Volume 2, Issue 2 February 2016 ISSN NO: 2454 - 7514

North Asian International Research Journal Consortium

North Asian International Research Journal Of Science, Engineering and Information Technology

NAIRJC JOURNAL PUBLICATION
North Asian International Research Journal Consortium
Nairjc.... North Asian International Research Journal Consortium

Welcome to NAIRJC

ISSN NO: 2454 -7514

North Asian International Research Journal of Science, Engineering & Information Technology is a research journal, published monthly in English, Hindi, Urdu all research papers submitted to the journal will be double-blind peer reviewed referred by members of the editorial board. Readers will include investigator in Universities, Research Institutes Government and Industry with research interest in the general subjects

Editorial Board

Address: - North Asian International Research Journals - 221 Gangoo, Pulwama, Jammu and Kashmir, India -192301, Cell: 09086405302, 09906662570, Ph. No: 01933-212815,
Email: nairjc5@gmail.com, nairjc@nairjc.com, info@nairjc.com Website: www.nairjc.com

A RSA ALOGORITHM OF CRYPTOGRAPHY IN NETWORK SECURITY AUTHORING INTERFACE with domain implementation

C.SURESH[1], V.VIDHYA[2], E.SHAMLI[3], R.MUTHULAKSHMI[4], P.MAHALAKSHMI[5]

[1]B.Tech Information Technology (Final year), vcsureshbala@gmail.com

[2] B.E Computer Science Engineering (Final year), vidhyaveeramanibe@gmail.com

[3] B.E Computer Science Engineering (Final year), shamlielumalai@gmail.com

[4] B.E Computer Science Engineering (third year), muthurameshcse@gmail.com

[5] B.E Civil Engineering (Final year),mahapalani01i@gmail.com

Mailam Engineering College, Mailam-604304, India, 04145-241515

ABSTRACT

In cryptography, **RSA** (which stands for Rivest, Shamir and Adleman who first publicly described it) is an algorithm for public-key cryptography. It is the first algorithm known to be suitable for signing as well as encryption, and was one of the first great advances in public key cryptography. RSA is widely used in electronic commerce protocols, and is believed to be secure given sufficiently long keys and the use of up-to-date implementations.

HISTORY

Clifford Cocks, a British mathematician working for the UK intelligence agency GCHQ, described an equivalent system in an internal document in 1973, but given the relatively expensive computers needed to implement it at the time, it was mostly considered a curiosity and, as far as is publicly known, was never deployed. His discovery, however, was not revealed until 1997 due to its top-secret classification, and Rivest, Shamir, and Adleman devised RSA independently of Cocks' work.

The RSA algorithm was publicly described in 1978 by Ron Rivest, Adi Shamir, and Leonard Adleman at MIT; the letters **RSA** are the initials of their surnames, listed in the same order as on the paper. MIT was granted U.S. Patent 4,405,829 for a "Cryptographic communications system and method" that used the algorithm in 1983. The patent would have expired on September 21, 2000 (the term of patent was 17 years at the time), but the algorithm was released to the public domain by RSA Security on 6 September 2000, two weeks earlier.[2] Since a paper describing the algorithm had been published in August 1977,[1] prior to the December 1977 filing date of the patent application, regulations in much of the rest of the world precluded patents elsewhere and only the US patent was granted. Had Cocks' work been publicly known, a patent in the US might not have been possible.

From the DWPI's abstract of the patent,

The system includes a communications channel coupled to at least one terminal having an encoding device and to at least one terminal having a decoding device. A message-to-be-transferred is enciphered to cipher text at the encoding terminal by encoding the message as a number M in a predetermined set. That number is then raised to a first predetermined power (associated with the intended receiver) and finally computed. The remainder or residue, C, is computed when the exponentiated number is divided by the product of two predetermined prime numbers (associated with the intended receiver).

OPERATION

The RSA algorithm involves three steps: **key generation, encryption and decryption**.

