Why cryptography is important computer science

Cryptography is usually referred to as the study of secret, while nowadays is most attached to the definition of encryption. Encryption is the process of converting plain text “ unhidden” to a cryptic text “ hidden” to secure it against data thieves. This process has another part where cryptic text needs to be decrypted on the other end to be understood. Fig. 1 shows the simple flow of commonly used encryption algorithms.

http://www. cse. wustl. edu/~jain/cse567-06/ftp/encryption_perf/fig1. gif

Cryptographic system is “ a set of cryptographic algorithms together with the key management processes that support use of the algorithms in some application context.” This definition defines the whole mechanism that provides the necessary level of security comprised of network protocols and data encryption algorithms.

The first documented use of cryptography in writing dates back to circa 1900 B. C. when an Egyptian scribe used non-standard hieroglyphs in an inscription. Some experts argue that cryptography appeared spontaneously sometime after writing was invented, with applications ranging from diplomatic missives to war-time battle plans. It is no surprise, then, that new forms of cryptography came soon after the widespread development of computer communications. In data and telecommunications, cryptography is necessary when communicating over any untrusted medium, which includes just about any network, particularly the Internet.

Within the context of any application-to-application communication, there are some specific security requirements, including:

Authentication: The process of proving one’s identity. (The primary forms of host-to-host authentication on the Internet today are name-based or address-based, both of which are notoriously weak.)

Privacy/confidentiality: Ensuring that no one can read the message except the intended receiver.

Integrity: Assuring the receiver that the received message has not been altered in any way from the original.

Non-repudiation: A mechanism to prove that the sender really sent this message.

Cryptography, then, not only protects data from theft or alteration, but can also be used for user authentication. There are, in general, three types of cryptographic schemes typically used to accomplish these goals: secret key (or symmetric) cryptography, public-key (or asymmetric) cryptography, and hash functions, each of which is described below. In all cases, the initial unencrypted data is referred to as plaintext. It is encrypted into ciphertext, which will in turn (usually) be decrypted into usable plaintext.

Cryptography provides information Security for

Defending against external/internal hackers

Defending against industrial espionage

Securing E-commerce

Securing bank accounts/electronic transfers

Securing intellectual property

Avoiding liability

Threats to Information Security

Pervasiveness of email/networks

Online storage of sensitive information

Insecure technologies (e. g. wireless)

Trend towards paperless society

Weak legal protection of email privacy

Types of Secret WritingSteganography

Steganography comes from the Greek word meaning covered writing. Dictionary. com defines steganography as the hiding of a message within another so that the presence of the hidden message is indiscernible. The key concept behind steganography is that the message to be transmitted is not detectable to the casual eye. In fact, people who the are not intended to be the recipients of the message should not even suspect that a hidden message exists.

The difference between steganography and cryptography is that in cryptography, one can tell that a message has been encrypted, but he cannot decode the message without knowing the proper key. In steganography, the message itself may not be difficult to decode, but most people would not detect the presence of the message. When combined,

steganography and cryptography can provide two levels of security. Computer programs exist which encrypt a message using cryptography, and hide the encryption within an image using steganography.

AThe three types of algorithms:

Secret Key Cryptography (SKC): Uses a single key for both encryption and decryption

Public Key Cryptography (PKC): Uses one key for encryption and another for decryption

Hash Functions: Uses a mathematical transformation to irreversibly “ encrypt” information

http://www. garykessler. net/library/images/crypto_types. gif

Secret Key Cryptography

With secret key cryptography, a single key is used for both encryption and decryption. The sender uses the key (or some set of rules) to encrypt the plaintext and sends the ciphertext to the receiver. The receiver applies the same key (or ruleset) to decrypt the message and recover the plaintext. Because a single key is used for both functions, secret key cryptography is also called symmetric encryption.

With this form of cryptography, it is obvious that the key must be known to both the sender and the receiver; that, in fact, is the secret. The biggest difficulty with this approach, of course, is the distribution of the key.

Secret key cryptography schemes are generally categorized as being either stream ciphers or block ciphers. Stream ciphers operate on a single bit (byte or computer word) at a time and implement some form of feedback mechanism so that the key is constantly changing. A block cipher is so-called because the scheme encrypts one block of data at a time using the same key on each block. In general, the same plaintext block will always encrypt to the same ciphertext when using the same key in a block cipher whereas the same plaintext will encrypt to different ciphertext in a stream cipher.

