Vietnam National University Ho Chi Minh City, University of Science
Department of Information Technology

Topic 6:

Asymmetric cipher

Assoc. Prof. Trần Minh Triết
PhD. Trương Toàn Thịnh

fit@hcmus

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

Introduction

- The problem that arises in conventional cryptosystems is the sharing of the key *k* between sender *A* and receiver *B*.
- In fact, the need to change the contents of the key *k* is necessary, so the exchange of information about the key *k* between *A* and *B* is necessary.
- To secure the key *k*, *A* and *B* must communicate with each other over a truly secure and secret communication channel.
- However, it is very difficult to guarantee the security of the communication channel, so the key *k* can still be detected by person *C*!

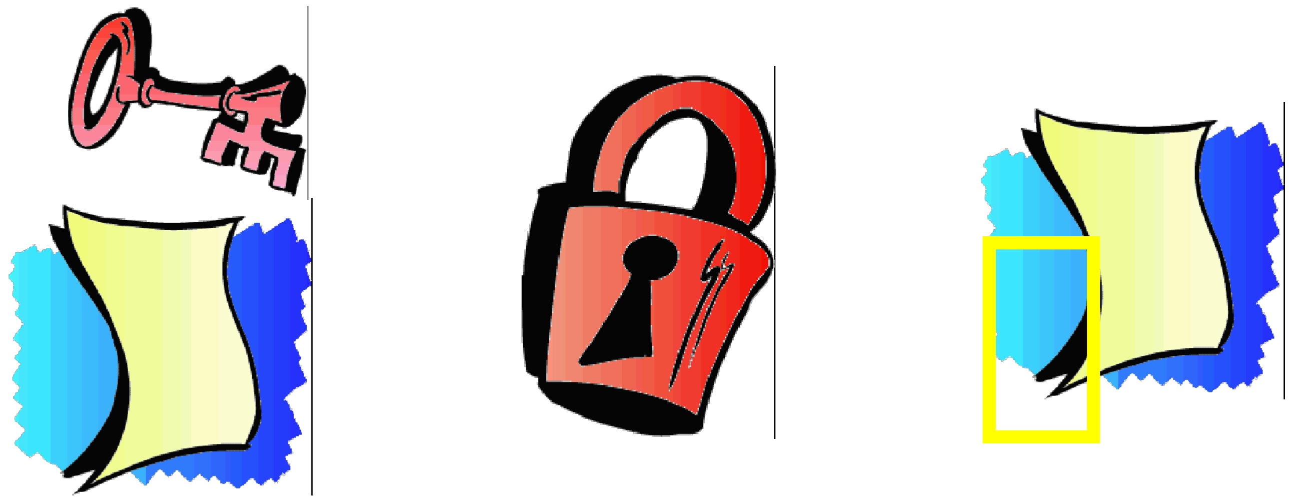
Introduction

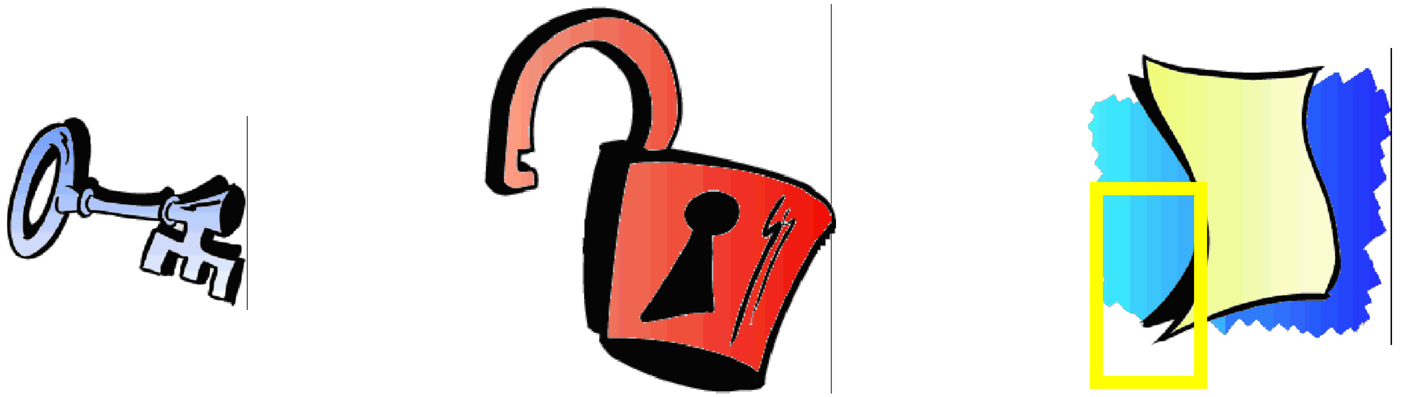
- The idea of a public key cryptosystem was introduced in 1976 by Martin Hellman, Ralph Merkle and Whitfield Diffie at Stanford University.
- Subsequently, the Diffie-Hellman method of Martin Hellman and Whitfield Diffie was published.
- In 1977, in the journal "The Scientific American", the authors Ronald Rivest, Adi Shamir and Leonard Adleman published the RSA method, the famous public key encryption method and widely used today in applications of encryption and protect information

