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 (1000 times slower than symmetric Algorithms!)

Basic Protocol for Public-Key Encryption

Alice		**Bob**
		$Key_B = (K_{pub_B}, K_{pr_B})$
	$\xleftarrow{\quad K_{pub_B} \quad}$	
x		
$y = e_{K_{pub_B}}(x)$		
	$\xrightarrow{\quad y \quad}$	
		$x = d_{K_{pr_B}}(y)$

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

Basic Key Transport Protocol

- In practice: Hybrid systems, incorporating asymmetric and symmetric algorithms
  - Examples: SSL/TLS protocol 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
2. Encryption of data is done using (fast) symmetric ciphers, e.g., block ciphers or stream ciphers

Basic Key Transport Protocol

- Example: Hybrid protocol with AES as the symmetric cipher

Alice		**Bob**
		$Key_B = (K_{pub_B}, K_{pr_B})$
	$\xleftarrow{K_{pub_B}}$	
Choose random symmetric key K $y_1 = e_{K_{pub_B}}(K)$		
	$\xrightarrow{y_1}$	
		$K = d_{K_{pr_B}}(y_1)$
message x $y_2 = AES_K(x)$		
	$\xrightarrow{y_2}$	
		$x = AES^{-1}{}_K(y_2)$

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 m, find x such that $a^x = y \bmod m$
    - (Exponentiation a^x: easy)
  - Elliptic Curves (EC) Schemes
    - Generalization of discrete logarithm
    - e.g., Elliptic Curve Diffie-Hellman key exchange (ECDH) and the Elliptic Curve Digital Signature Algorithm (ECDSA)

Key Lengths and Security Levels

- An algorithm is said to have a "security level of n bit" if the best known attack requires 2^n steps
  - Symmetric algorithms with security level of n have key of length n bit.
  - The relationship between cryptographic strength and security is not as straightforward in the asymmetric case

Symmetric	*ECC*	*RSA, DL*	*Remark*
64 Bit	128 Bit	≈ 700 Bit	Only short term security (a few hours or days)
80 Bit	160 Bit	≈ 1024 Bit	Medium security (except attacks from big governmental institutions etc.)
128 Bit	256 Bit	≈ 3072 Bit	Long term security (without quantum computers)

  - The exact complexity of RSA (factoring) and DL is difficult to estimate
  - The existence of quantum computers would probably be the end for ECC, RSA & DL (at least 2-3 decades away, and some people doubt that QC will ever exist)

Requirements

- Computationally easy
  - for a party to generate a key pair
  - to encrypt a message using a public key
  - for the receiver to decrypt a message using the private key
- 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
- Either of the two related keys can be used for encryption with the other used for decryption

General Facts about Public Key Systems

- Public Key Systems are much slower than Symmetric Key Systems
  - 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

Announcement

- Exam 1: Take Home Exam
  - Start: Monday October 15 at 10:00AM
  - End: Wednesday October 17 by 10:00AM SHARP
- Online students (Campus): You need to take the exam with the live class
- Online students (not in campus): You need to contact Charles Scott <scott@iit.edu> to schedule remote site proctors

NCIX DATA BREACH, September 2018

- *NCIX DATA BREACH*
    - Millions of Canadian and American consumers are now at risk thanks to a series of shady backroom deals that have resulted in records detailing 15 years of business being sold.

Outline

- The RSA Cryptosystem

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 or coprime if $gcd(a,b) = 1$
  - 12 and 7

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)

The RSA Cryptosystem

- RSA was independently invented by: Clifford Cocks (GCHQ)
  - 1973 (not published; classified)
  - Government Communications Headquarters
  - British Intelligence Agency
- Martin Hellman and Whitfield Diffie published their landmark public key paper in 1976[1]
- Ronald **R**ivest, Adi **S**hamir and Leonard **A**dleman proposed the asymmetric RSA cryptosystem in 1977[2]

[1] New Directions in Cryptography

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

Overview of RSA

- Probably the most commonly used asymmetric cryptosystem today, although elliptic curve cryptography (ECC) becomes increasingly popular
- Unlike the symmetric systems, RSA is based not on substitution and transposition.
- RSA is based on arithmetic involving very large integers numbers that are hundreds or even thousands of bits long
- RSA is mainly used for two applications
    - Transport of (i.e., symmetric) keys
    - Digital signatures

