Algoritmi di Crittografia

Corso di Laurea Magistrale in Informatica

A.A. 2018/2019

Algoritmi di Crittografia

Algoritmi di Crittografia

A (shared) key management problem

- Encryption and authentication protocols studied so far make use of keys shared by the parties that need to communicate
- For modern crypto applications (notably, *e-commerce*) the problem of securely sharing keys among parties is all but trivial
- Besides the security issues, a simple counting argument shows that there is also a serious *key management problem*
- In fact, if n people need to (secretly) communicate one with each other, the number of different keys required is $\frac{n(n-1)}{2}$, i.e., grows quadratically with the number of people
- For another example, an e-commerce site with hundred of millions customers would need to agree as many diferent shared keys
- In their 1976 paper, *New Directions in Cryptography*, Stanford's researchers Whitfield Diffie and Martin Hellman addressed this efficiency question and designed another scenario

Public key encryption

- If encryption and decryption used different keys, the encryption keys could be made public, i.e., obtainable by anyone
- In this scenario, if Alice wants to send confidential information to Bob, all she has to do is to get his public key P_B and encrypt the message with P_B
- Once the message has been encrypted, only Bob can made it clear using a key only available to him (i.e., his *secret* key S_B)
- With only the pair (P_B, S_B) it is then clear that Alice can send confidential messages to Bob but not the other way around

Requirements

- To implement a public key encryption scheme we need a *trapdoor* function f: it must be easy to compute $y = f(x)$ but, unless you know a piece of information k, the computation of $x = f^{-1}(y)$ is very hard
- Any subject who needs to receive confidential messages must possess a secret key, public key pair
- With public keys, there is still a key management problem: how can Alice get Bob's public key P_b (and rest assured that P_B is really Bob's)?
- The latter problem cannot be solved with Mathematics only, we need a *Key Management Infrastructure*

Diffie and Hellman’s contribution

- Diffie and Hellman were unable to devise a trapdoor function, and thus only gave a partial answer to the question of efficiency of key sharing
- They proposed a scheme for *sharing secrets* among two parties that communicate over an insecure channel
- This scheme is known as the *Diffie and Hellman key exchange protocol* (or, more simply, *DH-protocol*)
- The DH-protocol is today adopted in other secure Internet protocols, such as *TLS* and *SSH*
- Diffie and Hellman were awarded the Turing prize in 2015

Algoritmi di Crittografia

Cyclic groups

- Let $\mathcal{G} = (G, \cdot)$ be a finite multiplicative group and let $n = |G| + 1$
- For simplicity, in the following we will use G to refer both to the group and the underlying set
- The *order* of an element $x \in G$ is the minimum positive integer a such that $x^a = 1$, where (as customary) 1 is the identity element of G
- Note that, since G is finite, a is also finite and $a < n$
- G is *cyclic* is there exists an element g with order $n - 1$
- Such an element g is said a *generator* of the group; in fact, it is not difficult to prove that $G = \{g^t | t = 1, 2, \ldots, n-1\}$
- Group generators are also known as *primitive elements* of the group

Cyclic groups $\mathbf{Z}_p^*$, p prime

- If p is prime the set $\mathbf{Z}_p^*$ is a group under multiplication modulo p
- Known facts
  1. $\mathbf{Z}_p^*$ is cyclic
  2. The number of generators of $\mathbf{Z}_p^*$ is $\Omega\left(\frac{p-1}{\log p}\right)$
  3. There is an efficient test to recognize generators “provided that the factorization of $p-1$ is known”
- If g is a generator and $x \in \mathbf{Z}_p^*$, then the smallest integer d such that $x = g^d \bmod p$ is the *discrete logarithm* of x to the base g
- We also write $d = \log_g x$
- The best algorithm to compute the discrete logarithm (which is essentially the *Number Field Sieve*) runs in time $\Theta\left(e^{f(p)(\log p)^{1/3}(\log \log p)^{2/3}}\right)$, where $f(p) \rightarrow \sqrt[3]{\frac{64}{9}}$ and $\log$ here stands for the natural logarithm

