Internet Security – Cryptography – A Broader Look Inside

Presented by: D.R.Esesve, IT032022, SCIT

Abstract

The modern world is a real global village; it is getting smaller every day. We have a myriad of electronic gizmos that connect us to anyone or anything we want. A greater development in the same path is the Internet, a repository of never-ending information on any topic under the sun.

The most important thing while sharing information is security of the data. With data worth billions being exchanged, security becomes the key word. This paper discusses the different security mechanisms, their pros and cons. Entire discussion about each and every technology of the same is beyond the scope of the paper.

Introduction

Why do we need security?

Before the Internet changed the ways we communicate, cryptography, codes and ciphers were the province of government secrets, of a **J**ames **B**ond world where valuable but dangerous information was scrambled to ensure that it remained **'**For your eyes only**'**. For the rest of us, the seal of an envelope was sufficient proof that our mail had not been intercepted or tampered with. With the advent of the Internet, we could not be so confident about the privacy or secrecy of our communication. As data – whether it's personal correspondence, company trade secrets, financial data or our grocery shopping list – travels across the Internet, it passes through many computers. Because it is digital data rather than sealed pieces of paper, we have no way of knowing how many people have seen it, copied it, or had access to it. That is, without the security provided by the advanced cryptography which is now an everyday feature of many Internet services. Cryptographic security systems provide confidentiality, message integrity, authentication, and non-repudiation.

Why is cryptography a good solution?

Cryptography is the ideal solution for ensuring privacy and security on the Internet. The computational power of both the host server and the user's PC can be used to generate the codes used to scramble data so that it can only be read by a recipient who has received the correct code or key to decrypt it back into its original form. By using long numbers and complex mathematical formulae to generate the keys used to secure data, a high level of security can be achieved. The codes are sufficiently secure that even a powerful computer could not guess them – though the science of cryptanalysis is devoted to using computers to crack codes and find weaknesses in cryptographic systems. In fact, the scrutiny of security systems ensures that weaknesses in proposed systems are found quickly, and that systems are based on standards which have undergone long term, detailed analysis and testing are likely to be more secure than proprietary systems which have not been attacked and tested.

Types of Cryptography

- Single-key (symmetric-key) cryptography

- Public/Private-key (asymmetric-key) cryptography

Using Single Key (or symmetric) keys

The simplest forms of computer-based cryptography are secret key systems. Here, the same key is used both to encrypt (scramble) and decrypt (unscramble) the data. Both the sender and the recipient therefore need copies of the same keys. Secret key systems employ shorter key lengths, requiring less processing, making them particularly suitable for handling the encryption of bulk data. With secret key security, the risk of data being read is transferred to the risk of the private key being discovered or exposed. Security efforts must therefore focus around creating an architecture that keeps the key secret and safe, and a method of distributing keys which is also safe and secure. In practice this means that private key cryptography is used to secure communications between two parties who have already established a trusted relationship, or a separate secure communications channel. Algorithms (mathematical methods) used for private key cryptography include DES (Digital Encryption Standard), triple DES and the new AES or advanced encryption standard. This new algorithm has recently been developed to overcome the problem that computers have now become so powerful that they are potentially able to crack or decode DES keys that are used in real-life settings.

- ✓ Primary benefit: Good for bulk data = quick
- ✓ Primary problem: Keys must remain secret

(I.e. must have a secure channel for exchanging keys)

Standards for Single-key Cryptography

- **D**ata **E**ncryption **S**tandard (**DES**) – defined, endorsed and regulated by the U.S. Federal government – intended for trading partners within the U.S.
- **RSA** - **RC2** and **RC4** – proprietary **RSA** standards – intended for use when one or more partner is outside the U.S.

Using Public (or asymmetric) keys

Public key cryptography is a more recent innovation – it was first used commercially during the 1970s, and has now become the mainstay of Internet security architectures. Public key systems rely on a related pair of keys, one of which is kept private and used to decrypt data (the private key), and one which is made publicly available and used to encrypt data (the public key). The ability to make the public key widely available makes it much easier to exchange information with people regardless of whether or not you have established a trust relationship. This will be the case for all organizations wanting to use the Web to communicate and transact with hundreds, thousands or even millions of customers or users. Public key algorithms in common use include RSA, which creates pairs of keys from the prime factors of very large numbers, and elliptic curve cryptography, which uses keys derived from the mathematics of complex curves. A significant consequence of using complex numbers is the strain placed upon the computers that encrypt and decrypt data. This can severely impact the performance of cryptographic systems, such as secure Web servers, and has led to a growing market for cryptographic acceleration products.