RSA Key Generation

- Let p and q be two large prime numbers. Let $N = pq$ be the modulus.
  - p and q chosen at random
  - Primality tests
  - 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 $e \in \{1, 2, \ldots, \Phi(N)-1\}$ such that $gcd(e, \Phi(N)) = 1$
- Find d s.t. $ed \equiv 1 \bmod (p-1)(q-1)$.
  - Modular multiplicative inverse
  - Extended Euclidean algorithm
- *Public key* is (e, N).
- *Private key* is (d)
- 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.
  - Must be less than N.
- To encrypt message M compute $C = M^e \bmod N$
- To decrypt C compute $M = C^d \bmod N$

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

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
  - $5^{20} = 95367431640625 \equiv 25 \bmod 35$
  - The naïve approach is to multiply 5 by itself 20 times and then reduce the result $\bmod 35$
- 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

- A better way: square-and-multiply algorithm
- **Basic principle**: Determine binary representation of the exponent, then scan exponent bits from left to right and square/multiply operand accordingly
  - The idea is to build up the exponent one bit at a time.
  - At each step we double/square the current exponent and if the binary expansion of the number has a *1* in the corresponding position, we add to the exponent.
  - Take *mod* whenever possible.

More Efficient RSA: Square-and-Multiply: Example

- Computes 5^{20} without modulo reduction
  - Binary representation of exponent: $20 = (10100)_2$
  - $(1, 10, 101, 1010, 10100) = (1, 2, 5, 10, 20)$
  - Note that $2 = 1 \cdot 2,\ 5 = 2 \cdot 2 + 1,\ 10 = 2 \cdot 5,\ 20 = 2 \cdot 10$
  - $5^1 \equiv 5 \bmod 35$
  - $5^2 = (5^1)^2 = 5^2 \equiv 25 \bmod 35$
  - $5^5 = (5^2)^2 \cdot 5^1 = 25^2 \cdot 5 = 3125 \equiv 10 \bmod 35$
  - $5^{10} = (5^5)^2 = 10^2 = 100 \equiv 30 \bmod 35$
  - $5^{20} = (5^{10})^2 = 30^2 = 900 \equiv 25 \bmod 35$
- 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 quite slow on constrained devices such as smart cards
- Some important tricks: (not covered here)
  - Short public exponent e
  - Chinese Remainder Theorem (CRT)
  - Exponentiation with pre-computation

RSA in the Real World

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

(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 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: the property that a ciphertext can be transformed into another ciphertext which decrypts to a related plaintext without knowing the private key
  - it allows an attacker to modify the contents of a message
- Public key: (e,N). Private key: d. Encryption: $E(M) = M^e \bmod N$
- Eve can fiddle with two ciphertexts encrypted under the same key:
  - $E(M_1) \cdot E(M_2) = M_1{}^e \cdot M_2{}^e \bmod N = (M_1 \cdot M_2)^e \bmod N = E(M_1 \cdot M_2)$
- 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)

Padding

- Optimal Asymmetric Encryption Padding (OAEP) is a padding scheme often used together with RSA encryption.
- OAEP satisfies the following two goals:
    - Add an element of randomness.
    - Prevent partial decryption of ciphertexts (or other information leakage) by ensuring that an adversary cannot recover any portion of the plaintext.

Factoring assumption

- The factoring problem is to find a prime divisor 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?
  - Hard to say.
  - N is typically chosen to be at least $1024\ bits$ long, or for better security, $2048\ bits$ long.
  - 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.

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

- Diffie-Hellman Key Exchange

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.

Groups

- A group is a set of elements G together with an operation $\circ$ which combines two elements of G. A group has the following properties:
- $(G, \circ)$ forms a group because:
  - Closed: $\forall a, b \in G,\ a \circ b \in G$.
  - Associative: $\forall a, b, c \in G,\ (a \circ b) \circ c = a \circ (b \circ c)$
  - Identity (neutral) element: $\forall a \in G,\ 1 \circ a = a \circ 1 = a$
  - Inverse element: $\forall a \in G,\ \exists b \in G\ s.t.\ \ a \circ b = b \circ a = 1$
  - Commutative: $\forall a, b \in G,\ a \circ b = b \circ a$ (abelian group)
- In cryptography we use both multiplicative groups and additive groups

Groups

- $(\mathbb{Z},+)$ is a group:
  - i.e., the set of integers $\mathbb{Z} = \{\ldots,-2,-1,0,1,2,\ldots\}$ together with the usual addition forms an **abelian group**, where $e = 0$ is the identity element and $-a$ is the inverse of an element $a \in \mathbb{Z}$
- We need groups with a finite number of elements.