Stream ciphers come in several flavors but two are worth mentioning here. Self-synchronizing stream ciphers calculate each bit in the keystream as a function of the previous n bits in the keystream. It is termed “ self-synchronizing” because the decryption process can stay synchronized with the encryption process merely by knowing how far into the n-bit keystream it is. One problem is error propagation; a garbled bit in transmission will result in n garbled bits at the receiving side. Synchronous stream ciphers generate the keystream in a fashion independent of the message stream but by using the same keystream generation function at sender and receiver. While stream ciphers do not propagate transmission errors, they are, by their nature, periodic so that the keystream will eventually repeat.

Block ciphers can operate in one of several modes; the following four are the most important:

Electronic Codebook (ECB) mode is the simplest, most obvious application: the secret key is used to encrypt the plaintext block to form a ciphertext block. Two identical plaintext blocks, then, will always generate the same

ciphertext block. Although this is the most common mode of block ciphers, it is susceptible to a variety of brute-force attacks.

Cipher Block Chaining (CBC) mode adds a feedback mechanism to the encryption scheme. In CBC, the plaintext is exclusively-ORed (XORed) with the previous ciphertext block prior to encryption. In this mode, two identical blocks of plaintext never encrypt to the same ciphertext.

Cipher Feedback (CFB) mode is a block cipher implementation as a self-synchronizing stream cipher. CFB mode allows data to be encrypted in units smaller than the block size, which might be useful in some applications such as encrypting interactive terminal input. If we were using 1-byte CFB mode, for example, each incoming character is placed into a shift register the same size as the block, encrypted, and the block transmitted. At the receiving side, the ciphertext is decrypted and the extra bits in the block (i. e., everything above and beyond the one byte) are discarded.

Output Feedback (OFB) mode is a block cipher implementation conceptually similar to a synchronous stream cipher. OFB prevents the same plaintext block from generating the same ciphertext block by using an internal feedback mechanism that is independent of both the plaintext and ciphertext bitstreams.

Secret key cryptography algorithms that are in use today include:

DES: (Data Encryption Standard), was the first encryption standard to be recommended by NIST (National Institute of Standards and Technology). It is based on the IBM proposed algorithm called Lucifer. DES became a standard

in 1974. Since that time, many attacks and methods recorded that exploit the weaknesses of DES, which made it an insecure block cipher.

3DES: As an enhancement of DES, the3DES (Triple DES) encryption standard was proposed. In this standard the encryption method is similar to the one in original DES but applied 3 times to increase the encryption level. But it is a known fact that 3DES is slower than other block cipher methods.

AES: (Advanced Encryption Standard), is the new encryption standard recommended by NIST to replace DES. Rijndael (pronounced Rain Doll) algorithm was selected in 1997 after a competition to select the best encryption standard. Brute force attack is the only effective attack known against it, in which the attacker tries to test all the characters combinations to unlock the encryption. Both AES and DES are block ciphers.

Blowfish: It is one of the most common public domain encryption algorithms provided by Bruce Schneier – one of the world’s leading cryptologists, and the president of Counterpane Systems, a consulting firm specializing in cryptography and computer security.

Blowfish is a variable length key, 64-bit block cipher. The Blowfish algorithm was first introduced in 1993. This algorithm can be optimized in hardware applications though it’s mostly used in software applications.

Twofish: A 128-bit block cipher using 128-, 192-, or 256-bit keys. Designed to be highly secure and highly flexible, well-suited for large microprocessors, 8-bit smart card microprocessors, and dedicated hardware. Designed by a

team led by Bruce Schneier and was one of the Round 2 algorithms in the AES process.

Public-Key Cryptography

Public-key cryptography has been said to be the most significant new development in cryptography in the last 300-400 years. Modern PKC was first described publicly by Stanford University professor Martin Hellman and graduate student Whitfield Diffie in 1976. Their paper described a two-key crypto system in which two parties could engage in a secure communication over a non-secure communications channel without having to share a secret key.