Again, public key systems rely on the security of the private key. The level of protection required depends on the level of risk involved. For anyone handling very

high value information or large volumes of sensitive data, software protection alone is not enough. In these situations the keys are usually stored in a hardware security module where they are only accessible to authorized users and systems.

Standard for Public/Private-key Cryptography

- **RSA** – Public key cryptosystem for both encryption and authentication – Invented in 1977 at MIT by Ron Rivest, Adi Shamir and Leonard Adlemen – Part of many official standards worldwide

Managing keys

Thus the risk is transferred from the problem of securing the data to the problem of securing and managing keys. Fortunately, hardware security modules can be used with key management software to provide additional security to protect the keys so that they cannot be stolen. By using a layered security approach you can ensure that there is no single point of failure as there are several systems, physical security devices, software and operating procedures, protecting the keys.

Digital Encryption Standard

This is the most widely used encryption to date. It is used to encrypt millions of files ranging from matters of national security to bank accounts, and electronic funds transfer, which is used to move billions of dollars of investor's money each hour. The encryption system protects all digital information.

It was originally conceived by Chaude Shannon in the 1940s. Shannon was a trailblazer of the information theory. Shannon first proposed the theory using multiple transpositions and substitutions. This same system was formalized at IBM in the 1970s and formalized into a standard algorithm called Lucifer.

Cryptography

DES is a block cipher, which works in blocks of 64 bits. Each block of plaintext is enciphered separately, so if a set of information to be enciphered that was 128 bits large; each 64 bit block would be enciphered separately.

The algorithm for it is complex however very straight-forward. There are two basic steps: confusion and diffusion. After each of these steps is down, a permutation or shift is performed. This process is known as a round. To complete the algorithm, this process is repeated 16 times, giving the DES a 16 round algorithm. During each round, the 64-bit block is split. Each half is expanded to 48 bits, and then substituted with 48 bits of the key. The halves are then shifted using a permutation via an exclusive-or. The halves then go through another substitution algorithm and permutated again. These four steps make up the function. As with most ciphers, the heart of the algorithm is in the key. The key for the DES is a 56-bit number in a hexadecimal format. There are multiple keys, and hence, weak keys can be weeded out. There is a different 48-bit key produced for every round of the algorithm.

Cryptanalysis

56 keys and theoretically we could search all of them in a short amount of time. Some have estimated the cost of building the computer to do so (because of the special chips required) would be in excess of $20,000,000. Which as large a number as it may seem, is horrifically possible considering the importance of the DES as a whole. This has raised speculation for the

development of a large key, such as the one first developed for Lucifer by IBM which measured 128 bits.

Other proposed attacks, have included theories of differential cryptanalysis, which in this case would be the analyses of pairs of cipher text and pairs of plaintext. Certain patterns could theoretically show up in both, and these characteristics could help to decipher the code.

Software Implementations

Naturally we would have to have software to encode and decode for the DES. Versions of the DES on IBM mainframes can perform 32,000 encryptions per second.

Improved DES

Overview

I-DES takes a random n bit key > 56 bits long, and uses that key to select 8 strong S-boxes, and then encrypt the plaintext using 32 round DES with the alternate S-boxes. This is done initially by hashing the n bit key to a 128 bit key, and then dividing the 128 bit block. The n bit key is also hashed to a 56 bit block explained later. The 128 bit block is then divided into 8, 16 bit blocks. Each 16 bit block is used to determine the coordinates of a location on a 16*16*16*16 hyper-cube. Each location in the hyper-cube contains several pieces of information:

- A 56 bit block.
- A 56 bit permutation function.
- An S-box.
- A 16 bit block.
- A 16 bit permutation function.

Each permutation function could contain one of several types of functions, such as a simple XOR, perhaps a bitwise AND, a concatenation and one of several available has functions, or nearly anything else.

Once the coordinate is located, first the 56 bit block which is to be used for encryption is permuted according to the formula specified, using the current 56 bit block, and the one contained in that hyper-cube location as inputs. Then the S-box is selected and checked. If that S-box is already in use, then the 16 bit block used to select this location is permuted using the block contained in the location and the permutation function. The new 16 bit block is then used to select a new location to find an unused S-box. Once an unused S-box is found, it is loaded, and the selection continues for the next S-box.

