Class Notes for Lecture 2

System Security

Security Concerns: Information flow from source to destination without any third-party interference is the ideal flow. Third-party interference can be of four kinds –

Type of interference	Security Concern	Remarks
Interruption	Availability	The information channel between source and destination is interrupted and information is made unavailable to the destination.
Interception	Confidentiality	The information channel is intercepted and information is made available to an illegitimate party. There are two ways of achieving confidentiality – • By making information inaccessible • By making information accessible but unintelligible. For ex. Encrypted data
Modification	Authenticity	The information channel is interrupted; the information is modified by an illegitimate party and sent to the destination. There are two kinds of security concerns here • Authenticity of the information: Has the information been sent by a legitimate sender? • Integrity of the information: Has the information been modified?
Fabrication	Non-repudiability	The sender of a message should not be able to disown having sent the message. There are many kinds of fabrication • Receiver fabricating the message : The receiver fabricates the message as sent by a legitimate sender • Sender disowning having sent the message • Third-party sending message to receiver as a legitimate sender In all the above cases, non-repudiability is compromised.

How to achieve security: At the high level, the basic idea is to keep the adversaries (contending parties) separated. There are many ways to do this –

- Physical separation: Keep the adversaries physically separated.
- Temporal separation: Time slicing achieves temporal separation. For example, in an operating system two processes contending for CPU are temporally separated.
- Cryptographic separation: In cryptographic separation, the information is made available but is unintelligible and, therefore, of no use. In a sense, cryptography is used to logically separate legitimate parties from illegitimate ones.
- Logical separation: Access control techniques are a way to achieve logical separation. Even though the

resources are all at the same physical location, by defining access rules for the resources logical separation keeps resources safe from illegitimate parties.

Cryptography: Encode data in such a way that it is accessible only to legitimate parties. If we think of encryption as the process of locking data inside a safe, then the encryption algorithm is the lock and the encryption key is the key to the lock. Only those who have the key can open the lock.

There are two ways to make sure that only legitimate parties have access to the data –

- Keep the encryption algorithm secret (rely on secrecy of algorithm) : This is not a good idea for a number of reasons
    - It is very hard to develop a good encryption algorithm. That is why it is better to rely on well-known algorithms.
    - Keeping an algorithm secret cannot scale beyond a few users.
    - Security by obscurity is not a good idea. The basic assumption under security by obscurity is that by hiding the design and implementation of a security system, the flaws of the system will be hidden from the attackers and hence unlikely to be attacked. However, experience has shown that the same kind of flaws tend to exist in encryption algorithm designs, especially those that have been custom-crafted and hence has not been analyzed as well as previously published algorithms.
- Keep the encryption key secret: This is a better way to go. A well-known publicly available encryption algorithm is likely to have most flaws removed. Adversaries are less likely to discover flaws in the algorithms and hence it is more secure. Another reason is the reuse the encryption algorithms. One can provide the same level of security by simply changing the encryption key.

Key concepts and terminology of cryptography:

- Plaintext: Unencrypted data
- Ciphertext: Encrypted data
- Encryption Vs Decryption: Encryption and decryption are inverses of each other. Encryption converts plaintext into ciphertext while decryption converts ciphertext into plaintext. In most cases, the process of encryption and decryption are the same or very similar.
- Key Vs Algorithm
- Cryptanalysis: Breaking a security system by performing analysis on information about the system (encryption algorithm, plaintext-ciphertext dictionary etc.) to obtain the meaning of encrypted data without access to the secret key.

Types of attacks:

Type of attack	**Information known to cryptanalyst**	**Examples**
Ciphertext only	• Encryption algorithm • Ciphertext to be decoded	Tap on a wireless network, take down all the encrypted data and use the algorithm and this data to perform analysis.
Known plaintext	• Encryption algorithm	Some messages are frequently exchanged

	• Ciphertext to be decoded • One or more plaintext-ciphertext pairs formed with the key	in any communication protocol (e.g., Hello message). It is easy to decipher what these messages are using some knowledge of the system and the protocol. These messages and their ciphertexts act as a dictionary for further analysis.
Chosen plaintext	• Encryption algorithm • Ciphertext to be decoded • Chosen plaintext and its corresponding ciphertext generated by the secret key	Challenge-Response protocol between two legitimate parties. One of the parties acts like an oracle which will send ciphertext for a given plaintext. The attacker can use the oracle to form an exhaustive dictionary of plaintext-ciphertext. This dictionary along with the encryption algorithm can be used to launch an attack.
Chosen ciphertext	• Encryption algorithm • Ciphertext to be decoded • Chosen ciphertext and its corresponding plaintext generated by the secret key	This is the opposite of "Chosen plaintext" attack. In this case, the oracle decrypts the ciphertext (chosen by the attacker) and returns the corresponding plaintext.

