

CS378: Information Assurance and Security Cryptography

Dr. Bill Young
Department of Computer Sciences
University of Texas at Austin

Last updated: February 10, 2012 at 08:26

This is not a course in cryptography. The department offers one and you are advised to take it, if you plan to work in the security field.

Our point here will be to give some intuitions about:

- what are the key concepts of cryptography;
- how is it used as a tool for IA;
- how effective is it in that regard.

Introduction to Cryptography

What is the Goal?

Crypto is a key ingredient in any successful information assurance program. *But it's important to understand that cryptography is not the goal; it's a tool toward reaching one or more of several possible goals.*

If you think cryptography will solve your problem, then you don't understand cryptography ... and you don't understand your problem. –Bruce Schneier

Why would Schneier say such a thing? Do you agree?

To what IA goals is cryptography likely to contribute?

To what IA goals is cryptography likely to contribute?

Confidentiality: the information in a message is accessible only to authorized parties

Authentication: the receiver of a message can ascertain its origin.

Integrity: modification to a message can be detected

Non-repudiation: a sender cannot falsely deny originating the message

Note that we've stated these goals in terms of data in transit; they also apply to data at rest. [Explain this remark.](#)

The value of an encryption algorithm depends on the use. An algorithm that is appropriate for computers might not be appropriate for use by an agent in the field.

Shannon (1949) listed characteristics of good ciphers.

- ① The degree of secrecy required should determine the effort expending in encryption / decryption.
- ② The keys and algorithm should be free from complexity.
- ③ The implementation should be as simple as possible.
- ④ Errors in encryption should not propagate.
- ⑤ The size of the ciphertext shouldn't be more than the size of the plaintext.

[Which of these still apply to modern cryptography?](#)

Some of Shannon's criteria don't really apply to modern computer-based cryptography.

The following are suggested as other tests of worth for current cryptographic practice:

- is based on sound mathematics;
- has been analyzed by competent experts and found to be sound;
- has stood the test of time.

There are two basic types of encryption:

symmetric algorithms (also called "secret key") use a single key for both encryption and decryption;

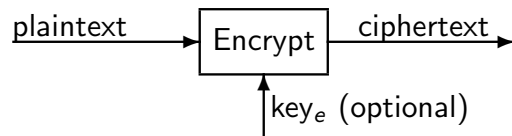
asymmetric algorithms (also called "public key") use different keys for encryption and decryption.

For any encryption approach, there are two major challenges:

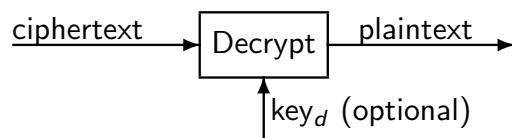
Key distribution: how do we convey keys to those who need them to establish secure communication.

Key management: given a large number of keys, how do we preserve their safety and make them available as needed.

The purpose of encryption is to render the message less useful / meaningful to the intruder. Conceptually, the process of encryption is quite simple:



as is the process of decryption:



Do these pictures apply equally well to symmetric and asymmetric algorithms?

Be sure that you understand the following terms:

- encryption, decryption
- key, keyspace
- plaintext, ciphertext
- cryptography, cryptanalysis, cryptology

What is the relationship between the key size and the keyspace?
Does reference to keyspace always make sense?

Cryptanalysis

Attacks on an encryption algorithm can be classified according to what information is available to the attacker.

Ciphertext-only attack: available is only the ciphertext of several messages encrypted with the same key/algorithm

Known plaintext: available is a quantity of ciphertext and corresponding plaintext.

Chosen plaintext attack: cryptanalyst can control the plain text to be encrypted and see the resulting ciphertext.

Adaptive chosen plaintext attack: chosen plaintext attack where the choice of plaintext may depend on the ciphertext from earlier attempts.

Cryptanalysis

Chosen ciphertext attack: the attacker selects a ciphertext and is given the corresponding plaintext. E.g., attacker gains access to the decryption device but not the key.

Chosen key attack: cryptanalyst has knowledge of the relationship among different keys.

Rubber-hose cryptanalysis: breaking cipher through threats, blackmail, or torture.

Recall *the Principle of Easiest Penetration*. Often it is more effective to attack the human users rather than the cryptographic algorithms. Many successful attacks succeed because the users are hurried, lazy, careless, naive or uninformed. Sometimes users can be bribed or coerced.

Excellent paper: “Why Cryptosystems Fail” by Ross Anderson, available on-line.