If the n bit key is exactly 56 bits, the selection step is bypassed, the original S-boxes are loaded, and encryption proceeds with 16 round DES. If the key is longer, the selection algorithm comes into play, and 32 round DES is used.

Strength

The most obvious attack is 2^n with an n bit key, since most attackers will look for other methods, this is not a likely point of attack. There is another way to brute force this algorithm. That would be to try every 56 bit key, along with every possible combination of S-boxes. That would raise the complexity to 2^56* P(b,8) where b is the number of available s-boxes. This would depend highly on finding a rather large number of acceptably strong S-boxes, and more exact numbers can not be determined, however if 64 S-boxes are used (and all are assumed to be strong), the attack is about 2^105. More S-boxes would increase the strength quickly. Note therefore that if the key is less than 105 bits in length, the brute force key attack is still easier.

I-DES *could* be made immune to differential cryptanalysis though. Schneier notes that using original S-boxes 19 rounds

is sufficient to make DES immune to differential cryptanalysis [Schneier 289]. While the original S-boxes were designed to be strong against differential cryptanalysis, and changing them would make them more vulnerable. I believe that if the S-boxes were created carefully, I-DES could be made just as good (even if the s-boxes were known), as 19 round DES.

Complexity

This method pays some hefty prices in both time and space complexity. While I-DES like DES is linear in time complexity, and constant in space complexity, the constants involved in the complexity calculations can become large, however it is reasonably possible to keep the size of the hypercube and S-boxes down to within 1 Megabyte

If the hypercube were stored as a series of bit streams, it would be possible to store an entire hypercube entry in less than 128 bits:

- 56 bits for the block.
- 3 bits for each permutation function (if the permutation function were stored separately, and referenced in this manner) Six bits total
- 16 bits for the extra permutation block
- 32 bits for the S-box (an S-box can be stored in 32 bits, as each integer in an S-box is less than 16, and can therefore be represented with four bits).

This leaves 110 bits total, if we leave some extra bits on each bit stream in case the algorithm needs to be expanded, we can easily keep the size of a single hypercube entry to 128 bits, This leaves us with 16*16*16*16*128 (8388608) bits or 1048576 bytes (1024 KB). Taking into account current computing standards, 1 MB of memory is no longer considered excessive.

In time complexity, I-DES would take twice as long (plus an average case of a smaller constant) as DES for the same length plaintext, because of the doubled number of Feistel rounds, however it is remotely possible for the I-DES selection algorithm to take an excessive amount of time, and if designed improperly to enter into an infinite loop. Care should be taken in designing the hypercube to avoid this possibility.

Justifications

There is another method described in *Applied Cryptography* Second Edition, namely permuting the individual S-boxes based on extra key bits. [Schneier 300] From the Standpoint of encryption, S-box selection is quite similar to S-box permutation, except that when S-boxes are selected from a database, they can be assured strong. Schneier further points out "If the S-boxes are key dependant, and chosen by a cryptographically strong method, then linear and differential cryptanalysis are much more difficult. Remember though that randomly generated S-boxes have very poor differential and linear characteristics; even if they are secret." He then demonstrates the method by which extra key bits are used to permute the S-boxes. I believe that I-DES could be stronger, if the hypercube, and S-boxes were chosen correctly. This will however take significant effort by many cryptographers and mathematicians, before a viable cryptographic algorithm could be created.

Real world security of I-DES

While I-DES is not going to be something considered being strong enough

to encrypt ultra secret files, it would be more than enough to encrypt the basic personal data needed for an electronic commerce transfer. As Mr. Schneier has pointed out several many times, having a door lock that is harder to pick does you little good if the thief gains entry by cutting a hole in your wall with a chain saw. Adding ridiculous levels of security is not necessary for many things, one must only add enough security through encryption to make other means more profitable.

Other possible applications: It would be possible to apply I-DES, or the general principles of I-DES to several other areas:

- This could be applied to anything that uses S-boxes. One must only change the contents of the hypercube accordingly.
- Triple I-DES would work well, and should add security just as well ad triple DES, as the encrypt decrypt encrypt scheme should create the same result.

The RSA Public Key Cryptosystem

In 1976 Diffie and Hellman proposed the idea of public key cryptosystems, and in 1978 in the paper "A Method for Obtaining Digital Signatures and Public Key Cryptosystems" Rivest, Shamir, and Adleman proposed a particular implementation of the Diffie-Hellman concept. Their scheme is known as the RSA public key cryptosystem, and one of the purposes of this talk is to make the system more familiar to you.