Steganography: While cryptography depends on hiding the content of the information, steganography depends on hiding the presence of information. The usual technique is to conceal secret data inside normal looking messages/pictures. Steganography is most-commonly used in terrorist or criminal activities where information needs to be communicated via communication channels which are being monitored for "suspicious" activity. It is also used for copyright protection using a ***watermark*** (invisible data encoded in messages that is retained in copies, and is robust in the face of typical image transformation operations).

There are multiple reasons for using steganographic techniques –

- As we have discussed earlier, one is to conceal secret data if communication is monitored for suspicious activity

- Steganographic techniques provide authenticity without the signature being overtly visible. (Visibility is not a cosmetic issue: an attacker will identify any overtly obvious mark; with some effort, it is quite likely that this mark can then be removed. From this point on, the attacker can make copies freely, without danger of prosecution.)

Symmetric Cryptography: It consists of an encryption algorithm, a decryption algorithm (which is, essentially, reverse of the encryption algorithm) and a secret key that is shared between the sender and the receiver. The sender encrypts the plaintext message using the encryption algorithm and the secret key and transmits the ciphertext to the receiver (via insecure channel). The receiver uses his/her secret key to then decrypt the message into plaintext.

It is equivalent to the message being put inside a safe for which both sender and receiver have the key. The same key works for both encryption and decryption. Logically, it is nothing but a "secure channel" (like the safe).

Stream and block ciphers:

- Stream cipher: A stream cipher is used to encrypt digital streams of data, one bit or byte at a time. For example, when a password is sent using a stream cipher, one may want to send each character as it is typed. This requires us to encrypt one byte at a time, rather than waiting for many characters that can then form a block.

- Block cipher: Most encryption mechanisms are block ciphers. Data is partitioned into blocks (typically 64 or 128 bits) and encryption is done on a per block basis.
    - Larger block sizes are preferred over smaller block sizes to deter statistical analysis. For instance, if we used a block size of 8 bits, it becomes easy to analyze the frequency distribution of each 8-bit character in ciphertext. If we used a simple codebook that replaces each plaintext byte with its corresponding cipher text, then this frequency can be used to break encryption easily. For instance, you might say that the most frequently occurring ciphertext byte corresponds to "e" in English, the next most frequently occurring ciphertext character must be "t" and so on. Note that this kind of statistical analysis can be used for encryption without even figuring out the key --- in other words, even a very large key does not help make the technique secure, when you have small block sizes.

      Statistical analysis is harder with larger block sizes, e.g., with 128-bit keys, you will need a very large table to keep track of frequency distributions.

- Block ciphers are used to generate stream ciphers. There are techniques to generate stream ciphers without introducing the types of attacks described above that exploit small block sizes.

Structure of Symmetric Cryptography:

Good ciphers are based on Shannon's concepts of "diffusion" and "confusion". Diffusion is the mechanism to disperse bit-patterns within each block of data. Confusion is to mix-up the order of bits in a block. This is done so that the encrypted data doesn't have a direct correspondence to input. Using statistical analysis to break an encryption algorithm becomes difficult once diffusion/confusion are used because the encrypted data does not preserve the statistical information of the plaintext.

Symmetric Crypto Algorithms:

- DES – This algorithm is not considered secure any more. The key used by the algorithm is 56 bits long, with 64-bit block size. Also, the algorithm was designed to work on hardware. It is slow when implemented in software. Security can be improved by using triple DES with two keys (112 bits), but speed is even worse.

- AES (128 bits) – This algorithm uses some elegant mathematics and is designed for efficient software implementation.

Public key (Asymmetric) Cryptography:

Drawbacks with Symmetric key cryptography

- Symmetric key cryptography is not scalable. There should be a key for each pair of sender and receiver. This may work for small groups but becomes infeasible as the group size increases.

- Key distribution is not easy. There is no easy and secure way for the sender and receiver to exchange the secret key. (The confidentiality of secret keys needs to be preserved during key distribution: this is harder to achieve than integrity, which is all that is required in the case of public keys.)