PKC depends upon the existence of so-called one-way functions, or mathematical functions that are easy to compute whereas their inverse function is relatively difficult to compute. Let me give you two simple examples:

Multiplication vs. factorization: Suppose I tell you that I have two prime numbers, 3 and 7, and that I want to calculate the product; it should take almost no time to calculate that value, which is 21. Now suppose, instead, that I tell you that I have a number, 21, and I need you tell me which pair of prime numbers I multiplied together to obtain that number. You will eventually come up with the solution but whereas calculating the product took milliseconds, factoring will take longer. The problem becomes much harder if I start with primes that have 400 digits or so, because the product will have ~800 digits.

Exponentiation vs. logarithms: Suppose I tell you that I want to take the number 3 to the 6th power; again, it is relatively easy to calculate 36A = A 729. But if I tell you that I have the number 729 and want you to tell me the two integers that I used, x and y so that logxA 729A = A y, it will take you longer to find the two values.

While the examples above are trivial, they do represent two of the functional pairs that are used with PKC; namely, the ease of multiplication and exponentiation versus the relative difficulty of factoring and calculating logarithms, respectively. The mathematical “ trick” in PKC is to find a trap door in the one-way function so that the inverse calculation becomes easy given knowledge of some item of information.

Generic PKC employs two keys that are mathematically related although knowledge of one key does not allow someone to easily determine the other key. One key is used to encrypt the plaintext and the other key is used to decrypt the ciphertext. The important point here is that it does not matter which key is applied first, but that both keys are required for the process to work. Because a pair of keys are required, this approach is also called asymmetric cryptography.

In PKC, one of the keys is designated the public key and may be advertised as widely as the owner wants. The other key is designated the private key and is never revealed to another party. It is straight forward to send messages under this scheme. Suppose Ram wants to send Bobby a message. Ram encrypts some information using Bobby’s public key; Bobby decrypts the ciphertext using his private key. This method could be also

used to prove who sent a message; Ram, for example, could encrypt some plaintext with her private key; when Bobby decrypts using Ram’s public key, he knows that Ram sent the message and Ram cannot deny having sent the message (non-repudiation).

Public key cryptography algorithms that are in use today include:

RSA: The first, and still most common, PKC implementation, named for the three MIT mathematicians who developed it – Ronald Rivest, Adi Shamir, and Leonard Adleman. RSA today is used in hundreds of software products and can be used for key exchange, digital signatures, or encryption of small blocks of data. RSA uses a variable size encryption block and a variable size key. The key-pair is derived from a very large number, n, that is the product of two prime numbers chosen according to special rules; these primes may be 100 or more digits in length each, yielding an n with roughly twice as many digits as the prime factors. The public key information includes n and a derivative of one of the factors of n; an attacker cannot determine the prime factors of n (and, therefore, the private key) from this information alone and that is what makes the RSA algorithm so secure. (Some descriptions of PKC erroneously state that RSA’s safety is due to the difficulty in factoring large prime numbers. In fact, large prime numbers, like small prime numbers, only have two factors!) The ability for computers to factor large numbers, and therefore attack schemes such as RSA, is rapidly improving and systems today can find the prime factors of numbers with more than 200 digits.

Diffie-Hellman: After the RSA algorithm was published, Diffie and Hellman came up with their own algorithm. D-H is used for secret-key key exchange only, and not for authentication or digital signatures.

Digital Signature Algorithm (DSA): The algorithm specified in NIST’s Digital Signature Standard (DSS), provides digital signature capability for the authentication of messages.

ElGamal: Designed by Taher Elgamal, a PKC system similar to Diffie-Hellman and used for key exchange.

Elliptic Curve Cryptography (ECC): A PKC algorithm based upon elliptic curves. ECC can offer levels of security with small keys comparable to RSA and other PKC methods. It was designed for devices with limited compute power and/or memory, such as smartcards and PDAs.

Hash Functions

Hash functions, also called message digests and one-way encryption, are algorithms that, in some sense, use no key (Figure 1C). Instead, a fixed-length hash value is computed based upon the plaintext that makes it impossible for either the contents or length of the plaintext to be recovered. Hash algorithms are typically used to provide a digital fingerprint of a file’s contents, often used to ensure that the file has not been altered by an intruder or virus. Hash functions are also commonly employed by many operating systems to encrypt passwords. Hash functions, then, provide a measure of the integrity of a file.

Hash algorithms that are in common use today include:

Message Digest (MD) algorithms: A series of byte-oriented algorithms that produce a 128-bit hash value from an arbitrary-length message.