Subgroups

- A subgroup S of a group G is a subset of G that is itself a group (with the same operation)
- If G is finite and S is a subgroup of G, then the following holds true: the order of S is a divisor of the order of G
- Also, for any divisor d of the group order, there is exactly one subgroup of order d
- As an example, the subgroups of $\mathbf{Z}_7^* = \{1, 2, 3, 4, 5, 6\}$ can only have order 1, 2, 3, or 6:
- Any subgroup can be expressed as the powers of a group element (the *generator* of the subgroup)
- For $\mathbf{Z}_7^*$ we have
  - the subgroup of order 1: $\{1\}$
  - the subgroup of order 2: $\{1, 6\}$, generated by 6
  - the subgroup of order 3: $\{1, 2, 4\}$, generated by 2 (or 4)
  - the whole group $\mathbf{Z}_7^*$, with generators 3 and 5

The D-H Protocol

- Alice and Bob “publicly” agree on the group $G = \mathbf{Z}_p^*$ and a corresponding group generator g
- Alice chooses a number a uniformly at random in $\mathbf{Z}_p^*$, computes $x = g^a$, and sends x to Bob
- Bob receives x from Alice, chooses a number b uniformly at random in $\mathbf{Z}_p^*$ and computes $y = g^b$ as well as $z_B = x^b \mod p$; finally, he sends y to Alice
- Alice receives y from Bob and computes $z_A = y^a \mod p$
- Optionally, Alice and Bob apply some (common) transformations to z_A and z_B to derive a shared key to adopt in symmetric protocols

Properties

- It is easy to prove that $z_A = z_B$. In fact:

$$\begin{aligned} z_A &= y^a \bmod p \\ &= (g^b \bmod p)^a \bmod p \\ &= (g^b)^a \bmod p \\ &= (g^a)^b \bmod p \\ &= (g^a \bmod p)^b \bmod p \\ &= x^b \bmod p \\ &= z_B \end{aligned}$$

- Eve's problem (also known as the *Diffie-Hellman problem*) is to compute z ($z = z_A = z_B$) given $x = g^a$ (we ignore the mod reduction when understood) and $y = g^b$
- Solving the discrete logarithm problem is clearly sufficient to solve the D-H problem
- The converse is also believed to be true

The man-in-the-middle attack

- The most vicious assault to D-H is the well-known *meet-in-the-middle* attack
- That's simple. Eve might intercept Alice's $x = g^a$ message and send to Bob a different one, namely $x_1 = g^{e_1}$, where $e_1 \in \mathbf{Z}_p^*$ is any number of her choice
- Similarly, Eve might intercept Bob's $y = g^b$ message and send to Alice $x_2 = g^{e_2}$
- Neither Alice nor Bob can detect they're talking to Eve
- Now, when Alice sends an encrypted message to Bob, Eve simply decrypts it using the key she has in common with her and then forwards to Bob the same message encrypted with the key she has in common with him

A public key encryption scenario

- Instead of directly interacting one with each other, Alice and Bob might choose random values a and b and publish $x = g^a \bmod p$ and $x = g^a \bmod p$, respectively, in an appropriate digital directory
- If Alice wants to communicate with Bob, she must get y from the directory, and similarly Bob gets x
- In this way Alice uses the same random value x in each D-H protocol execution (not only with Bob)
- It can be proved that this fact does not reduce security (hint: “random times constant” is as random as “random times random”)

A public key encryption scheme

Key generation Choose d uniformly at random from $\mathbf{Z}_p^*$; compute $e = g^d$; publish $k_p = e$ and keep $k_s = d$ secret

Encryption (1) Get the public key k_p of the intended message receiver; choose r uniformly at random from $\mathbf{Z}_p^*$ and compute the values $x = g^r$ and $z = k_p^r (= g^{dr})$

Encryption (2) Using a pre-specified algorithm (e.g., `sha256`), use z to derive a symmetric key k and encrypt the message m using a pre-specified (block) algorithm, i.e., compute $c = E(k, m)$; then send the pair (x, c) to the receiver

Decryption Use the secret key k_s, compute $z = x^{k_s} (= g^{dr})$; from z derive the symmetric key k and then decrypt $m = D(k, c)$