Introduction

- A public-key system uses 2 types of keys in the same key pair:
  - Public key is widely available and used in information encryption
  - Private key is held by only one person and is used to decrypt information encrypted with the public key.
- These coding methods exploit *f* mappings for which the back mapping f^{-1} is very difficult compared to performing the *f*.
- Only when the private key is known can the reverse mapping f^{-1} be performed.

Public-key cryptosystem





RSA

- In 1978, R.L. Rivest, A. Shamir and L. Adleman proposed the RSA public key encryption system (also known as the "MIT system"). In this method, all calculations are performed on $\mathbb{Z}_{\mathrm{n}}$ where n is the product of two distinct odd primes p and q.
- So, we have $\varphi(n) = (p - 1) \times (q - 1)$
  - Let $P = C = Z_n$ and define: $K = \{((n, p, q, a, b): n = p \times q, p, q$ are distinct prime numbers, $a \times b \equiv 1 \pmod{\varphi(n)}\}$
  - For each $k = (n, p, q, a, b) \in K$, let:
    - $e_k(x) = x^b \bmod n$ and $d_k(y) = y^a \bmod n$, for $x, y \in Z_n$
  - n and b are published (public-key)
  - p, q, a are kept secret (private-key)

RSA

- Generate two prime numbers with large values p and q
- Compute $n = p \times q$ and $\varphi(n) = (p - 1) \times (q - 1)$
- Randomly choose an integer b $(1 < b < \varphi(n))$ such that $\gcd(b, \varphi(n)) = 1$
- Compute $a = b^{-1} \bmod \varphi(n)$ (using extended Euclide algorithm)
- n and b are published (public-key)
- p, q, a are kept secret (private-key)

Some attacks on RSA

- The security nature of the RSA method is based on the fact that the cost of invalidating encrypted information will be too great to be considered impossible.
- Since the key is public, RSA cracking attacks often rely on the public key to determine the corresponding private key. It is important to rely on n to calculate p, q of n, from which d can be calculated.

Using $\varphi(n)$

- Suppose the attacker knows the value $\varphi(n)$. Then the determination of the value p, q is taken to solve the following two equations: $n = p \times q$
- Let $q = n/p$, we have a quadratic equation: $\varphi(n) = (p - 1) \times (q - 1)$ and $p^2 - (n - \varphi(n) + 1) \times p + n = 0$
- p, q are the two solutions of this quadratic equation. However, the problem of detecting the value $\varphi(n)$ more difficult than determining two prime factors of n.

Using factorization algorithm *p*-1

- Algorithm (input n and B)

```
a = 2
for j = 2 to B do a = a^j mod n   //a^{2 × 3 × … × B} = a^{B!}
        d = gcd(a - 1, n)
        if 1 < d < n then d is a prime factor of n (success)
else cannot determine the prime factor of n (failure)
```

- Algorithm Pollard p - 1 (1974) is one of the efficient simple algorithms used to factorize large integers. The input parameter of the algorithm is an (odd) integer n that needs to be factored into prime factors and the limit value B.
- Assume $n = p \times q$ (p, q unknown) and B is a large enough integer, for each prime factor $k \leq B \wedge k \mid (p-1) \Rightarrow (p-1) \mid B!$