MD2 : Designed for systems with limited memory, such as smart cards.

MD4 : Developed by Rivest, similar to MD2 but designed specifically for fast processing in software.

MD5 : Also developed by Rivest after potential weaknesses were reported in MD4; this scheme is similar to MD4 but is slower because more manipulation is made to the original data. MD5 has been implemented in a large number of products although several weaknesses in the algorithm were demonstrated by German cryptographer Hans Dobbertin in 1996 (“ Cryptanalysis of MD5 Compress”).

Secure Hash Algorithm (SHA): Algorithm for NIST’s Secure Hash Standard (SHS). SHA-1 produces a 160-bit hash value and was originally published as FIPS 180-1 and (aka SHA-2) describes five algorithms in the SHS: SHA-1 plus SHA-224, SHA-256, SHA-384, and SHA-512 which can produce hash values that are 224, 256, 384, or 512 bits in length, respectively.

RIPEMD: A series of message digests that initially came from the RIPE (RACE Integrity Primitives Evaluation) project. RIPEMD-160 was designed by Hans Dobbertin, Antoon Bosselaers, and Bart Preneel, and optimized for 32-bit processors to replace the then-current 128-bit hash functions. Other versions include RIPEMD-256, RIPEMD-320, and RIPEMD-128.

HAVAL (HAsh of VAriable Length): Designed by Y. Zheng, J. Pieprzyk and J. Seberry, a hash algorithm with many levels of security. HAVAL can create hash values that are 128, 160, 192, 224, or 256 bits in length.

Whirlpool: A relatively new hash function, designed by V. Rijmen and P. S. L. M. Barreto. Whirlpool operates on messages less than 2256 bits in length, and produces a message digest of 512 bits. The design of this has function is very different than that of MD5 and SHA-1, making it immune to the same attacks as on those hashes (see below).

Tiger: Designed by Ross Anderson and Eli Biham, Tiger is designed to be secure, run efficiently on 64-bit processors, and easily replace MD4, MD5, SHA and SHA-1 in other applications. Tiger/192 produces a 192-bit output and is compatible with 64-bit architectures; Tiger/128 and Tiger/160 produce a hash of length 128 and 160 bits, respectively, to provide compatibility with the other hash functions mentioned above.

Why Three Encryption Techniques?

So, why are there so many different types of cryptographic schemes? Why can’t we do everything we need with just one?

The answer is that each scheme is optimized for some specific application(s). Hash functions, for example, are well-suited for ensuring data integrity because any change made to the contents of a message will result in the receiver calculating a different hash value than the one placed in the transmission by the sender. Since it is highly unlikely that two different

messages will yield the same hash value, data integrity is ensured to a high degree of confidence.

Secret key cryptography, on the other hand, is ideally suited to encrypting messages, thus providing privacy and confidentiality. The sender can generate a session key on a per-message basis to encrypt the message; the receiver, of course, needs the same session key to decrypt the message.

Key exchange, of course, is a key application of public-key cryptography (no pun intended). Asymmetric schemes can also be used for non-repudiation and user authentication; if the receiver can obtain the session key encrypted with the sender’s private key, then only this sender could have sent the message. Public-key cryptography could, theoretically, also be used to encrypt messages although this is rarely done because secret-key cryptography operates about 1000 times faster than public-key cryptography.

http://www. garykessler. net/library/images/crypto_3ways. gif

FIGURE : Sample application of the three cryptographic techniques for secure communication.

Figure puts all of this together and shows how a hybrid cryptographic scheme combines all of these functions to form a secure transmission comprising digital signature and digital envelope. In this example, the sender of the message is Ram and the receiver is by Bobby.

A digital envelope comprises an encrypted message and an encrypted session key. Ram uses secret key cryptography to encrypt her message

using the session key, which she generates at random with each session. Ram then encrypts the session key using Bobby’s public key. The encrypted message and encrypted session key together form the digital envelope. Upon receipt, Bobby recovers the session secret key using his private key and then decrypts the encrypted message.

The digital signature is formed in two steps. First, Ram computes the hash value of her message; next, she encrypts the hash value with her private key. Upon receipt of the digital signature, Bobby recovers the hash value calculated by Ram by decrypting the digital signature with Ram’s public key. Bobby can then apply the hash function to Ram’s original message, which he has already decrypted (see previous paragraph). If the resultant hash value is not the same as the value supplied by Ram, then Bobby knows that the message has been altered; if the hash values are the same, Bobby should believe that the message he received is identical to the one that Ram sent.