Choosing a good prime p

- Question: is any large prime suitable for D-H protocols?
- First of all, if g is given, it may happen that its order is not large (i.e., g might not be a generator of the whole group)
- What happens in this case?
- The secret information g^{ab} is an element of the subgroup $G_g = \{1, g, g^2, \ldots, g^q\}$, with q being g's order
- But then, if q is (relatively) small, Eve might try a brute force approach to devise the key

Small subgroups

- Even if g is a generator, Eve might replace, e.g., $y = g^b$ (the message Bob sends to Alice) with a different element $y = w$ with small order
- When Alice computes $z = y^a$, she's in fact computing w^a, which is another element of the small order subgroup
- Again, in this case, brute force might become an option for Eve
- To face this problem we should avoid prime numbers p such that $p - 1$ has small divisors
- Besides the small subgroup problem, a strong reason to avoid small divisors of $p - 1$ is that the discrete logarithm problem may become easy
- In this respect, Pohlig and Hellman have discovered an algorithm for computing the discrete logarithm which runs in $O(\log^2 p)$ time if $p - 1$ has only small prime factors.

The so-called *safe primes*

- A safe prime avoids small subgroups "by construction"
- Let q be prime; if also $p = 2q + 1$ is prime, then it is a *safe prime*
- If p is a safe prime, then the prime factorization of $p - 1$ is simply $p - 1 = 2q$
- It follows that the proper subgroups of $\mathbf{Z}_p^*$ have order q and 2, call them Q and T, respectively, with $T = \{1, -1 \bmod p\} = \{1, p - 1\}$
- The generator of T is $p - 1$, which can be easily avoided (and easily detected if Eve uses it in a "substitution")
- Picking any other element at random can give a generator of the full group or a generator of Q, and it turns out that it can be easily checked which of the two is the case

Squares and non-squares

- Let $p = 11$ and thus $p = 2 \cdot 5 + 1$, i.e., $q = 5$
- If we square all the elements of $\mathbf{Z}_{11}^*$ we get set $\{1, 3, 4, 5, 6\}$, i.e., half the element of the group
- This is true in general: half the elements of $\mathbf{Z}_p^*$ are squares and half are non-squares
- Since the squares form clearly a group, it cannot be other but Q
- Also, each square is a generator for Q
- For instance, in $\mathbf{Z}_{11}^*$ we have,

$$\begin{aligned} Q &= \{3, 3^2 = 9, 3^3 = 5, 3^4 = 4, 3^5 = 1\} \\ &= \{4, 4^2 = 5, 4^3 = 9, 4^4 = 3, 4^5 = 1\} \\ &= \{5, 5^2 = 3, 5^3 = 4, 5^4 = 9, 5^5 = 1\} \\ &= \{9, 9^2 = 4, 9^3 = 3, 9^4 = 5, 9^5 = 1\} \end{aligned}$$

- Conversely, any non-square (possibly $p - 1$ excluded) must be a generator of the whole group

Using safe prime

- In order not to spill any information abou the secret numbers, it is advisable to select a square as the g value
- In fact, if g is a non-square, then g^a is odd iff a is odd
- In other words, the parity of g^a would reveal the last bit of a
- Recall also that testing if $x \in \mathbf{Z}_p^*$ is a square can be done efficiently

Using safe prime (cont.d)

- The protocol to devise p and q is thus the following
- Select large primes q until $p = 2q + 1$ is also a prime
- Choose $r \in \mathbf{Z}_p^*$ and computes $s = r^2 \mod p$
- If $s = 1$ or $s = p - 1$, discard and choose a different value for r
- Otherwise p is a suitable prime for use in D-H protocol, together with $g = s$

Generating the *p* and *g* parameters with openssl

- The `openssl` suite has commands for manipulating D-H parameters
- Safe primes can be generated using the following command:

```
openssl dhparam -outform PEM -out dhcert.pem 2048
```

- The above automatically uses 2 as the generator
- The certificate can be viewed using the following command

```
openssl dhparam -inform PEM -in dhcert.pem
                -check -text
```

Shared secrets and symmetric keys

- A serious error in the application of D-H protocols is to use the shared secret (that is, a number) as a cryptographic key
- The reason is that while the secret is a random element from a set of numbers, its single bits may not be 0 or 1 with equal probability
- To focus the problem, consider the simple case of $\mathbf{Z}_{11}^*$