It turns out that the threat model commonly used by cryptosystem designers was wrong: most frauds were not caused by cryptanalysis or other technical attacks, but by implementation errors and management failures. This suggests that a paradigm shift is overdue in computer security. (from the abstract)

Also called “secret key” ciphers, require that sender and receiver have access to a single key.

The main issue with symmetric encryption is *key distribution*. Presents a “chicken and egg” problem: if I have a secure channel to get the key to you, why don’t I just use that channel to send the message? *How would you answer that question?*

Some symmetric encryption algorithms: AES, Blowfish, DES, 3DES, Serpent, Twofish, CAST-128, IDEA, Skipjack.

Diffie-Hellman

Recall that symmetric encryption requires that the parties agree on a secret key. *How do you distribute the key if you don’t have a secure channel? Is it even possible?*

Diffie-Hellman

Recall that symmetric encryption requires that the parties agree on a secret key. *How do you distribute the key if you don’t have a secure channel? Is it even possible?*

Diffie-Hellman (1976) is the first viable approach for key exchange over an insecure channel.

Steps in the algorithm:

- ① Alice and Bob agree on a prime number p and a base g .
- ② Alice chooses a secret number a , and sends Bob $(g^a \bmod p)$.
- ③ Bob chooses a secret number b , and sends Alice $(g^b \bmod p)$.
- ④ Alice computes $((g^b \bmod p)^a \bmod p)$.
- ⑤ Bob computes $((g^a \bmod p)^b \bmod p)$.

Both Alice and Bob can use this number as their key.

By itself, this protocol is vulnerable to a man-in-the-middle attack. *What is a man-in-the-middle attack? Describe this attack?*

The Diffie-Hellman key exchange is vulnerable to a man-in-the-middle attack. In this attack, an opponent Carol intercepts Alice's public value and sends her own public value to Bob. When Bob transmits his public value, Carol substitutes it with her own and sends it to Alice. Carol and Alice thus agree on one shared key and Carol and Bob agree on another shared key. After this exchange, Carol simply decrypts any messages sent out by Alice or Bob, and then reads and possibly modifies them before re-encrypting with the appropriate key and transmitting them to the other party. This vulnerability is present because Diffie-Hellman key exchange does not authenticate the participants. Possible solutions include the use of digital signatures and other protocol variants.

–RSA labs

Another important distinction in symmetric algorithms is between stream and block ciphers.

Stream ciphers convert one bit/symbol of plaintext directly into a symbol of ciphertext.

Block ciphers encrypts a group of plaintext bits/symbols as one block.

Most modern symmetric encryption algorithms are block ciphers. The distinction is not as clear cut as you might think at first. **Why is that?**

Public Key Encryption

In a symmetric algorithm, the key must remain secret for the algorithm to be secure. In 1976, Diffie and Hellman proposed **public key encryption** (asymmetric encryption) in which a key does not have to be kept secret.

In 1997, it was publicly disclosed that asymmetric key algorithms were developed by James H. Ellis, Clifford Cocks, and Malcolm Williamson at GCHQ in the UK in the early 1970s. The researchers independently developed Diffie-Hellman key exchange and a special case of RSA. The GCHQ cryptographers referred to the technique as **non-secret encryption**.

Public Key Encryption

The idea is to use a publicly disclosed key to encrypt and a secret key to decrypt. This drastically reduces the number of keys that have to be protected. **Explain this assertion.**

The requisite relationship is:

$$P = D(K_{priv}, E(K_{pub}, P).)$$

Also, for *some* public key systems, RSA in particular, E and D commute and either key can be used in either function. That is:

$$P = D(K_{priv}, E(K_{pub}, P)) = D(K_{pub}, E(K_{priv}, P))$$

This is crucial in some uses of RSA. **But it's not true in general for public key cryptosystems.** I don't actually know if it's true for *any* others.

Explain why public key systems largely solve the key distribution problem.

Particularly with public key systems, a different notation is common. The public key for subject A is denoted K_A and the corresponding private key is K_A^{-1} .

Encryption or decryption of a message M with key K is denoted as $\{M\}_K$. Thus, what we previously wrote as:

$$P = D(K_{pub}, E(K_{priv}, P))$$

would be denoted

$$P = \{\{\{P\}_{K_A^{-1}}\}_{K_A}$$

assuming that the keys were associated with subject A .

In a *symmetric* encryption system, sending a message provides confidentiality and authentication (sort of). [Explain](#).

What guarantees do you get sending a message in an *asymmetric* encryption system?