Using factorization algorithm *p*-1

- Analysis
  - At the end of the loop (step 2): $a \equiv 2^{B!} \pmod{n} \Rightarrow a \equiv 2^{B!} \pmod{p}$
  - Due to $p|n$, Fermat theorem allows: $2^{p-1} \equiv 1 \pmod{p}$
  - Due to $(p - 1) \mid B!$, at step 3 of algorithm: $a \equiv 1 \pmod{p}$
  - At step 4: $p \mid (a - 1)$ and $p \mid n$, if $d = gcd(a - 1, n)$ then $d = p$
- Example
  - Assume $n = 79523$ (= 281 × 283)
    - Using algorithm p–1 with $B = 8$, determine $a = 51705$ & $d = gcd(a – 1, n) = 281$
    - In this case, prime factorization is successful due to value $281 – 1 = 280 = 2^3 \times 5 \times 7 \Rightarrow 280 \mid B!$
  - Assume $n = 76693$ (= 271 × 283)
    - Using algorithm p–1 with $B = 8$, determine $a = 71515$ & $d = gcd(a – 1, n) = 1$
    - In this case, prime factorization failed due to the value $271 – 1 = 270 = 2 \times 3^3 \times 5 \Rightarrow 270 \nmid B!$

Using factorization algorithm *p*-1

- In algorithm $p - 1$, we have $B - 1$ exponentiations of modulo, each requires a maximum $2 \times \log_2 B$ modulo multiplication using squaring and multiplying algorithms
- Computing GCD using extended Euclidean algorithm with complexity $O((\log_2 n)^3)$.
- So, algorithmic complexity is $O(B \times \log_2 B(\log_2 n)^2 + (\log_2 n)^3)$
- The probability of choosing the value B is relatively small and satisfying the condition is very low.
- When increasing the value of B (such as $B \approx \sqrt{n}$) then the algorithm will succeed, but this algorithm will not be faster than the divisibility algorithm as shown above.

Using factorization algorithm p-1

- This algorithm is only effective when attacking the RSA method in the case that n has a prime factor p that (p - 1) has only very small prime divisors.
- We can easily construct an RSA public-key cryptosystem that is secure against the attack algorithm p - 1. The simplest way is to find a large prime p_1, which $p = 2p_1 + 1$ is also prime numbers, similarly find q_1 large prime and $q = 2q_1 + 1$ prime.

Key-break based on repeat attacks

- Simons and Norris: RSA systems can be vulnerable to repeated attacks. If the adversary knows public key-pair $\{n, e\}$ and cipher-text C, it can compute the sequence of the following:

$$\left.\begin{array}{l} \square\ C_1 = C^e \ (\mathrm{mod}\ n) \\ \square\ C_2 = C_1{}^e \ (\mathrm{mod}\ n) \\ \square\ \ldots \\ \square\ C_\mathrm{i} = C_{i\text{-}1}{}^e \ (\mathrm{mod}\ n) \end{array}\right\} \ \textit{next } C = (\textit{previous } C)^e$$

- If there is a C_j in $C_1, C_2, C_3, \ldots., C_i$ such that $C_j = C$, then we have $M = C_{j\text{-}1}$ because $C_\mathrm{j} = C_{j\text{-}1}{}^e$ (mod n) and $C = M^e$ (mod n)
- Example: if attacker knows $\{n, e, C\} = \{35, 17, 3\}$, it computes
  - $C_1 = C^e$ (mod n) = 317 (mod 35) = 33
  - $C_2 = C_1{}^e$ (mod n) = 3317 (mod 35) = 3
  - Because $C_2 = C \Rightarrow M = C_1 = 33$

Information hiding in the RSA system

- *RSA* is characterized by information that is not always concealed.
- Assume sender sends $e = 17$, $n = 35$. If it wants to send any data in {1, 6, 7, 8, 13, 14, 15, 20, 21, 22, 27, 28, 29, 34}, then result of encryption is the same as original data. So, $M = M^e \bmod n$.
- If $p = 109$, $q = 97$, $e = 865$ then RSA there is absolutely no hiding information, because $\forall M, M = M^{865} \bmod (109 \times 97)$
- For each value of n, there are at least 9 cases where the resulting encoding is the original source data.
- We have, $M = M^e \bmod n$, or: $M = M^e \bmod p$ & $M = M^e \bmod q$ (*)
  - For each e, each equality in (*) have at least 3 solutions $\in \{0, 1, -1\}$.
  - Number of unmasked messages (not changed after encryption): $m = [1 + gcd(e - 1, p - 1)][1 + gcd(e - 1), q - 1]$

