RSA Algorithm

Tyulbashev Vladislav, 106

MSU, 7.11.2013

RSA Algorithm

RSA is an algorithm for public-key cryptography that is based on the presumed difficulty of factoring large integers, the factoring problem. RSA stands for Ron Rivest, Adi Shamir and Leonard Adleman, who first publicly described the algorithm in 1977. Clifford Cocks, an English mathematician, had developed an equivalent system in 1973, but it wasn't declassified until 1997.

A user of RSA creates and then publishes the product of two large prime numbers, along with an auxiliary value, as their public key. The prime factors must be kept secret. Anyone can use the public key to encrypt a message, but with currently published methods, if the public key is large enough, only someone with knowledge of the prime factors can feasibly decode the message. Whether breaking RSA encryption is as hard as factoring is an open question known as the RSA problem.

RSA Problem

In cryptography, the RSA problem summarizes the task of performing an RSA private-key operation given only the public key. The RSA algorithm raises a message to an exponent, modulo a composite number N whose factors are not known. As such, the task can be neatly described as finding the eth roots of an arbitrary number, modulo N. For large RSA key sizes (in excess of 1024 bits), no efficient method for solving this problem is known; if an efficient method is ever developed, it would threaten the current or eventual security of RSA-based cryptosystems—both for public-key encryption and digital signatures.

More specifically, the RSA problem is to efficiently compute P given an RSA public key (N, e) and a ciphertext $C \equiv P^e \pmod{N}$. The structure of the RSA public key requires that N be a large semiprime (i.e., a product of two large prime numbers), that $2 < e < N$, that e be coprime to $\varphi(N)$, and that $0 \leq C < N$. C is chosen randomly within that range; to specify the problem with complete precision, one must also specify how N and e are generated, which will depend on the precise means of RSA random keypair generation in use.

As of 2010, the most efficient means known to solve the RSA problem is to first factor the modulus N, which is believed to be impractical if N is sufficiently large (see integer factorization). The RSA key setup routine already turns the public exponent e, with this prime factorization, into the private exponent d, and so exactly the same algorithm allows anyone who factors N to obtain the private key. Any C can then be decrypted with the private key.

Just as there are no proofs that integer factorization is computationally difficult, there are also no proofs that the RSA problem is similarly difficult. By the above method, the RSA problem is at least as easy as factoring, but it might well be easier. Indeed, there is strong evidence pointing to this conclusion: that a method to break the RSA method

cannot be converted necessarily into a method for factoring large semiprimes. This is perhaps easiest to see by the sheer overkill of the factoring approach: the RSA problem asks us to decrypt one arbitrary ciphertext, whereas the factoring method reveals the private key: thus decrypting all arbitrary ciphertexts, and it also allows one to perform arbitrary RSA private-key encryptions. Along these same lines, finding the decryption exponent d indeed is computationally equivalent to factoring N, even though the RSA problem does not ask for d. In addition to the RSA problem, RSA also has a particular mathematical structure that can potentially be exploited without solving the RSA problem directly. To achieve the full strength of the RSA problem, an RSA-based cryptosystem must also use a padding scheme like OAEP, to protect against such structural problems in RSA.

Algorithm detail description

$$a^{\varphi(m)} \equiv 1 \pmod{m}$$

1. Key generation.
   a) Generation two big prime numbers. (Using prime test [$O(\log^2)$])
   b) n = p * q
   c) ϕ(n) = ϕ(p) * ϕ(q) = (p — 1)*(q — 1).
   d) Choose such **e**, that gcd(e, ϕ(n)) = 1. **e** ~ 2^{16}.
   e) Calculate d = e^{-1} mod ϕ(n)

The **public key** consists of the modulus *n* and the public (or encryption) exponent *e*. The **private key** consists of the modulus *n* and the private (or decryption) exponent *d*, which must be kept secret. *p*, *q*, and φ(*n*) must also be kept secret because they can be used to calculate *d*.

2. Encryption.
Alice transmits her public key (n, e) to Bob and keeps the private key secret. Bob then wishes to send message M to Alice.
He first turns M into an integer m, such that 0 ≤ m < n by using an agreed-upon reversible protocol known as a padding scheme. He then computes the ciphertext c corresponding to
$c = m^e \pmod{n}$
This can be done quickly using the method of exponentiation by squaring. Bob then transmits c to Alice.

3. Decryption.
Alice can recover m from c by using her private key exponent d via computing
$m = c^d \pmod{n}$.
Given m, she can recover the original message M.

Speed

RSA Algorithm is slow, because of many multiplications. You can encrypt/decrypt at speed of several kb/sec. It is too slow to use it for encrypting session. So, usually, there is an AES session. AES is symmetric algorithm. It is security and fast. Servers swap key for session, by using rsa. So, we have fast and security session.

Current situation and conclusion

We have recently been informed by a large corporation, that NSA breaked algorithm. Algorithm is perfect, but there is a problem when we generate big random numbers. Backdoor from NSA is in generator. They can predict 'random' numbers that it gives to us. So, there is no security in our world. Expertes think that there are for about half realizations of RSA Algorithm are compromised. RSA is using everywhere — https, ssl, money transfers. So, we have a big problem with NSA now. Also there is a problem with a Quantum computer. It can break RSA. But we don't have Quantum computer, and can't check it.

Sources

http://habrahabr.ru/post/99376/

http://habrahabr.ru/post/200858/

Gary L. Miller, "Riemann's Hypothesis and Tests for Primality

Don Coppersmith, "Small Solutions to Polynomial Equations, and Low Exponent RSA Vulnerabilities", Journal of Cryptology, v. 10, n. 4, Dec. 1997