Modular Arithmetic: Groups

- Recall that so far all operations were done *mod n*
- We then have a group: $\mathbb{Z}_n$ = $\{0, 1, \ldots, n-1\}$
  - $\mathbb{Z}_n$ is just a convenient notation for numbers between *0* and *n-1*.
- Problem: Inverses only exist for elements *a* such that *gcd(a,n)=1*
- We can define another group $\mathbb{Z}_n^*$ = $\{a \in \mathbb{Z}_n \mid gcd(a,n)=1\}$
  - All numbers in $\mathbb{Z}_n$ that are relatively prime to *n*.
  - $\mathbb{Z}_n^*$ forms an abelian group under multiplication modulo *n*. The identity element is *1*
  - i.e., defined as the set of positive integers smaller than *n* which are relatively prime to *n*
- Note:
  - $\mathbb{Z}_p^*$, *p* is prime, forms a multiplicative group. $\mathbb{Z}_p^*$ = $\{1, 2, 3, \ldots, p-1\}$

Finite Group

- A group $(G,\circ)$ is a finite if it has a finite number of elements. We denote the **cardinality** or **order** of the group G by $|G|$
- Example:
  - $(\mathbb{Z}_n,+)$: the cardinality of $\mathbb{Z}_n$ is $|\mathbb{Z}_n|=n$ since $\mathbb{Z}_n=\{0,1,2,\ldots,n-1\}$
  - $\mathbb{Z}_p^*$: the cardinality of $\mathbb{Z}_p^*$ equals **Euler's phi function** evaluated for p,
    - i.e., $|\mathbb{Z}_p^*| = \Phi(p)$
    - For instance, the group $|\mathbb{Z}_9^*|$ has a cardinality of $\Phi(9) = 3^2-3^1 = 6$.

Order of an element

- The order $ord(\alpha)$ of an element α of a group $(\mathbb{Z}_p^*, \circ)$: is the smallest positive integer k such that $\alpha^k = \underbrace{\alpha \circ \alpha \circ \alpha \circ \ldots \circ \alpha}_{k\ times} = 1$ where 1 is the identity element of G.
- Example: $\mathbb{Z}_{11}^*$={1,2,3,4,5,6,7,8,9,10}. Q: order of $\alpha = 3$?
  - $\alpha^1 = 3$
  - $\alpha^2 = 9$
  - $\alpha^3 = 27 \equiv 5$
  - $\alpha^4 = \alpha^3\ \alpha = 5\ .\ 3 \equiv 4 \bmod 11$
  - $\alpha^5 = \alpha^4\ \alpha = 4\ .\ 3 \equiv 1 \bmod 11$
  - $\alpha^6 = \alpha^5\ \alpha = 1\ .\ 3 \equiv 3 \bmod 11$
  - $\alpha^7 = \alpha^6\ \alpha = 3\ .\ 3 \equiv 9 \bmod 11$
  - $\alpha^8 = \alpha^7\ \alpha = 9\ .\ 3 \equiv 5 \bmod 11$
  - $\alpha^9 = \alpha^8\ \alpha = 5\ .\ 3 \equiv 4 \bmod 11$
  - $\alpha^{10} = \alpha^9\ \alpha = 4\ .\ 3 \equiv 1 \bmod 11$
  - $\alpha^{11} = \alpha^{10}\ \alpha = 1\ .\ 3 \equiv 3 \bmod 11$
  - ...
- The powers of α run through the sequence {3,9,5,4,1} indefinitely.