Comments

- The key to being able to decrypt the information is to get the p and q values that make up the n value.
- Given these 2 values, we can calculate $\varphi(n) = (p - 1) \times (q - 1)$ & $d = e^{-1} \bmod \varphi(n)$ according to extended Euclidean algorithm.
- If the integer n can be factored, i.e., the values of p and q can be determined, then the security of the RSA method is no longer guaranteed.
- Thus, the security of the RSA method based on computers at the present time is not capable of solving the factorization of very large integers.
- In 1994, Peter Shor, a scientist at AT&T labs, came up with an algorithm that could efficiently parse very large integers on a quantum computer.

Prime number problem

- To ensure the security of the RSA system, the integer $n = p \times q$ must be large enough that the factorization of n cannot be easily performed. Currently, prime factorization algorithms can handle integers over 250 decimal digits, or 829 binary digits (*bits*).
- To be safe, the primes p and q need to be large enough, for example, over 125 digits. The problem here is to solve the problem: how to quickly and accurately check whether a positive integer n is prime or composite?
- By definition, a positive integer n is prime $\Leftrightarrow$ n only divisible by 1 and n (consider only positive integers). So, n is prime number $\Leftrightarrow$ n has no positive divisors $\in [2, \ldots, \sqrt{n}]$
- So, n is prime number $\Leftrightarrow \forall i \in [2, \ldots, \sqrt{n}], \neg(n \equiv 0 \bmod i)$

https://en.wikipedia.org/wiki/RSA_numbers

Miller-Rabin algorithm

- Verify that a positive integer *n* is prime by the above method will give correct results.
- However, the processing time of the algorithm is obviously very large, or even impossible, in the case of relatively large *n*.
- In fact, verifying that a positive integer *n* is prime often applies methods belonging to the Monte Carlo group of algorithms
- Example:
  - Algorithm Solovay-Strassen
  - Algorithm Miller-Rabin

Miller-Rabin algorithm

- Monte Carlo algorithm group
  - Used in the affirmation or negation of something. The algorithm always gives an answer and the answer obtained is only likely to be “Yes” or “No”.
  - The “yes-biased Monte Carlo” algorithm is a Monte Carlo algorithm in which the answer “Yes” is always correct but the answer “No” may not be correct.
- Miller-Rabin . Algorithm
  - Advantages of fast processing (positive integer *n* is checked in time proportional to $log_2 n$, i.e., a number of bits in the *n*’s binary representation)
  - It is possible that the algorithm's conclusion is not completely correct, that is, there is a chance that a composite number *n* will be concluded to be prime, although the probability of the incorrect conclusion is not high.
  - Overcome by executing the algorithm many times to reduce the possibility of false conclusions below the allowable threshold $\rightarrow$ highly reliable conclusions

fit@hcmus

Miller-Rabin algorithm

- The idea is based on the following two theorems
- Fermat's little theorem:
  - n is prime number $\Rightarrow a^{n-1} \equiv 1 \bmod n$ $(1 < a < n)$
  - Example $n = 561$, we have $5^{560} \equiv 1 \bmod 561$ (But $561 = 3 \times 11 \times 17$)
    - Call 561 is a Fermat pseudo-primes with base 5
  - Example $n = 341$, we have $2^{340} \equiv 1 \bmod 341$ (But $341 = 11 \times 31$)
    - Call 341 is a Fermat pseudo-primes with base 2
  - Clearly, inverting this theorem may not be correct.
  - However: $a^{n-1} \not\equiv 1 \bmod n$ $(1 < a < n) \Rightarrow n$ is composite (Correct)
- If n is prime $\Rightarrow$ '1' has 2 solutions modulo n: '1' & '$n - 1$'
  - Note: writing '-1' standing for '$n - 1$'