- Can't easily be used for digital signatures.

Public key cryptography uses two keys, one for encryption and the other for decryption. It should be computationally infeasible to compute one given the other. Every "principal" generates a pair of keys – a private key and a public key. The private key is private to the principal and need not be revealed to anyone while the public key can be freely distributed to anyone.

Public key cryptography depends on the secrecy of the private key and the integrity of the public key.

Encryption in Public Key Cryptography: Alice has to send a message to Bob and wants to make sure that only Bob can read the message. Alice uses Bob's public key to encrypt the message and transmits the ciphertext to Bob. Since only Bob has access to Bob's private key, only he will be able to decrypt the message to plaintext. (See picture from the slides.)

This ensures ***confidentiality*** of the message.

Encrypt (plaintext input, receiver's public key) ---- Transmit-- → Decrypt(ciphertext input, receiver's private key)

Authentication in Public Key Cryptography: Alice has to send a message to Bob and Bob needs to verify that the message has, in fact, been sent by Alice. Alice uses Alice's private key to encrypt the message and transmits the ciphertext to Bob. Bob uses Alice's public key to decrypt the message and authenticates Alice (since, only Alice has access to her private key and only she could have encrypted the message)

One of the problems that need to be tackled by authentication protocols is the possibility of replay attacks. Even without cracking the key, an attacker can replay an encrypted message to try to authenticate himself as Alice. To avoid this, the message from Alice should include additional information such as a timestamp. The protocol will be even more robust if Alice has to encrypt a piece of information chosen by Bob, rather than encrypt a known quantity such as timestamp that may be present in other messages from Alice. Such protocols are referred to as challenge-response protocols, where the principal that is attempting to authenticate itself is required to respond to a challenge from the other end. To prevent replay attacks, the challenge should be unpredictable and never be reused --- such a quantity is called a nonce (no more than once).

This ensures ***authenticity*** of the message.

Encrypt (plaintext input, sender's private key) ---- Transmit-- → Decrypt(ciphertext input, sender's public key)

Encryption Vs Signing:

- Encryption: This operation is done to ensure confidentiality. The encoding is done using receiver's public key, so the ciphertext can be decrypted only by that person's private key.
- Signing: This is done to ensure authenticity. The encoding is done using sender's private key and the ciphertext can be decrypted by everyone, since all of them may have the sender's public key. One can be sure that the message came only from the person whose public key is used for decoding.

Issues with RSA Algorithm: The strength of the RSA algorithm relies on the fact that large numbers are difficult/ take long time to factorize. Hence, the algorithm depends on large prime numbers (e.g. 512/1024 bits) to produce a hard-to-factorize large number. Conventional techniques are inefficient to generate/test for prime numbers. Miller-Rabin test uses probabilistic tests to generate large prime numbers probabilistically. It performs a series of tests and probability that the number is non-prime goes down exponentially with every test. The number is picked if the confidence is high enough.

Random Numbers: Random numbers are used at a variety of techniques like session key generation, RSA key generation, nonces etc. It is very essential to have cryptographically strong random numbers which are unpredictable. Pseudo random numbers are not good enough because they are predictable, given the number generation algorithms and the seed. Most popular random number generators use natural randomness in real world, for example, key stroke intervals, network packet arrival characteristics etc. Arbitrary large random numbers can be generated by concatenating a series of random bits.

Conventional Vs Public Key Cryptography:

Feature	Conventional Cryptography	Public Key Cryptography
Speed of execution	Conventional cryptography is fast. Software implementation on current PCs can perform encryption at the rate of few MBps.	Public key cryptography is much slower. About 3 orders of magnitude slower.
Key distribution	Key distribution is difficult. There is no easy and secure way by which sender and receiver can exchange the secret key, since confidentiality of keys cannot be compromised.	Key distribution is easier because the public keys can be freely distributed. Integrity of public keys should not be compromised but confidentiality is not needed.
Non-repudiability	Since the sender and receiver share the same key, non-repudiability is compromised.	Non-repudiability is preserved.

Solution:

Use conventional cryptography to encrypt bulk data.

Generate a symmetric key (session key) for the encryption of data using a cryptographically strong random number generator, and then send the key to the receiver (i.e., small amount of data) by encrypting it using receiver' public key (assuming the integrity of the receiver' public key).