Cyclic Groups

- Keep computing powers of α until we obtain the identity element 1
- Cyclic Group
  - A group G which contains an element α with maximum order $ord(\alpha) = |G|$ is said to be cyclic. Elements with maximum order are called primitive elements/roots or generators.
- An element α of a group G with maximum order is called a **generator** since every element a of G can be written as a power $\alpha^i = a$ of this element for some i
- i.e., α generates the entire group
- Cyclic groups are the basis of discrete logarithm cryptosystems. '

Cyclic Groups

- Example: $\mathbb{Z}_{11}^{*}$={1,2,3,4,5,6,7,8,9,10}
- Q: Check whether $\alpha = 2$ happens to be a primitive element of $\mathbb{Z}_{11}^{*}$?
  - $\alpha^{1}=2$
  - $\alpha^{2}=4$
  - $\alpha^{3}=8$
  - $\alpha^{4}=5$
  - $\alpha^{5}=10$
  - $\alpha^{6}=9$
  - $\alpha^{7}=7$
  - $\alpha^{8}=3$
  - $\alpha^{9}=6$
  - $\alpha^{10}=1$
  - $\alpha^{11}=2$
- $ord(2) = 10 = |\mathbb{Z}_{11}^{*}|$
- $\alpha = 2$ is a generator of $\mathbb{Z}_{11}^{*}$

Primitive root

- We say α is a primitive root of n if α generates all of $\mathbb{Z}_n^*$.
- Not every integer n has primitive roots but every prime p does.
- For every prime p, $(\mathbb{Z}_n^*, \cdot)$ is an abelian finite cycle group
  - i.e., the multiplicative group of every prime field is cyclic
- Let G be a finite cyclic group. Then it holds that
  1. The number of primitive elements of G is $\Phi(|G|)$.
  2. If $|G|$ is prime, then all elements $a \neq 1 \in G$ are primitive.

Primitive root example

- Let p=19, so $\Phi(p)$=18 and $\Phi(|G|)$=$\Phi(18)$=$\Phi(2) \cdot \Phi(9) = 6$.
- Consider $\alpha = 2$ and $\alpha = 5$.
- The subgroups $\mathbf{S}_\alpha$ of $\mathbb{Z}_p$ generated by each α is given by the table:

k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2^k	2	4	8	16	13	7	14	9	18	17	15	11	3	6	12	5	10	1
5^k	5	6	11	17	9	7	16	4	1	5	6	11	17	9	7	16	4	1

- We see that 2 is a primitive root since $\mathbf{S}_2 = \mathbb{Z}_p^*$ but 5 is not

Logarithms mod p

- Let $y = b^x$ over the reals. The ordinary $base\text{-}b$ logarithm is the inverse of exponentiation, so $x = log_b(y)$
- The discrete logarithm is defined similarly, but now arithmetic is performed in $\mathbb{Z}_p^*$ for a prime p.

$$y \equiv b^x \ mod\ p, \quad x = log_b(y)\ mod\ p$$

- Fact: If b is a primitive root of p, then $log_b(y)$ is defined for every $y \in \mathbb{Z}_p^*$.

Discrete Log Problem

- Given is the finite cyclic group $\mathbb{Z}_p^*$ of order $p-1$ and a primitive element α *in* $\mathbb{Z}_p^*$ and another element β *in* $\mathbb{Z}_p^*$. The Discrete Log Problem (DLP) is the problem of determining the integer $1 \leq x \leq p-1$ such that:

$$\alpha^x \equiv \beta \; mod \; p$$

  - Put another way, compute $log_\alpha(\beta)$
- No efficient algorithm is known for this problem and it is believed to be intractable.
  - **Brute-force**: compute $\alpha^x \; mod \; p$ for $x=1,2,...,p-1$.
  - Better algorithm exists, but still of exponential time

Diffie-Hellman Key Exchange (DHKE)

- The first public key cryptosystem proposed
  - Invented by Williamson (GCHQ) and, independently, by D and H (Stanford)
- First practical method for establishing a shared secret over an unsecured communication channel
- A "key exchange" algorithm:
  - Used to establish a shared symmetric key.
  - Called a symmetric key exchange protocol
  - Not for encrypting or signing.
  - Based on the discrete log problem
- 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 is a cornerstone of modern cryptography used for VPNs, HTTPS websites, email, and many other protocols.