Miller-Rabin algorithm

- Suppose we need to check if n is prime or not?
  - Analyze $n = m \times 2^k + 1$ (m odd) $\Leftrightarrow n - 1 = m \times 2^k$
  - Randomly choose $1 < a < n - 1$ (Why not choose 1 and $n - 1$?)
  - Consider: $a^{2^k \times m}$, $a^{2^{k-1} \times m}$, $a^{2^{k-2} \times m}$, …, $a^{2^2 \times m}$, $a^{2 \times m}$, a^m
    - If $a^m \cong \pm 1 \bmod n \Rightarrow a^{2^k \times m} \cong a^{n-1} \cong 1 \bmod n$ (n is prime and stop)
    - If $a^m \not\cong 1 \bmod n$ & $a^m \not\cong n - 1 \bmod n$
      - Try $(a^m)^2 \cong n - 1 \bmod n$ ($a^{m \times 2} \not\cong 1 \bmod n$, due to $a^m \not\cong 1 \bmod n$ & $a^m \not\cong n - 1 \bmod n$)
        - If correct $\Rightarrow a^{2^k \times m} \cong a^{n-1} \cong 1 \bmod n$ (n is prime and stop)
      - Try $((a^m)^2)^2 \cong n - 1 \bmod n$ ($a^{m \times 2^2} \not\cong 1 \bmod n$, due to $a^{m \times 2} \not\cong 1 \bmod n$ & $a^{m \times 2} \not\cong n - 1 \bmod n$)
        - If correct $\Rightarrow a^{2^k \times m} \cong a^{n-1} \cong 1 \bmod n$ (n is prime and stop)
      - …
      - Try $a^{2^{k-1}} \cong n - 1 \bmod n$ ($a^{m \times 2^{k-1}} \not\cong 1 \bmod n$, due to $a^{m \times 2^{k-2}} \not\cong 1 \bmod n$ & $a^{m \times 2^{k-2}} \not\cong n - 1 \bmod n$)
        - If correct $\Rightarrow a^{2^k \times m} \cong a^{n-1} \cong 1 \bmod n$ (n is prime and stop)
        - If incorrect $\Rightarrow n$ is composite (Correct 100%)

- Suppose we need to check if n is prime or not?
  - Analyze $n = m \times 2^k + 1$ (m odd) $\Leftrightarrow n - 1 = m \times 2^k$
  - Randomly choose $1 < a < n - 1$ (Why not choose 1 and $n - 1$?)
  - Consider: , …,
    - If $a^m \cong \pm 1 \bmod n \Rightarrow \cong a^{n-1} \cong 1 \bmod n$ (n is prime and stop)
    - If a^m 1 mod n & a^m n - 1 mod n
      - Try $(a^m)^2 \cong n - 1 \bmod n$ (mod n, due to a^m 1 mod n & a^m n - 1 mod n)
        - If correct $\Rightarrow \cong a^{n-1} \cong 1 \bmod n$ (n is prime and stop)
      - Try $((a^m)^2)^2 \cong n - 1 \bmod n$ (mod n, due to $a^{m \times 2}$ 1 mod n & $a^{m \times 2}$ n - 1 mod n)
        - If correct $\Rightarrow \cong a^{n-1} \cong 1 \bmod n$ (n is prime and stop)
      - …
      - Try $\cong n - 1 \bmod n$ (mod n, due to 1 mod n & n - 1 mod n)
        - If correct $\Rightarrow \cong a^{n-1} \cong 1 \bmod n$ (n is prime and stop)
        - If incorrect $\Rightarrow n$ is composite (Correct 100%)

fit@hcmus

Miller-Rabin algorithm

- Description: positive integer analysis $n = 2^k m + 1$ with m odd
  - Randomly choose a positive integer $a \in \{1, 2, ..., n - 1\}$
  - Compute $b = a^m \bmod p$
  - if $b \equiv 1 \pmod{p}$ then “p is prime” and stop
  - for $i = 0$ to $k - 1$
    - if $b \equiv p - 1 \pmod{p}$ then “p is prime” and stop
    - else $b = b^2 \bmod p$
  - Conclude “p is composite”
- Comments:
  - A “yes-biased Monte Carlo” algorithm for the statement “positive integer n is composite”.
  - The probability of the wrong conclusion, i.e., the algorithm making the conclusion “n is prime” when n is indeed composite, is only 25%.
  - If we apply the algorithm k times with different values of a and we still get the conclusion “n is prime”, then the correct probability of this conclusion is $1 - 4^{-k} \to 1$, with large-enough k.