RSA encryption and decryption: The procedures of encryption and decryption in the RSA scheme can be demonstrated. The worksheet on RSA encryption and decryption implements the following procedures.

Encryption: Convert your message to a string of integers using the correspondence A = 00, B = 01... Y = 24, Z = 25, space = 26. From right to left, break the message into six-digit chunks M_i, adding spaces at the beginning if this is necessary to. Complete the first chunk to six digits. Encrypt each chunk M_i as $E_i = M_i^{17}$ mod 8616460799, and send the sequence of E_i's. The public key is the pair of integers (e, n) = (17, 8616460799).

Decryption: After receiving the sequence E_1, E_2... each term will be decrypted by dragging out the decryption key, whose secrecy was preserved with great care until now. The decryption key is the pair of integers (d, n) = (2027358833, 8616460799), and recovering the original sequence M_1, M_2 by computing successively $P_i \equiv E_i^d$ mod n.

To see how this works, let's start by converting the message "SEND GOODS CHARGE ACCT ETC" as follows:

S E	N D	G O O	D S
26 18 04	13 03 26	06 14 14	03 18 26
C H A	R G E	A C	C T
02 07 00	17 06 04	26 00 02	02 19 26
E C T			
04 19 02			

Thus, the message to be encrypted consists of 9 chunks M_1 = 261804, M_2 = 130326... M_9 = 041902. On the Maple worksheet this appears as the sequence

M = 261804, 130326... 041902.

As stated above, I apply modular exponentiation to the 17th power to the list

to produce a new sequence E, which is the encrypted message. Note on the worksheet that taking an ordinary 17th root of a term in the sequence E does not produce the original term of plaintext. The reason for this is that we have carried out exponentiation to the 17th power modulo n rather than ordinary exponentiation.

However, I may apply modular exponentiation to the power of the decryption exponent, and I see that the sequence P coincides with the original message sequence M. Note how the encoding time compares with the time of decoding. In the next paragraphs we explain the miracle.

How it works: There are two major issues here. The first issue is how to find the decryption exponent that undoes the work of the encryption exponent modulo n. The second is how to make this system secure.

For a fixed modulus n, finding pairs of encryption and decryption exponents exploits a theorem of Euler's and the Euclidean algorithm for finding greatest common divisors of pairs of nonnegative integers. To explain Euler's theorem, we need some terminology.

Definition 1: Let $n > 1$ be a natural number. Then $\phi(n)$ is the number of elements in the set $\{a : 1 \leq a < n$ and the greatest common divisor of a and n is $1\}$.

For example, if $n = 8$, then the numbers between 1 and 8 prime to 8 are 1,3,5, and 7, so $\phi(8) = 4$. It is easy to see that if $n = p$, a prime, then $\phi(p) = p - 1$. Common implementations of the RSA cryptosystem use the fact that if $n = pq$ is a product of distinct primes, then $\phi(n) = (p - 1)(q - 1)$. There is a general formula for $\phi(n)$, but its evaluation requires knowing the prime factorization of n.

Theorem (Euler): *Let $n > 1$. If the greatest common divisor of a and n is 1, then*

$$a^{\phi(n)} \equiv 1 \bmod n.$$

If we are manufacturing keys for a fixed modulus n, then we seek positive integers e and d for which there is an integer t such that the equality

$$d(e - t)\phi(n) = 1$$

holds. If we can find such integers, then, for any integer a relatively prime to n,we have

$$(a^e)d \equiv a^{1 + t\phi(n)} \equiv a(a^{\phi(n)})^t \equiv (a)(1) \equiv a \bmod n.$$

so that (e, n) is an encryption key and (d, n) is the corresponding decryption key. When the procedure of encryption is published, some specification of what constitutes an allowable message is required to make sure that the relatively primality hypothesis of Euler's theorem is satisfied.