RSA involves a **public** key and a **private key.** The public key can be known to everyone and is used for encrypting messages. Messages encrypted with the public key can only be decrypted using the private key. The keys for the RSA algorithm are generated the following way:

1. Choose two distinct prime numbers p and q.
    - For security purposes, the integers p and q should be chosen uniformly at random and should be of similar bit-length. Prime integers can be efficiently found using a primality test.
2. Compute $n = pq$.
    - n is used as the modulus for both the public and private keys
3. Compute $\varphi(pq) = (p - 1)(q - 1)$.. ($\varphi$ is Euler's totient function).
4. Choose an integer e such that $1 < e < \varphi(pq)$, and e and $\varphi(pq)$ share no divisors other than 1 (i.e. e and $\varphi(pq)$ are coprime).
    - e is released as the public key exponent.
    - Choosing e having a short addition chain results in more efficient encryption. Small public exponents (such as $e = 3$) could potentially lead to greater security risks.[3] In practice, e is usually $2^{16}+1$.
5. Determine d (using modular arithmetic) which satisfies the congruence relation .
    - Stated differently, $ed - 1$ can be evenly divided by the totient $(p - 1)(q - 1)$.
    - This is often computed using the extended Euclidean algorithm.
    - d is kept as the private key exponent.

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.

Note:

- PKCS#1 v2.0 and PKCS#1 v2.1 specifies using the Carmichael function instead of Euler's totient, where lcm is the least common multiple.
- For efficiency the following values may be precomputed and stored as part of the private key:
    - p and q: the primes from the key generation,
    - and ,
    - .

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 $0 < m < n$ by using an agreed-upon reversible protocol known as a padding scheme. He then computes the ciphertext c corresponding to:

This can be done quickly using the method of exponentiation by squaring. Bob then transmits c to Alice.

DECRYPTION

Alice can recover m from c by using her private key exponent d by the following computation:

Given m, she can recover the original message **M** by reversing the padding scheme.

(In practice, there are more efficient methods of calculating c^d using the pre computed values above.)

A WORKED EXAMPLE

Here is an example of RSA encryption and decryption. The parameters used here are artificially small, but one can also use OpenSSL to generate and examine a real keypair.

1. Choose two prime numbers

   $p = 61$ and $q = 53$

2. Compute $n = pq$
3. Compute the product of totients. For primes the totient is maximal and equals $x - 1$. Therefore
4. Choose any number $e > 1$ that is coprime to 3120. Choosing a prime number for e leaves you with a single check: that e is not a divisor of 3120.

   $e = 17$

5. Compute d such that e.g., by computing the modular multiplicative inverse of e modulo :

   $d = 2753$

   since $17 \cdot 2753 = 46801$ and 46801 mod 3120 = 1, this is the correct answer.

   (iterating finds (**15** times 3120)+1 divided by 17 is 2753, an integer, whereas other values in place of 15 do not produce an integer. The extended euclidean algorithm finds the solution to Bézout's identity of 3120x2 + 17x-367=1, and -367 mod 3120 is 2753)

The **public key** is (n = 3233, e = 17). For a padded message m the encryption function is or abstractly:

The **private key** is (n = 3233, d = 2753). The decryption function is or in its general form:

For instance, in order to encrypt m = 123, we calculate

To decrypt c = 855, we tap

.

Both of these calculations can be computed efficiently using the square-and-multiply algorithm for modular exponentiation. In real life situations the primes selected would be much larger, however in our example it would be relatively trivial to factor n, 3233, obtained from the freely available public key back to the primes p and q. Given e, also from the public key, we could then compute d and so acquire the private key.

PADDING SCHEMES

When used in practice, RSA is generally combined with some padding scheme. The goal of the padding scheme is to prevent a number of attacks that potentially work against RSA without padding:

- When encrypting with low encryption exponents (e.g., e = 3) and small values of the m, (i.e. $m < n^{1/e}$) the result of m^e is strictly less than the modulus n. In this case, ciphertexts can be easily decrypted by taking the eth root of the ciphertext over the integers.

- If the same clear text message is sent to e or more recipients in an encrypted way, and the receivers share the same exponent e, but different p, q, and n, then it is easy to decrypt the original clear text message via the Chinese remainder theorem. Johan Håstad noticed that this attack is possible even if the cleartexts are not equal, but the attacker knows a linear relation between them.[4] This attack was later improved by Don Coppersmith.[5]
- Because RSA encryption is a deterministic encryption algorithm – i.e., has no random component – an attacker can successfully launch a chosen plaintext attack against the cryptosystem, by encrypting likely plaintexts under the public key and test if they are equal to the ciphertext. A cryptosystem is called semantically secure if an attacker cannot distinguish two encryptions from each other even if the attacker knows (or has chosen) the corresponding plaintexts. As described above, RSA without padding is not semantically secure.
- RSA has the property that the product of two ciphertexts is equal to the encryption of the product of the respective plaintexts. That is Because of this multiplicative property a chosen-ciphertext attack is possible. E.g. an attacker, who wants to know the decryption of a ciphertext $c = m^e \bmod n$ may ask the holder of the private key to decrypt an unsuspicious-looking ciphertext $c' = cr^e \bmod n$ for some value r chosen by the attacker. Because of the multiplicative property c' is the encryption of $mr \bmod n$. Hence, if the attacker is successful with the attack, he will learn $mr \bmod n$ from which he can derive the message m by multiplying mr with the modular inverse of r modulo n.