In a *symmetric* encryption system, sending a message provides confidentiality and authentication (sort of). [Explain](#).

What guarantees do you get sending a message in an *asymmetric* encryption system?

- **Confidentiality:** if S sends to R the message $\{M\}_{K_R}$, then no one but R can recover M .
- **Authenticity:** if S sends to R the message $\{M\}_{K_S^{-1}}$, then no one but R knows the message must have come from S . [Why?](#)

Can you get both confidentiality and authentication in one message? [Explain why or why not.](#)

The basis of any public key system is the identification of a function that is easily computed, but difficult to invert without additional information. This is called a *one-way* function.

For example, it is straightforward to multiply two large primes p_1 and p_2 together. However, given the result $p_1 p_2$, it can be very difficult to factor it to recover the p_1 and p_2 .

However, given $p_1 p_2$ and either of p_1 or p_2 , it is again straightforward to recover the other, simply by dividing.

Thus, multiplication is a one-way function, because computing the inverse is effectively infeasible for products of large primes.

The **Rivest-Shamir-Adelman (RSA)** algorithm is a public key system which relies on the difficulty of factoring large numbers.

Two keys, d and e , are used for encryption and decryption. Because the use of the keys in RSA is symmetric, either can be the private key. The algorithm is such that:

$$\{\{P\}_d\}_e = P = \{\{P\}_e\}_d.$$

A plaintext block P is encrypted as $(P^e \bmod n)$. The decrypting key is chosen so that: $(P^e)^d \bmod n = P$.

Because the encrypting exponentiation is performed modulo n , an interceptor would have to factor P^e to recover the plaintext. This is difficult. However, the legitimate receiver knows d and merely computes $(P^e)^d \bmod n = P$. This is much easier.

A public key system can be based on any one-way function. A rich source of these is the set of NP-complete problems. These are infeasible to solve (requiring at least exponential time) but a proposed solution can be checked in polynomial time.

Merkle and Hellman proposed a public key system based on the **knapsack problem**: given a set of integers and a target sum, find a subset of the integers that sum to the target.

The algorithm is theoretically very secure, but has practical weaknesses. Just because a problem is difficult, in general, doesn't mean it is difficult in specific instances. [Explain](#).

Efficiency of Encryption

If public key systems have real benefits, why not just scrap symmetric encryption?

Efficiency of Encryption

If public key systems have real benefits, why not just scrap symmetric encryption?

Public key systems simplify key distribution but are typically very slow. Thus, they are used only for specialized purposes.

A public key encryption *may take 10,000 times as long* to perform as a symmetric encryption, because it depends on multiplication and division, rather than simple bit operations.

For this reason, symmetric encryption is the work horse of commercial cryptography. Still, asymmetric encryption plays some important functions.

Some commercially important systems use a combination of symmetric and asymmetric encryption, utilizing the best features of each.

Pretty Good Privacy (PGP): email encryption system.

Secure Socket Layer (SSL): add-on to the HTTP standard to allow secure network communication

For example, to send confidentially a message M in PGP, you do the following:

$$S \rightarrow R : \{K\}_{K_R}, \{M\}_K$$

where K is a new session key.

To send an authenticated message, use:

$$S \rightarrow R : \{h(M)\}_{K_S^{-1}}, M$$

What is a digital signature? What is it for? What characteristics would you want it to have?

What is a digital signature? What is it for? What characteristics would you want it to have?

Any signature scheme should have the following characteristics:

- authentic:** convinces the recipient that the signer deliberately signed the document
- unforgeable:** no other party could have produced the signature
- not reusable:** signature is bound to the document and cannot be transferred to another document
- unalterable:** after signing, the signature cannot be modified
- cannot be repudiated:** the signer cannot later deny having signed the document.

The signature is typically a function $S(P, M)$ of the signer's identity P and the message M .

What mechanism do we already have that provides some of these capabilities?

What mechanism do we already have that provides some of these capabilities?

Public key cryptosystems provide a convenient framework for digital signing of documents because *the underlying assumption is that only A has access to A's private key*. Why is that assumption crucial?

If S wishes to send a signed, confidential message to R , then S can send:

$$\{\{M\}_{K_S^{-1}}\}_{K_R}$$

Upon receipt, R knows that the message came from S and that its confidentiality was protected. Explain how. Could we have done the two encryptions in the opposite order? Why or why not?

For a public key system to work, there must be a way for a party to get the public keys of others in the system. How might that work?

- You could get the key directly from each party.
- You could get the key from a trusted key server/central repository.
- You could obtain the key from a trusted third party willing to “vouch for” the association.