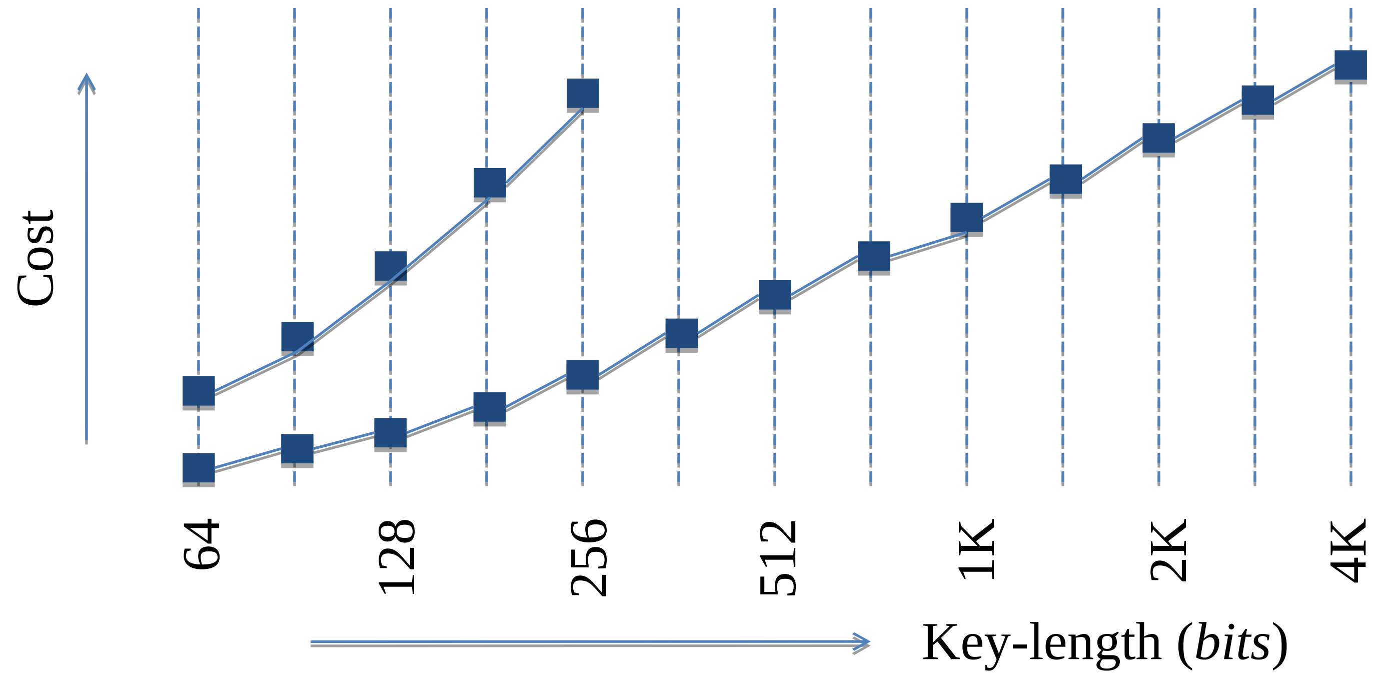
Arithmetic processing

- Compute $z = x^b \bmod n$
- Algorithm "squared and multiplied"
  - Let b be $b_{l-1}b_{l-2}\ldots b_1 b_0$, $b_i \in \{0, 1\}$, $0 \le i < l$
  - $z = 1$
  - $x = x \bmod n$
  - **for** $i = l$ **-** 1 **down to** 0
    - $z = z^2 \bmod n$
    - **if** $b_i = 1$ **then** $z = z \times x \bmod n$

Symmetry vs asymmetry

- Conventional encryption methods have the advantage of being very fast compared to public key encryption methods.
  - Because the key used for encryption is also used for decryption, it is necessary to keep the contents of the key and the key secret, called the secret key. Even in the case of a direct exchange of keys, the key is still detectable. The difficult problem for these encryption methods is the key exchange problem.
- The public key is more vulnerable than the secret-key.
  - To find the secret key, the decryptor needs some more information related to the characteristics of the source text before encrypting to find the decryption clue instead of having to use the key extraction method.
  - Determining whether the message after decryption is indeed the original message before encryption is again a difficult problem.
  - For public keys, cracking is completely possible provided there are enough resources and processing time.

Symmetry vs asymmetry



Graph comparing the cost of breaking secret-keys & public-keys