To avoid these problems, practical RSA implementations typically embed some form of structured, randomized padding into the value m before encrypting it. This padding ensures that m does not fall into the range of insecure plaintexts, and that a given message, once padded, will encrypt to one of a large number of different possible ciphertexts.

Standards such as PKCS#1 have been carefully designed to securely pad messages prior to RSA encryption. Because these schemes pad the plaintext m with some number of additional bits, the size of the un-padded message **M** must be somewhat smaller. RSA padding schemes must be carefully designed so as to prevent sophisticated attacks which may be facilitated by a predictable message structure. Early versions of the PKCS#1 standard (up to version 1.5) used a construction that turned RSA into a semantically secure encryption scheme. This version was later found vulnerable to a practical adaptive chosen ciphertext attack. Later versions of the standard include Optimal Asymmetric Encryption Padding (OAEP), which prevents these attacks. The PKCS#1 standard also incorporates processing schemes designed to provide additional security for RSA signatures, e.g., the Probabilistic Signature Scheme for RSA (RSA-PSS).

In the common case where RSA is used to exchange symmetric keys, key encapsulation provides a simpler alternative to padding. Instead of generating a random symmetric key, padding it and then encrypting the padded version with RSA, a random integer m between 1 and n-1 is generated and encrypted directly using RSA. Both the sender and receiver generate identical symmetric keys by applying the same key derivation function to *m*.[6]

SIGNING MESSAGES

Suppose Alice uses Bob's public key to send him an encrypted message. In the message, she can claim to be Alice but Bob has no way of verifying that the message was actually from Alice since anyone can use Bob's public key to send him encrypted messages. So, in order to verify the origin of a message, RSA can also be used to sign a message. Suppose Alice wishes to send a signed message to Bob. She can use her own private key to do so. She produces a hash value of the message, raises it to the power of *d*mod *n* (as she does when decrypting a message), and attaches it as a "signature" to the message. When Bob receives the signed message, he uses the same hash algorithm in conjunction with Alice's public key. He raises the signature to the power of *e*mod *n* (as he does when encrypting a message), and compares the resulting hash value with the message's actual hash value. If the two agree, he knows that the author of the message was in possession of Alice's private key, and that the message has not been tampered with since.

Note that secure padding schemes such as RSA-PSS are as essential for the security of message signing as they are for message encryption, and that the same key should never be used for both encryption and signing purposes[*citation needed*].

Security and practical considerations

INTEGER FACTORIZATION AND RSA PROBLEM

See also: RSA Factoring Challenge and Integer factorization records

The security of the RSA cryptosystem is based on two mathematical problems: the problem of factoring large numbers and the RSA problem. Full decryption of an RSA ciphertext is thought to be infeasible on the assumption that both of these problems are hard, i.e., no efficient algorithm exists for solving them. Providing security against *partial* decryption may require the addition of a secure padding scheme.[*citation needed*]

The RSA problem is defined as the task of taking *e*th roots modulo a composite *n*: recovering a value *m* such that $c = m^e \bmod n$, where (n,e) is an RSA public key and *c* is an RSA ciphertext. Currently the most promising approach to solving the RSA problem is to factor the modulus *n*. With the ability to recover prime factors, an attacker can compute the secret exponent *d* from a public key (n,e), then decrypt *c* using the standard procedure. To accomplish this, an attacker factors *n* into *p* and *q*, and computes $(p - 1)(q - 1)$ which allows the determination of *d* from *e*. No polynomial-time method for factoring large integers on a classical computer has yet been found, but it has not been proven that none exists. See integer factorization for a discussion of this problem.

As of 2010, the largest (known) number factored by a general-purpose factoring algorithm was 768 bits long (see RSA-768), using a state-of-the-art distributed implementation. RSA keys are typically 1024–2048 bits long. Some experts believe that 1024-bit keys may become breakable in the near term (though this is disputed); few see any way that 4096-bit keys could be broken in the foreseeable future. Therefore, it is generally presumed that RSA is secure if *n* is sufficiently large. If *n* is 300 bits or shorter, it can be factored in a few hours on a personal computer, using software already freely available. Keys of 512 bits have been shown to be practically breakable in 1999 when RSA-155 was factored by using several hundred computers and are now factored in a few weeks using common hardware.[7] A theoretical hardware device named

TWIRL and described by Shamir and Tromer in 2003 called into question the security of 1024 bit keys. It is currently recommended that n be at least 2048 bits long.[8]

In 1994, Peter Shor showed that a quantum computer could factor in polynomial time, breaking RSA.