Diffie-Hellman Key Exchange (DHKE)

- Let p be prime, α be a generator
- 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

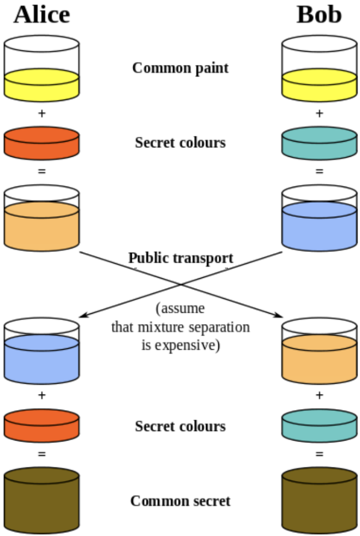
Diffie-Hellman Key Exchange (DHKE)

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

Alice		**Bob**
choose random private key		choose random private key
$a = k_{pr,A} \in \{2, \ldots, p-2\}$		$b = k_{pr,B} \in \{2, \ldots, p-2\}$
Compute corresponding public key		Compute corresponding public key
$A = k_{pub,A} \equiv \alpha^a \ mod \ p$		$B = k_{pub,B} \equiv \alpha^b \ mod \ p$
	$\xrightarrow{k_{pub,A} = A}$	
	$\xleftarrow{k_{pub,B} = B}$	
Compute common secret		Compute common secret
$K_{AB} = B^a \equiv (\alpha^a)^b \ mod \ p$		$K_{AB} = A^b \equiv (\alpha^b)^a \ mod \ p$

- 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

- The Diffie-Hellman domain parameters are $p = 29$ and $\alpha = 2$. The protocol proceeds as follows:

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

- ElGamal Cryptosystem
  - Proposed by Taher Elgamal in 1985
  - Can be viewed as an extension of the DHKE protocol

A variant of DHKE

- Bob goes first followed (at some point) by Alice.

Alice	**Bob**
	choose random y
	$B \equiv \alpha^{b}$ *mod* p
	Send B to Alice
choose random a	
$A \equiv \alpha^{a}$ *mod* p	
Send A to Bob	
$k_{ab} = B^{a} \equiv \alpha^{ba}$ *mod* p	$k_{ab} = B^{b} \equiv \alpha^{ab}$ *mod* p

Comparison with first DHKE protocol

- The difference here is that Bob completes his action at the beginning and no longer has to communicate with Alice.
- Alice, at a later time, can complete her half of the protocol and send a to Bob, at which point Alice and Bob share a key.

Turning DHKE into a public key cryptosystem

- "Principle of ElGamal Encryption"
- Consider two parties, Alice and Bob.
- Alice wants to send an encrypted message x to Bob
- Alice and Bob first complete DHKE to derive a shared key k_M
- Alice uses this key as a multiplicative mask to encrypt x as $y \equiv x\; k_M \mod p$.

Principle of ElGamal Encryption

- Bob computes his private key d and public key β.
  - This key pair does not change, i.e., it can be used for encrypting many messages
- Alice computes her private key i and public key K_E (Ephemeral key).
  - Alice has to generate a new public-private key pair for the encryption of every message
  - K_E is ephemeral (existing only temporarily) key, hence the index E
- Joint key is denoted by k_M because it is used for masking the plaintext

Alice		**Bob**
		(a) choose $d = k_{pr,B} \in \{2, \ldots, p-2\}$ (b) compute $\beta = k_{pub,B} \equiv \alpha^d \bmod p$
	$\xleftarrow{\ \beta\ }$	
(c) choose $i = k_{pr,A} \in \{2, \ldots, p-2\}$ (d) compute $k_E = k_{pub,A} \equiv \alpha^i \bmod p$		
	$\xrightarrow{\ k_E\ }$	
(e) compute $k_M \equiv \beta^i \bmod p$ (g) encrypt message $x \in \mathbb{Z}_p^*$ $y \equiv x \cdot k_M \bmod p$		(f) compute $k_M \equiv k_E^d \bmod p$
	$\xrightarrow{\ y\ }$	
		(h) decrypt $x \equiv y \cdot k_M^{-1} \bmod p$