What are the pros and cons of each of these?

Given the public key K_R for some entity R , how do you know for sure that the key is actually R 's public key?

This question is at the very heart of any public key infrastructure (PKI). Because if these *bindings* are not accurate, then trust breaks down in the system. Why?

Certification addresses the need for *trust* in computer systems. How do two entities that are mutually suspicious establish a relationship of trust. One way is to rely on a third party who “vouches for” the trustworthiness of one or both of the parties.

We may believe a party's affiliation or ask for independent validation. In general, we have a *trust threshold*, a degree of trust we're willing to confer without going further in the certification process. This threshold may depend on the size or nature of the transaction.

The police, Chamber of Commerce, Better Business Bureau, and credit reporting agencies all function in part as certification authorities for such transactions in the “real” world.

Electronically, certification is accomplished with digital signatures and hash functions.

A public key and user's identity are bound together in a *certificate*. This is then signed by a *certification authority (CA)*, vouching for the accuracy of the binding.

The following are possible steps. Suppose X is the president of the company and her immediate subordinate is Y . Each have a public key pair.

- ① X publishes K_X to all employees.
- ② Y securely passes message $M = \{Y, K_Y\}$ to X .
- ③ X produces a hash H of the message, i.e., $h(\{Y, K_Y\})$.
- ④ X produces $\{M, \{H\}_{K_X^{-1}}\}$.

This last then becomes Y 's *certificate*.

What has been accomplished? What assurances does the certificate provide to a third party? What assumptions are required?

Y 's certificate is X 's affirmation of her identity. Anyone can validate the signature with X 's public key. Only X could have produced it. The hash ensures that it was not altered.

Now Y can certify the identify of her subordinates in a similar manner. She appends her certificate to each of theirs. This provides a chain of validations back to X .

Thus, an individual's certificate contains a chain of evidence rooted at some authority.

There is also a need for trust in situations where there is not a single hierarchy, such as on the Internet. Two individuals may not have a common "superior." Some entity may be designated as a certification authority (notary public, personnel office, security officer in a company, etc.).

On the Internet, several groups serve as "root certification authorities": C & W HTK, SecureNet, Verisign, Baltimore Technologies, Deutsche Telecom, Certiposte, and several others. These tend to be structured around national boundaries.

Why should a user trust the root CAs any more than any other certifier?

There are four base structures that certification authorities can adopt:

- Single root:** one agency acredits every other certification authority.
- Multiple roots:** many certification authorities with separate spheres of control.
 - Peers:** each organization has its own CA and a collection of CAs from which it will accept certificates.
 - Bridge:** one organization may have many CAs specializing in specific areas or relationships.

Consider the advantages and disadvantages of each of these schemes. When would each be appropriate?

The International Telecommunications Union (ITU) has issued a standard (X.509) for digital certificates that is the basis for many other protocols. An X.509v3 certificate has the following components:

- 1 **Version:** version of X.509 used;
- 2 **Serial number:** unique among certificates issued by this issuer;
- 3 **Signature algorithm identifier:** identifies the algorithm and params used to sign the certificate;
- 4 **Issuer's distinguished name:** with serial number, makes all certificates unique;
- 5 **Validity interval:** start and end times for validity;
- 6 **Subject's distinguished name:** identifies receiver of the certificate;
- 7 **Subject's public key info:** identifies algorithm, params, and public key;

X.509 Certificates (Cont.)

- 8 **Issuer's unique id:** used if an Issuer's distinguished name is ever reused;
- 9 **Subject's unique id:** same as field 8, but for the subject;
- 10 **Extensions:** version specific information;
- 11 **Signature:** identifies the algorithm and params, and the signature (encrypted hash of fields 1 to 10).

To validate the certificate, the user obtains the issuer's public key for the algorithm (field 3) and deciphers the signature (field 11). Then uses the information in the signature field (field 11) to recompute the hash and compare with the received value. Finally, check the validity interval.

Key Management

Modern secure systems use symmetric, asymmetric and hybrid methods for a variety of purposes. This imposes significant requirements for generating, protecting, distributing, storing and managing keys.

Comment on how symmetric and asymmetric systems differ with regard to these functions:

- generating keys
- distributing keys
- managing keys (how many are there)

A key management system must control keying material through the lifetime of the keys to prevent unauthorized disclosure, modification, substitution, replay and improper use.

- *Data confidentiality*: keys and other data are kept confidential when transmitted or stored.
- *Modification detection*: counter the threat of modification of data items.
- *Replay detection/timeliness*: have old data been interjected into current conversations or storage.