KEY GENERATION

Finding the large primes p and q is usually done by testing random numbers of the right size with probabilistic primality tests which quickly eliminate virtually all non-primes.

Numbers p and q should not be 'too close', lest the Fermat factorization for n be successful, if $p - q$, for instance is less than $2n^{1/4}$ (which for even small 1024-bit values of n is 3×10^{77}) solving for p and q is trivial. Furthermore, if either $p - 1$ or $q - 1$ has only small prime factors, n can be factored quickly by Pollard's $p - 1$ algorithm, and these values of p or q should therefore be discarded as well.

It is important that the private key d be large enough. Michael J. Wiener showed[9] that if p is between q and $2q$ (which is quite typical) and $d < n^{1/4}/3$, then d can be computed efficiently from n and e. There is no known attack against small public exponents such as $e = 3$, provided that proper padding is used. However, when no padding is used or when the padding is improperly implemented then small public exponents have a greater risk of leading to an attack, such as for example the unpadded plaintext vulnerability listed above. 65537 is a commonly used value for e. This value can be regarded as a compromise between avoiding potential small exponent attacks and still allowing efficient encryptions (or signature verification). The NIST Special Publication on Computer Security (SP 800-78 Rev 1 of August 2007) does not allow public exponents e smaller than 65537, but does not state a reason for this restriction.

SPEED

RSA is much slower than DES and other symmetric cryptosystems. In practice, Bob typically encrypts a secret message with a symmetric algorithm, encrypts the (comparatively short) symmetric key with RSA, and transmits both the RSA-encrypted symmetric key and the symmetrically-encrypted message to Alice.

This procedure raises additional security issues. For instance, it is of utmost importance to use a strong random number generator for the symmetric key, because otherwise Eve (an eavesdropper wanting to see what was sent) could bypass RSA by guessing the symmetric key.

KEY DISTRIBUTION

As with all ciphers, how RSA public keys are distributed is important to security. Key distribution must be secured against a man-in-the-middle attack. Suppose Eve has some way to give Bob arbitrary keys and make him believe they belong to Alice. Suppose further that Eve can *intercept* transmissions between Alice and Bob. Eve sends Bob her own public key, which Bob believes to be Alice's. Eve can then intercept any ciphertext sent by Bob, decrypt it with her own private key, keep a copy of the message, encrypt the message with Alice's public key, and send the new ciphertext to Alice. In principle, neither Alice nor Bob would be able to detect Eve's presence. Defenses against such attacks are often based on digital certificates or other components of a public key infrastructure.

TIMING ATTACKS

Kocher described a new attack on RSA in 1995: if the attacker Eve knows Alice's hardware in sufficient detail and is able to measure the decryption times for several known ciphertexts, she can deduce the decryption key *d* quickly. This attack can also be applied against the RSA signature scheme. In 2003, Boneh and Brumley demonstrated a more practical attack capable of recovering RSA factorizations over a network connection (e.g., from a Secure Socket Layer (SSL)-enabled webserver). This attack takes advantage of information leaked by the Chinese remainder theorem optimization used by many RSA implementations.

One way to thwart these attacks is to ensure that the decryption operation takes a constant amount of time for every ciphertext. However, this approach can significantly reduce performance. Instead, most RSA implementations use an alternate technique known as cryptographic blinding. RSA blinding makes use of the multiplicative property of RSA. Instead of computing c^dmod n, Alice first chooses a secret random value r and computes $(r^e c)^d$mod n. The result of this computation is and so the effect of r can be removed by multiplying by its inverse. A new value of r is chosen for each ciphertext. With blinding applied, the decryption time is no longer correlated to the value of the input ciphertext and so the timing attack fails.