This scheme also provides nonrepudiation since it proves that Ram sent the message; if the hash value recovered by Bobby using Ram’s public key proves that the message has not been altered, then only Ram could have created the digital signature. Bobby also has proof that he is the intended receiver; if he can correctly decrypt the message, then he must have correctly decrypted the session key meaning that his is the correct private key.

The Significance of Key Length

In a article in the industry literature (circa 9/98), a writer made the claim that 56-bit keys do not provide as sufficient protection for DES today as they did

in 1975 because computers are 1000 times faster today than in 1975. Therefore, the writer went on, we should be using 56, 000-bit keys today instead of 56-bit keys to provide adequate protection. The conclusion was then drawn that because 56, 000-bit keys are infeasible (true), we should accept the fact that we have to live with weak cryptography (false!). The major error heris that the writer did not take into account that the number of possible key values double whenever a single bit is added to the key length; thus, a 57-bit key has twice as many values as a 56-bit key (because 257 is two times 256). In fact, a 66-bit key would have 1024 times the possible values as a 56-bit key.

But this does bring up the issue, what is the precise significance of key length as it affects the level of protection?

In cryptography, size does matter. The larger the key, the harder it is to crack a block of encrypted data. The reason that large keys offer more protection is almost obvious; computers have made it easier to attack ciphertext by using brute force methods rather than by attacking the mathematics (which are generally well-known anyway). With a brute force attack, the attacker merely generates every possible key and applies it to the ciphertext. Any resulting plaintext that makes sense offers a candidate for a legitimate key. This was the basis, of course, of the EFF’s attack on DES..

Public Key Certificates and Certificate Authorities

Certificates and Certificate Authorities (CA) are necessary for widespread use of cryptography for e-commerce applications. While a combination of secret

and public key cryptography can solve the business issues discussed above, crypto cannot alone address the trust issues that must exist between a customer and vendor in the very fluid, very dynamic e-commerce relationship. How, for example, does one site obtain another party’s public key? How does a recipient determine if a public key really belongs to the sender? How does the recipient know that the sender is using their public key for a legitimate purpose for which they are authorized? When does a public key expire? How can a key be revoked in case of compromise or loss?

The basic concept of a certificate is one that is familiar to all of us. A driver’s license, credit card, or SCUBA certification, for example, identify us to others, indicate something that we are authorized to do, have an expiration date, and identify the authority that granted the certificate.

As complicated as this may sound, it really isn’t! Consider driver’s licenses. I have one issued by the State of Vermont. The license establishes my identity, indicates the type of vehicles that I can operate and the fact that I must wear corrective lenses while doing so, identifies the issuing authority, and notes that I am an organ donor. When I drive outside of Vermont, the other jurisdictions throughout the U. S. recognize the authority of Vermont to issue this “ certificate” and they trust the information it contains. Now, when I leave the U. S., everything changes. When I am in Canada and many other countries, they will accept not the Vermont license, per se, but any license issued in the U. S.; some other countries may not recognize the Vermont driver’s license as sufficient bona fides that I can drive. This analogy represents the certificate chain, where even certificates carry certificates.

For purposes of electronic transactions, certificates are digital documents. The specific functions of the certificate include:

Establish identity: Associate, or bind, a public key to an individual, organization, corporate position, or other entity.

Assign authority: Establish what actions the holder may or may not take based upon this certificate.

Secure confidential information (e. g., encrypting the session’s symmetric key for data confidentiality).

Typically, a certificate contains a public key, a name, an expiration date, the name of the authority that issued the certificate, a serial number, any pertinent policies describing how the certificate was issued and/or how the certificate may be used, the digital signature of the certificate issuer, and perhaps other information.

http://www. garykessler. net/library/images/crypto_cert. gif

A sample abbreviated certificate is shown in Figure. This is a typical certificate found in a browser; while this one is issued by GTE Cybertrust, many so-called root-level certificates can be found shipped with browsers. When the browser makes a connection to a secure Web site, the Web server sends its public key certificate to the browser. The browser then checks the certificate’s signature against the public key that it has stored; if there is a match, the certificate is taken as valid and the Web site verified by this certificate is considered to be “ trusted.”