1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	0	0	1	0
prob{b=1}	0.2	0.4	0.5	0.5

Hashing is the solution

- Cryptographic keys must be derived from the secret numbers by applying a cryptographic hash function (or some appropriate *key derivation function*)
- For instance, by using MD5 we get the following results, when applied to the 10 elements of $\mathbf{Z}_{11}^*$

mean	4.89
std	1.52

while, if using SHA256, we obtain

mean	5.07
std	1.55

Choosing a suitable magnitude for p

- How large must p be?
- As of today (i.e., using the best known discrete logarithm algorithm), to achieve 128 bit security we should use primes of roughly 7000 bits (actually a little less)
- A. K. Lenstra and E. R. Verheul made a thorough study of how large the primes should be to warrant security within a given year

year	size
2022	2048
2038	3072
2050	4096

The ElGamal encryption scheme

- This is a public key (and signature) scheme based on the D-H protocol for key exchange
- As in the "public version" of the D-H protocol, Alice publishes a description of an appropriate cyclic group G, i.e., generator g and prime modulus p
- Alice also publishes her public key $A = g^a$, where a is her private key, a number she has chosen uniformly at random in $\{1, 2, \ldots, p-1\}$
- When Bob wants to send an encrypted message to Alice, he obviously must first obtain the data published by Alice
- Bob then choses b uniformly at random in $\{1, 2, \ldots, p-1\}$ and then computes $B = g^b$ as well as the the secret value $S = A^b = g^{ab}$, that will be shared with Alice

The ElGamal encryption scheme (cont.d)

- To encrypt the message m, Bob simply computes the product $C = m \cdot S$ and sends the pair $\langle B, C\rangle$ to alice
- Upon receipt of $\langle B, C\rangle$, Alice computes $S = B^a = g^{ba}$ and then $m = C \cdot S^{-1}$
- Two observations are in order:
  1. Each time Bob must send a message Alice, he computes a “fresh” private key b and the corresponding public value $B = g^b$
  2. In real scenarios, the ElGamal system is used not to exchange the “true” message but rather the key of a symmetric cryptosystem
- To justify 1, observe that if an adversay knew m, then s/he could recover S and decrypt all future messages. b is referred to as an *ephemeral key*
- The reason for 2 has to do with the cost of computations in G and the size of symmetric keys vs typical size of real messages

Algoritmi di Crittografia

Rivest, Shamir, Adleman

- Turing prize in 2002 “For their ingenious contribution for making public-key cryptography useful in practice.”
- RSA public-key cryptosystem is based on a trapdoor one-way function
- A part from this, there are some similarities with D-H
- Given a message m (regarded as a number), RSA encryption amounts to computing $c = m^e \mod n$, for some publicly known parameters e and n
- Decryption, that is, going back from c to m, is believed to be computationally infeasible, for a suitably large and accurately constrained value of n
- However, if you know the factorization of n, which is chosen as the product of two large primes, then decryption is easy as well

Some usuful results from number theory

- If p is prime, then for any $a \in \{1, \ldots, p-1\}$ we have $a^{p-1} \bmod p = 1$ (Fermat's Little theorem)
- If $LCD(p, q) = 1$, i.e., p and q are relatively prime, then

$$\left.\begin{array}{r} x \bmod p = d \\ \\ x \bmod q = d \end{array}\right\} \Rightarrow x \bmod pq = d$$

Proof. $x \bmod p = d$ implies $x = ph + d$, for some $h \geq 0$. Similarly, $x = qk + d$, $k \geq 0$; but

$$x = ph + d = qk + d \qquad \Rightarrow \qquad ph = qk$$

That is, ph (and qk) is a multiple of both p and q, and since the latter are relatively prime, $lcm(p, q) = pq$ and $ph = tpq$, for some $t \geq 1$. But this in turn implies $x = tpq + d$, i.e. $x \bmod pq = d$, since $d < pq$.