ElGamal Encryption Protocol

Alice		**Bob**
		choose large prime p choose primitive element $\alpha \in \mathbb{Z}_p^*$ or in a subgroup of $\mathbb{Z}_p^*$ choose $k_{pr} = d \in \{2, \ldots, p-2\}$ compute $k_{pub} = \beta = \alpha^d \bmod p$
	$\xleftarrow{k_{pub}=(p,\alpha,\beta)}$	
choose $i \in \{2, \ldots, p-2\}$ compute ephemeral key $k_E \equiv \alpha^i \bmod p$ compute masking key $k_M \equiv \beta^i \bmod p$ encrypt message $x \in \mathbb{Z}_p^*$ $y \equiv x \cdot k_M \bmod p$		
	$\xrightarrow{(k_E, y)}$	
		compute masking key $k_M \equiv k_E^d \bmod p$ decrypt $x \equiv y \cdot k_M^{-1} \bmod p$

ElGamal Encryption Protocol

- ElGamal is a probabilistic encryption scheme, i.e., encrypting two identical messages x_1 and x_2, where $x_1, x_2 \in \mathbb{Z}_p^*$ using the same public key results (with extremely high likelihood) in two different ciphertexts $y_1 \neq y_2$
- This is because i is chosen at random from $\{2, 3, \ldots, p-2\}$ for each encryption, and thus also the session key $k_M = \beta^i$ used for encryption is chosen at random for each encryption.

ElGamal Encryption Protocol: Example

Alice		**Bob**
message $x = 26$		generate $p = 29$ and $\alpha = 2$
		choose $k_{pr,B} = d = 12$
		compute $\beta = \alpha^d \equiv 7 \bmod 29$
	$\xleftarrow{k_{pub,B}=(p,\alpha,\beta)}$	
choose $i = 5$		
compute $k_E = \alpha^i \equiv 3 \bmod 29$		
compute $k_M = \beta^i \equiv 16 \bmod 29$		
encrypt $y = x \cdot k_M \equiv 10 \bmod 29$		
	$\xrightarrow{y,k_E}$	
		compute $k_M = k_E^d \equiv 16 \bmod 29$
		decrypt
		$x = y \cdot k_M^{-1} \equiv 10 \cdot 20 \equiv 26 \bmod 29$

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 this protocol relies on Eve's presumed inability to compute K from a and b and the public information p and α.
- This is sometime called the Diffie-Hellman problem and, like discrete log, is believed to be intractable.
  - 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
- Certainly the Diffie-Hellman problem is no harder that discrete log. However, it is not known to be as hard as discrete log.
  - It is unknown if this could be done without solving discrete logarithm first.
  - 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 (MiM)

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

Alice, a		**Eve**, e		**Bob**, b
	$\xrightarrow{\alpha^a \ mod\ p}$		$\xrightarrow{\alpha^e \ mod\ p}$	
	$\xleftarrow{\alpha^e \ mod\ p}$		$\xleftarrow{\alpha^b \ mod\ p}$	

- Eve shares secret α^{ae} with Alice.
- Eve shares secret α^{be} with Bob.
- Alice and Bob don't know Eve exists (MiM)

Diffie-Hellman: MiM

- How to prevent MiM 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).
- You **must** be aware of MiM attack on Diffie-Hellman

How crypto fails in practice[3]

- **Socat** is an all-purpose command-line network tool that can connect almost any type of network resource and supports virtually any network protocol. It makes use of DH.
- Socat was found to be using a hardcoded, $1024\text{-}bit$ non-prime Diffie-Hellman parameter.

[3] Dan Goodin, "Crypto flaw was so glaring it may be intentional eavesdropping backdoor", Arstechnica 2/2/2016

Lessons Learned

- 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 the Diffie-Hellman protocol in $\mathbb{Z}_p^*$ the prime p should be at least *1024 bits* long. This provides a security roughly equivalent to an 80 bit symmetric cipher.
- For a better long-term security, a prime of length *2048 bits* should be chosen.

Lessons Learned

- The ElGamal scheme is an extension of the DHKE where the derived session key is used as a multiplicative masked to encrypt a message.
- ElGamal is a probabilistic encryption scheme, i.e., encrypting two identical messages does not yield two identical ciphertexts.
- For the ElGamal encryption scheme over $\mathbb{Z}_p^*$, the prime p should be at least *1024* bits long, i.e., $p > 2^{1000}$.