In the situation of equation (1), it is clear that is m is a common divisor of e and $\phi(n)$ then m is also a divisor of 1, and so the greatest common divisor of e and $\phi(n)$ is 1. On the other hand, there is a procedure for finding the greatest common divisor of a pair of integers $0 < b < a$ without factoring them both. This procedure, which goes back to Euclid, is as follows: Consider the following sequence of equalities

$a = b(q_1) + r_1$, where $0 \leq r_1 < b$,
$b = r_1(q_2) + r_2$, where $0 \leq r_2 < r_1$,
$r_1 = r_2(q_3) + r_3$, where $0 \leq r_3 < r_2$,

and so on. The sequence of remainders $r_1 > r_2 > r_3 \ldots$ is a strictly decreasing sequence of

positive integers, so the sequence eventually reaches zero and there is a last strictly positive remainder r_N

$r_{N-2} = r_{N-1}(q_N) + r_N$, Where $0 \leq r_N < r_{N-1}$.

$r_{N-1} = r_N(q_{N+1})$

Theorem (Euclid): *The last strictly positive remainder r<SUBN< sub> in the sequence of divisions listed above is the greatest common divisor of a and b. Moreover, if we rewrite all of these equations in the form*

$r_N = r_{N-2} - r_{N-1}(q_N)$

$r_{N-1} = r_{N-3} - r_{N-2}(q_{N-1})$

...

$r_1 = a - b(q_1)$

and if we substitute from top to bottom in the first equation, we obtain an equation

$ax + by = r_N$,

expressing r_N, the greatest common divisor of a and b, as a linear combination of a and b with integer coefficients.

Here is a worked out example.

Verify: The greatest common divisor of 1547 and 560 is 7. We write this sometimes as GCD(1547,560) = 7 and sometimes even more briefly as (1547,560) = 7 when it is clear from the context that we are discussing the greatest common divisor

$$1547 = 560(2) + 427$$
$$560 = 427(1) + 133$$
$$427 = 133(3) + 28$$
$$133 = 28(4) + 21$$
$$28 = 21(1) + 7$$

and 7 is obviously the last nonzero remainder. Also, rewrite the equations and put them in the reverse order, as in the statement of the theorem, and verify that successive substitutions produce the identity

$7 = 1547(21) - 560(58)$

Verify: If our RSA modulus $n = 55$, then $\phi(55) = 40$. Show also that if (3,55) is an encryption key, then (27,55) is a decryption key. Using these facts RSA encode the sequence 2,4,6,8. What happens if you try to encode the sequence 3,5,7?

The Maple worksheet describes some Maple tools you may use to create more involved examples.

Finally, let me briefly explain what makes the RSA system relatively secure. The security of the system rests on the presumed difficulty of factoring large integers, say, integers with around 200 digits that are the product of 2 primes, each of around 100 digits. A little analysis of the procedure makes it clear that the owner of the decryption key (*d, n*) must also keep $\phi(n)$ secret, or an outsider will simply run the Euclidean algorithm on the pair consisting of the encryption exponent *e* and $\phi(n)$, thus obtaining a decryption exponent *d*. Short of stealing $O(n)$ from the creator of the key pair, the only way known at present of cracking the system is to calculate $\phi(n)$ by factoring *n*.

How hard can this be? The firm RSA Laboratories has a web page of challenge numbers for people who think their algorithms and machines are good at factoring. One of their 140-digit challenge numbers was factored on a workstation, but the machine did nothing else for 10 months, I understand. I attended a lecture by an RSA Labs employee, in which he communicated the feeling that if you judiciously choose a pair of 120-digit prime numbers to create your modulus *n*, then your decryption key will probably be secure for your lifetime. Who could ask for more?

Conclusion

Cryptography is not magic. It does not remove the risk that data can be intercepted, but moves the risk to an area of a system which can be protected more easily. Cryptography only provides security when used as part of a well-designed architecture which has been designed to protect the known areas of risk. The security of private keys is of particular importance, and these are usually stored in special secure hardware modules. Standards are particularly important in cryptography because they guarantee that a system has been peer-reviewed and tested. All systems have weak points, but those of well understood public key systems are known and have been worked around. New algorithms such as AES have undergone testing by researchers and industry experts to expose any potential problems. Cryptography is one area where novelty does not necessarily equate to improvement, and users should beware of claims made on behalf of untested technologies.

References

1. Electronic frontiers - Australia http://www.eff.org/pub/Privacy/Crypto
2. http://www.math.nmsu.edu/crypto/public_html/index.html
3. Preliminary Specifications for Improved Data Encryption Standard (I-DES) - By Kenneth Cramer
4. The RSA Public Key Cryptosystem for Beginners - *Ross Staffeld* – NMSU
5. Interplanetary Internet – World of Information Technology – Feb, 2002