Network Security

Sasikumar Gurumurthy, *Member, IAENG*

***Abstract*— Network security is a complicated subject, historically only tackled by well-trained and experienced experts. However, as more and more people become ``wired'', an increasing number of people need to understand the basics of security in a networked world. This document was written with the basic computer user and information systems manager in mind, explaining the concepts needed to read through the hype in the marketplace and understand risks and how to deal with them. Some history of networking is included, as well as an introduction to TCP/IP and *internetworking*. We go on to consider risk management, network threats, firewalls, and more special-purpose secure networking devices. This is not intended to be a ``frequently asked questions'' reference, nor is it a ``hands-on'' document describing how to accomplish specific functionality. It is hoped that the reader will have a wider perspective on security in general, and better understand how to reduce and manage risk personally, at home, and in the workplace.**

I. Introduction

A basic understanding of computer networks is requisite in order to understand the principles of network security. In this section, we'll cover some of the foundations of computer networking, then move on to an overview of some popular networks.Following that, we'll take a more in-depth look at TCP/IP, the network protocol suite that is used to run the Internet and many intranets. Once we've covered this, we'll go back and discuss some of the threats that managers and administrators of computer networks need to confront, and then some tools that can be used to reduce the exposure to the risks of network computing.

A. What is a Network?

A ``network'' has been defined as ``any set of interlinking lines resembling a net, *a network of roads* |an interconnected system, *a network of alliances*.'' This definition suits our purpose well: a computer network is simply a system of interconnected computers. *How* they're connected is irrelevant, and as we'll soon see, there are a number of ways to do this

B. Cryptography

Cryptography (or cryptology; derived from Greek κρύπτω *krýpto* "hidden" and the verb γράφω *gráfo* "to write" or λέγειν *legein* "to speak")[1] is the practice and study of hiding information. In modern times, cryptography is considered a branch of both mathematics and computer science, and is affiliated closely with information theory, computer security, and engineering.

Sasikumar Gurumurthy is with the VIT University, Vellore, Tamil Nadu, India. He is now with the School of computing science and Engineering (corresponding author to provide phone: 0416-2202016; fax: 0416-2243092; e-mail: g.sasikumar@vit.ac.in).

Cryptography is used in applications present in technologically advanced societies; examples include the security of ATM cards, computer passwords, and electronic commerce, which all depend on cryptography.

C. Cryptography RSA

In cryptography, **RSA** is an algorithm for public-key cryptography. It was the first algorithm known to be suitable for signing as well as encryption, and 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.

II. History Of Network

The algorithm was publicly described in 1977 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. 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. MIT was granted US patent 4405829 for a "Cryptographic communications system and method" that used the algorithm in 1983. The patent expired on 21 September 2000. 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 either.

A. Operation

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 large random prime numbers *p* and *q*
2. Compute
   - is used as the modulus for both the public and private keys
3. Compute the totient.
4. Choose an integer *e* such that , and *e* and share no factors other than 1 (i.e. *e* and are coprime)

ISBN: 978-988-17012-9-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

- e is released as the public key exponent

5. Compute d to satisfy the congruence relation ; i.e. for some integer k.
   - d is kept as the private key exponent

Notes on the above steps:

Step 1: Numbers can be probabilistically tested for primality.
Step 3: changed in PKCS#1 v2.0 to, where LCM is the least common multiple, instead of.
Step 4: A popular choice for the public exponents is = 2^{16} + 1 = 65537. Some applications choose smaller values such as = 3, 5, 17 or 257 instead. This is done to make encryption and signature verification faster on small devices like smart cards but small public exponents can lead to greater security risks.[2]
Steps 4 and 5 can be performed with the extended Euclidean algorithm; see modular arithmetic.

The **public key** consists of the modulus and the public (or encryption) exponent. The **private key** consists of the modulus and the private (or decryption) exponent which must be kept secret. For efficiency a different form of the **private key** can be stored:

- and : the primes from the key generation,
- and ,
- .

All parts of the private key must be kept secret in this form. and are sensitive since they are the factors of, and allow computation of given . If and are not stored in this form of the private key then they are securely deleted along with other intermediate values from key generation. Although this form allows faster decryption and signing by using the Chinese Remainder Theorem, it is considerably less secure since it enables side channel attacks. This is a particular problem if implemented on smart cards, which benefit most from the improved efficiency. (Start with $y = x^e mod n$ and let the card decrypt that. So it computes $y^d (\text{mod } p)$ or $y^d (\text{mod } q)$ whose results give some value z. Now, induce an error in one of the computations. Then gcd $(z - x, n)$ will reveal p or q.).

III. Messages

A. Encryption

Alice transmits her public key to Bob and keeps the private key secret. Bob then wishes to send message **M** to Alice. He first turns **M** into a number < by using an agreed-upon reversible protocol known as a padding scheme. He then computes the ciphertext corresponding to: This can be done quickly using the method of exponentiation by squaring. Bob then transmits to Alice.

B. Decryption

Alice can recover from by using her private key exponent by the following computation: Given, she can recover the original message **M**. The above decryption procedure works because first Now, and hence and which can also be written as and for proper values of and. If is not a multiple of then and are coprime because is prime; so by Fermat's little theorem and therefore, using the first expression for, If instead is a multiple of , then Using the second expression for , we similarly conclude that Since and are distinct prime numbers, they are relatively prime to each other, so the fact that both primes divide $m^{ed} - m$ implies their product divides $m^{ed} - m$, which means Thus, 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
   $n = 61 * 53 = 3233$
3. Compute the totient
4. Choose $e > 1$ coprime to 3120
   $e = 17$
5. Compute such that e.g., by computing the modular multiplicative inverse of e modulo :
   $d = 2753$
   $17 * 2753 = 46801 = 1 + 15 * 3120$.