Adaptive chosen ciphertext attacks

In 1998, Daniel Bleichenbacher described the first practical adaptive chosen ciphertext attack, against RSA-encrypted messages using the PKCS #1 v1 padding scheme (a padding scheme randomizes and adds structure to an RSA-encrypted message, so it is possible to determine whether a decrypted message is valid.) Due to flaws with the PKCS #1 scheme, Bleichenbacher was able to mount a practical attack against RSA implementations of the Secure Socket Layer protocol, and to recover session keys. As a result of this work, cryptographers now recommend the use of provably secure padding schemes such as Optimal Asymmetric Encryption Padding, and RSA Laboratories has released new versions of PKCS #1 that are not vulnerable to these attacks.

Branch prediction side-channel analysis attacks

A side-channel attack using branch prediction analysis (BPA) has been described. Many processors use a branch predictor to determine whether a conditional branch in the instruction flow of a program is likely to be taken or not. Usually these processors also implement simultaneous multithreading (SMT). Branch prediction analysis attacks use a spy process to discover (statistically) the private key when processed with these processors.

Simple Branch Prediction Analysis (SBPA) claims to improve BPA in a non-statistical way. In their paper, "On the Power of Simple Branch Prediction Analysis"[10], the authors of SBPA (Onur Aciicmez and Cetin Kaya Koc) claim to have discovered 508 out of 512 bits of an RSA key in 10 iterations.

APPLICATIONS

• RSA is widely used for encryption and decryption in message communication for making the communication secure.
• It is used for digital signature.
• It is used for key distribution.
• RSA is used in e-commerce and remote banking.

CONCLUSION

After the study, I find that RSA is a powerful and most widely used scheme for encryption / decryption and digital signature. It is more secure than that of DES and others. But as we know that the key length for secure RSA use has increased over recent years, and this has put a heavier processing load on applications using RSA. This burden has ramifications, especially for electronic commerce sites that conduct large numbers of transactions. Recently, a competing system has begun to challenge RSA: elliptic curve cryptography (ECC). The principal attraction of ECC, compared to RSA, is that it appears to offer equal security for a far smaller key size; thereby reducing processing overhead but the confidence level in ECC is not yet as high as that is in RSA. Also RSA is fundamentally easier to explain than that of ECC.

REFERENCES

[1] Lu Wang. and Kais Hun wu S. (2013) 'Attached-RTS: Eliminating an Exposed Terminal Problem in Wireless Network' IEEE Transactions on Parallel and Distributed Systems', Vol.24, No.7, pp. 1289-1299.

[2] Mishra, R. Kayak, S. Verna, K. and Singh, D. (2011) 'Survey on Techniques to Resolve Problems Associated with RTS/CTS Mechanism', Proc. Int'l Conf. Comm., Computing and Security (ICCCS), Vol.3, No.1, ISSN 2250-3501.

[3] Halpern, D. Hu, W. Sheath, A. and Wetherill, D. (2008) 'Predictable 802.11 Packet Delivery from Wireless Channel Measurements', Proc. ACM SIGCOMM, Vol.7, No.1, pp. 135-145.

[4] He, N. Xu, Y. Cao, J. Li, Z. Chen, H. and Ran, Y. (2010) 'ROME: Rattles Online MDS Code for Wireless Data Broadcasting', in Proc. of IEEE Globecom, Vol.32,No.3,pp. 1243-1265

[5] LAN MAN Standards Committee of the IEEE Computer Society. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. ANSI/IEEE Std. 802.11, 1999 Edition.

[6] V.r.Bharghavan, A. Demers, S. Shenker, and L. Zhang, "MACAW: A media access protocol for wireless LANs," in *Proceedings of ACM SIGCOMM '94*.1994, pp. 212–225, ACM.

[7] L. Kleinrock and F. A. Tobagi, "Packet switching in radio channels: Part 2 - the hidden node problem in carrier sense multiple access modes and the busy tone solution," *IEEE Transactions on Communications*, vol. COM-23, no. 12, pp. 1417–1433, 1975.

[8] C .K. Toh, *Ad Hoc Mobile Wireless Networks: Protocols and Systems*, Prentice Hall, December 2001.

[9] IEEE 802.11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.

IEEE-SA.2007.

[10] R. Mishra, S. Nayak, K. Verma, D. Singh, "Survey on techniques to resolve problems associated with RTS/CTS mechanism", in *Pro .of ICCCS*, 2011.

[11] M. Vutukuru, K. Jamieson, H. Balakrishnan, " Harnessing exposed terminals in wireless networks", in *Proc. of USENIX NSDI*,

2008.

[12] D. Halperin, T. Anderson, and D. Wetherall, "Taking the sting out of carrier sense: Interference Cancellation for wireless LANs", in *Proc. of ACM MOBICOM*, 2008.

Publish Research Article