- *Entity authentication*: corroborate that identities are as claimed.
- *Data origin authentication*: is the source of a message as claimed.
- *Nonrepudiation of Reception*: proof that a message was actually received.
- *Notarization*: register messages to attest content and origin.

A key management system must provide a set of management services:

- *Entity Registration*: establish a registry of identities/credentials to allow authentication to the system.
- *Key generation*: allow producing keys that are of sufficient quality, secure, and not predictable.
- *Certification*: issue certificates for authentication purposes.
- *Authentication/Verification*: verify identities; three basic types:
 - entity authentication
 - message content authentication
 - message origin authentication

- *Key Distribution*: secure provide keys to parties requiring them
 - *encipherment*: key may be protected by encrypting with another key
 - *modification detection codes*: add redundancy to detect changes whether accidental or malicious
 - *replay detection*: measures to detect an attempt to foist off an old message as current
 - *proof of knowledge of the key*: allows proving that a party has and is able to use the key (e.g., challenge/response).

- **Key Maintenance:** preserve, protect and dispose of keys as needed
 - **Key Storage:** secure facility for protecting and accessing keys.
 - **Key Archival:** keys for notarization or non-repudiation may need long term storage.
 - **Key Replacement:** update key material when stale or compromised.
 - **Key Recovery:** re-establish keys lost through human error, software bugs or hardware malfunction
 - **Key Deletion:** securely destroy keys no longer needed.

The National Security Agency oversees encryption for the U.S. government. They classify encryption devices into four categories:

- Type 1:** NSA endorsed classified or controlled cryptographic item for classified or sensitive U.S. government information.
- Type 2:** NSA endorsed unclassified cryptographic equipment, assemblies or components for sensitive but unclassified U.S. government information.
- Type 3:** Unclassified cryptographic equipment, assembly, or component used, when appropriately keyed, for encrypting or decrypting unclassified sensitive U.S. Government or commercial information,
- Type 4:** algorithms that are registered by the NIST but are not FIPS published. Unevaluated commercial cryptographic equipment, assemblies, or components that neither NSA nor NIST certify for any Government usage.

NSA Encryption Classes (2)

In addition NSA recognizes two classes or “suites” of algorithms:

- Suite A:** NSA unpublished algorithms intended for highly sensitive communication and critical authentication systems.
- Suite B:** NSA endorsed cryptographic algorithms for use as an interoperable cryptographic base for both unclassified information and most classified information.

Don't the NSA restrictions violate the “no security by obscurity” dictum? Explain this apparent contradiction.

Maintaining Secrecy and Integrity

One might try to protect encryption technology by:

- restricting access to all implementations of the algorithm
- keeping the algorithm secret (security through obscurity)
 - NSA Type 1 and Type 2 encryption devices may implement classified algorithms
 - Kerchoff's law says that the best designs should allow the attacker to know everything but the plaintext and key.
 - A Type 3 algorithm is widely published, with several open source implementations.
- Hardware encryption devices typically use tamper-resistance technologies to prevent all but the most determined attackers from compromising the device
 - For consumer-grade devices, tamper-resistance may mean encasing the circuit in epoxy (“potting”)
 - Banking devices such as the IBM 4758 automatically zero-out the secret keys if drilled into, frozen or X-rayed
- Keep keys secret (depends on secure transfer of keys—bootstrapping problem). *Explain this remark.*

Many governments regulate the import, export and use of cryptography. Crypto is classified as a “munition” (weapon of war) or “dual use” (having civilian applications) technology.

During the Cold War, CoCom embargoed trade of munitions (including crypto) from the 23 member states of the Soviet Bloc.

CoCom is an acronym for Coordinating Committee for Multilateral Export Controls. CoCom was established in 1947, during the Cold War, to put an embargo on Western exports to East Bloc countries.

Until 1996, export of encryption technology from the U.S. required individual case-by-case licensing for each customer *when key sizes of greater than 40 bits could be used.*

Popular products (e.g. Netscape browser and Lotus Notes) were released in two forms: one with an effective 40-bit key where the other bits were revealed in each protocol exchange) and another allowing 128 bits.

While still regulated, the U.S. and other signatories to the Wassenaar Arrangement are more lenient. License exemptions exist for exporting open source crypto.

Some Poetry

What does this poem mean?

*Mary had a little key
(It's all she could export)
And all the email that she sent
Was opened at the Fort.*

—Ron Rivest