The Euler “totient” function

- Given $n > 0$, the Euler's (or *totient*) function, denoted $\varphi(n)$, is the number of integers in the range $[1, n]$ tha are relatively prime with n
- For instance, $\varphi(10) = 4$, $\varphi(30) = 8$, while $\varphi(p) = p - 1$, for any prime p
- $\varphi(n)$ can be easily computed from the prime factorization of n
- In fact, it holds that

$$\varphi(n) = n \prod_{p|n} \left(1 - \frac{1}{p}\right)$$

- In particular, if n is the product of two primes, p and q, the general formula gives

$$\varphi(n) = n\left(1 - \frac{1}{p}\right)\left(1 - \frac{1}{q}\right) = (p-1)(q-1)$$

The *Chinese Remainder Theorem* (*CRT*)

- We consider a number n which is the product of two primes, p and q
- The CRT holds for an arbitrary composite number n, but (for simplicity) we limit n to the above form, since that is what is needed in RSA
- CRT helps to simplify RSA computations. In a nutshell it gives the theory for computing modulo p and modulo q rather than modulo n
- Let $\mathbf{Z}_n = \{0, \ldots, n-1\}$ with the usual $+$ and $\times$ operations modulo n
- For any $x \in \mathbf{Z}_n$, let x_p and x_q denote $x \bmod p$ and $x \bmod q$, respectively
- The first important result is that if $y_p = x_p$ and $y_q = x_q$, where $y \in \mathbf{Z}_n$, then necessarily $x = y$
  Proof. Let $d = x - y$. We clearly have $d_p = d_q = 0$. But then, from what we have just proved above, $d \bmod n = 0$, which implies $x \equiv y \pmod{n}$. But $x, y \in \mathbf{Z}_n$, hence this really means $x = y$.

Chinese remainder Theorem (cont.d)

- The above result tells that, if p and q are primes and $n = pq$, then the map $x \to (x_p, x_q)$ is a bijection in $\mathbf{Z}_n$
- The inverse map $(x_p, x_q) \to x$ can be computed efficiently by means of the Garner's formula
- Let $\hat{q} = q^{-1} \bmod p$, and let:

$$z = ((x_p - x_q)\hat{q} \bmod p)q + x_q$$

Then $z = x$.
Proof. First of all we note that $z' = (x_p - x_q)\hat{q} \bmod p$ is at most $p - 1$ and hence $z'q \geq (p-1)q = n - q$. Since $x_q < q$, it follows that $z = z' + x_q \in \mathbf{Z}_n$. Now, it is easy to see that, by the very definition, $z \bmod q = x_q$. All it remains to prove is that $z \bmod p = x_p$

Chinese remainder Theorem (cont.d)

- Recall that

$$z = ((x_p - x_q)\hat{q} \bmod p)q + x_q$$

Thus

$$\begin{aligned} z \bmod p &= (((x_p - x_q)\hat{q} \bmod p)q + x_q) \bmod p \\ &= (((((x_p - x_q)\hat{q} \bmod p)(q \bmod p)) \bmod p) + \\ &\qquad\qquad\qquad\qquad (x_q \bmod p)) \bmod p \\ &= ((x_p - x_q) \bmod p + (x_q \bmod p)) \bmod p \\ &= x_p \bmod p \\ &= x_p \end{aligned}$$

Some usuful results from number theory

- One last useful fact
- For composite n, $\mathbf{Z}_n$ is (only) a ring, not a field (i.e., $\mathbf{Z}_n \setminus \{0\}$ it is not a multiplicative group)
- This means that not all $x \in \mathbf{Z}_n$ have multiplicative inverse
- More specifically, x has a multiplicative inverse iff $gcd(x, n) = 1$, otherwise x is a divisor of zero
- The inverse of an element $x \in \mathbf{Z}_n$, $x \neq 0$, when it exists (which is always the case if n is prime), can be found using the *Extended Euclidean Algorithm*

The (textbook) RSA protocol

- **Input**: a positive integer b representing a bit length
- Choose two primes, p and q, uniformly at random in the range $[1, 2^b - 1]$
- Compute $n = pq$ and $\varphi(n) = (p-1)(q-1)$
- Choose an integer e in $\mathbf{Z}_n$ such $gcd(e, \varphi(n)) = 1$, and compute $d = e^{-1} \bmod \varphi(n)$
- Publish the pair (e, n) and keep d as the secret key
- **Encryption** Given m compute $c = m^e \bmod n$
- **Decryption** Given c, compute $m = c^d \bmod n$