The **public key** is (n = 3233, e = 17). For a padded message the encryption function is: The **private key** is (n = 3233, d = 2753).The decryption function is: For example, to encrypt $m = 123$, we calculate, To decrypt c = 855, we calculate. Both of these calculations can be computed efficiently using the square-and-multiply algorithm for modular exponentiation.

C. 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 receiver's shares 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 [3]. This attack was later improved by Don Coppersmith [4].

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

ISBN: 978-988-17012-9-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

ciphertext $c=m^e \bmod n$ may ask the holder of the secret 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).

D. 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 $d \bmod n$ (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 $e \bmod n$ (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 secret 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.

E. Security

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 scheme. The RSA problem is defined as the task of taking eth 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 2005, the largest number factored by a general-purpose factoring algorithm was 663 bits long (see RSA-200), 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 256 bits or shorter, it can be factored in a few hours on a personal computer, using software already freely available. Keys of 512 bits (or less) have been shown to be practically breakable in 1999 when RSA-155 was factored by using several hundred computers.

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. In 1994, Peter Shor published Shor's algorithm, showing that a quantum computer could in principle perform the factorization in polynomial time. However, quantum computation is still in the early stages of development and may never prove to be practical.

IV. Practical Considerations

A. 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. 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 $3x10^{77}$) 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 secret key d be large enough. Michael J. Wiener showed[5] 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,

ISBN: 978-988-17012-9-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

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.

B. 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.

C. 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 secret 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.

D. 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 $c^d \bmod n$, Alice first chooses a secret random value r and computes $(r^e c)^d \bmod n$. The result of this computation is $r\ m \bmod n$ 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.

E. 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.

F. Branch Prediction analysis attacks

Branch prediction analysis is also called BPA. 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", 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.

G. RSA as an Internet meme

This section needs additional citations for verification. Please help improve this article by adding reliable references. Unsourced material may be challenged and removed. During the mid-1990s, the RSA algorithm, specifically a highly condensed implementation in Perl[6], became the subject of a widespread protest of United States encryption regulations, which at the time treated encryption algorithm as weapons and prohibited the export of strong encryption software such as RSA, and in particular the actions of the government against Phil Zimmerman, the creator of Privacy, Which had leaked out over the Internet in violation of the laws at the time.

The canonical implementation was designed to fit in a signature file used by a Usenet newsreader or email client, and was dependent on a Unix environment, particularly the dc calculator program, for its mathematical operations; however, portable pure Perl implementations also exist.

Protest websites allowed users within the US to trivially violate the export regulations by emailing copies of the perl script to servers outside the United States, while net

ISBN: 978-988-17012-9-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

personality and online activist Joel Furr, among others, marketed t-shirts containing the offending code, and protesters continued to find ever more esoteric ways to flout the rules, including but not limited to tattoos and mailing labels. The regulations were later relaxed.

V. Conclusion

Even large traditional companies such as Dow Chemical, Boeing, AlliedSignal, and General Electric are installing sophisticated e-business systems. It is unlikely that jumbo jets or railroad engines will be purchased on the Internet in the same way that consumer items are purchased with a single mouse click, but the Internet can be used to make available product information and facilitate data transfer. Analysts expect more than 30 percent of all business-to-business commerce to take place on the Internet before 2005 (Bourge, 1999, p. 2). General Electric already uses e-commerce to buy commodity items and is saving an estimated 10 percent to 15 percent when compared with non-Internet purchasing. Because business supply chains are Some common words found in the essay are: Infrastructure PKI, Dow Chemical, Long-term Internet, Internet Trade, Internet Analysts, Companies Verisign, SET SET, AlliedSignal Electric, Sales Internet, Americans Internet, public key, pc week, public key certificates, digital certificates, security solutions, souccar 1999, private key, integrity non-repudiation, quantum cryptography, key certificates, certificates digital, public key infrastructure, secure sockets layer, souccar 1999 1, authentication integrity non-repudiation.

Acknowledgment

The author thanks to the Department of computer science, VIT University and Special thanks to Dr.M.Khalid, Director, SCSE, for his kind financial support. This work has been (Partially) supported by the research program in SCSE, VIT University, India.

References

[1] William stallings, cryptography and network security: principles and practice, third edition, upper saddle river, NJ: prentice hall, 1999.
[2] Scheneier.B. applied cryptography, Newyork: wiley, 1996.
[3] Johann van der Merwe, Dawouda.Dawoud, Stephen McDonald,(April2007)'A Fully Distributed Proactively Secure Threshold-Multisignature cheme',IEEE Transactions on parallel and Distributed systems,vol18,no.4.
[4] L.Harn and Y. Xu, (1994) 'Design of Generalized Elgamal Type Digital Signature Schemes Based on Discrete Logartihms'ElectronicsLetters,Vol.30,no. 24,pp-2025-2026.
[5] T. M. Wong, C. Wang, and J.M.Wing, (Dec 2002) 'Verifiable Secret Redistribution for Archive System', Proc. First Int'l IEEE Security in storage Workshop.
[6] L. Zhou and Z.J. Haas, (1999)'Securing Ad Hoc Networks', IEEENetwork, special issue on network Security, vol. 13, no. 6, pp. 24-30.
[7] William Stallings, 'Cryptography and network security-principles and Practices' Prentice Hall of India, 3 rd Edition.

ISBN: 978-988-17012-9-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)