CPS221 Lecture: Encryption

last revised 10/27/14

Objectives

1. To introduce secret key, public key, and non reversible encryption
2. To introduce authentication based on certificates

Materials:

1. Projectable of binary code for “This is a test”
2. PublicKeyPrivateKeyDemo program
3. Projectable of Figure 2.3 from Stallings
4. Projectable of Figure 29.3 from Forouzan
5. Projectable of Figure 2.2 from Stallings
6. Encryption methods demo program
7. Projectable of Figure 29.12 from Forouzan
8. Projectable of Figure 29.9 from Forouzan
9. Projectable of Figure 29.14 from Forouzan
10. Projectable of Figure 29.16 from Forouzan
11. Projectable of Figure 29.19 from Forouzan

I. **Introduction**

A. Encryption is concerned with replacing information (called plaintext) with an encrypted form (called ciphertext)

1. Examples of encryption strategies often use text as an example, so that what is being encrypted is a sequence of characters.

   In practice, though, most encryption strategies are designed to work with binary data, and so can handle any type of information.

2. Of course, textual information can be represented as a sequence of numeric codes for individual characters; and each of these codes can be represented as a sequence of bits. The binary representation of a text is then the result of concatenating the codes for each individual character.

a) The examples in the book represent characters by numbers in the range 0 .. 25 (plus space = 26)

b) More commonly, characters are represented using an encoding scheme such as ASCII or Unicode.

Example: The message "This is a test" could be represented by the sequence of (binary) ASCII codes

PROJECT

```
01010100 01101000 01101001 01110011 00100000
01101001 01110011 00100000 01100001 00100000
01110100 01100101 01110011 01110100
```

3. However, for the sake of clarity, some of our examples will involve encryption of individual characters (or their numeric codes)

B. Most types of encryption are reversible - there is an inverse operation that (decryption) converts ciphertext back into plaintext.

1. A very, very simple example: replace each letter with its alphabetic successor (Z with A).

So "MEET ME AT NOON" encrypts to "NFFU NF BU OPPO"

This can be reversed by replacing each encrypted letter with its alphabetic predecessor (A with Z)

2. Reversible encryption (using a much stronger strategy, of course) is often used to protect information confidentiality.

a) Sensitive information can be stored in encrypted form. This is a defense against snooping (unless the snooper can obtain access to the plaintext form of the information before it is encrypted or after it is decrypted by a legitimate user, or can somehow discover the key)

b) An attacker who masquerades as someone having legitimate access to information cannot learn anything from it without also knowing how to decrypt it

c) Sensitive information can be transmitted in encrypted form. This is a defense against interception.

3. Reversible encryption can also be used to protect information integrity, since generally one cannot meaningfully modify information one cannot read.

a) This can be used to protect against unauthorized modification or masquerading.

b) It is, however, not necessarily a defense against replaying - since an attacker can replay an encrypted message even if he cannot understand it!

c) We will see that reversible encryption can also be used in strategies used for authentication and to protect against repudiation.

C. Irreversible encryption is a process that does not have an inverse.

1. A very, very simple example: form the sum of the ASCII codes of all the letters in a message (ignore spaces and punctuation)

So "MEET ME AT NOON" encrypts to
77 + 69 + 69 + 84 + 77 + 69 + 65 + 84 + 78 + 79 + 79 + 78 = 908

2. Irreversible encryption (using a much stronger strategy, of course) can be used to ensure that the contents of a message have not been altered in transit, by separately transmitting both the message and the result of encrypting it. If encrypting the message as received in the same way produces the same result as the encrypted version, then it is less likely that the message has been altered in transit.

(This is what the MD5 checksum you used in lab earlier in the course was for)

II. **Reversible Encryption**

A. Reversible encryption strategies have two components - an algorithm, and a key.

1. The encryption algorithm is generally not regarded as a secret. Indeed, efforts to protect information by hiding the method used to protect it (security by obscurity) are generally not strong.

2. Instead, encryption depends upon a key - whose exact form depends on the algorithm being used.

B. Reversible encryption algorithms fall into two broad categories.

1. Symmetric key strategies, in which the same key that is used to encrypt a message is also used to decrypt it

   a) In this case, if encryption is used for a message, then both the sender of the message and the receiver must know the key - it is a "shared secret". For this reason, strategies like this are sometimes called "shared key" or "secret key" strategies

   b) Of course, if encryption is used to protect stored information, only the owner of the information may need to know the key.

2. Asymmetric key strategies, in which different (but related) keys are used for encryption and decryption.

   a) In particular, if a messages is encrypted with one key, it can only be decrypted using the other key. (The key used to encrypt is of no use for decrypting.)

   b) Such strategies are sometimes called "public key" strategies because one of the keys (the one used for encryption) can be made totally public, while the other (known as the private key) is known only to one individual.

   c) To send a message using such a strategy, the sender can encrypt it using the recipient's public key. But only the recipient can decrypt it, using the private key.

DEMO: PublicKeyPrivateKeyDemo program (use 11 as the public key and 59 as the private key)

NOTE: This example is meant simply to illustrate the idea - it is far from secure!

3. Some comparisons of the two approaches:

   a) Shared key strategies suffer from several problems:

      (1) If it is desired to minimize the number of people knowing a given key, then it becomes necessary for each pair of users to share a unique key. For example, in a group of 10 people:

         (a) There are 45 different pairs. (Order doesn't matter). Hence, 45 secret keys are needed.

         (b) Each person needs to know 9 secret keys - one for each person being communicated with.

      (2) Such strategies also suffer from what is known as "the key distribution problem" - how are keys to be communicated between individuals in such a way that a third party does not become aware of the key?

      (3) Since both sender and receiver need to know the same key, such strategies are not directly useful for spontaneous communication

   b) Public key strategies do not suffer from these problems, but encryption and decryption typically involves a much greater amount of computation, making these strategies too inefficient to use for large messages.

c) In practice, a hybrid approach is often used, in which a secret key (known as a session key) is used just for one communication between a pair of individual, with a public key system used to actually transmit the key.

C. It is also possible to classify algorithms as stream algorithms or block algorithms.

1. In a stream algorithm, each character or byte is encrypted and later decrypted individually.

   PROJECT Stallings Figure 2.3 (p. 48) bottom

2. In a block algorithm, blocks of binary data (perhaps created from text) are the units of encryption and decryption.

   PROJECT Stallings Figure 2.3 (p. 48) top

III.**Shared Secret Key Reversible Strategies**

A. Shared Secret key strategies are of three different types

1. Substitution (ciphering) strategies
2. Transposition strategies
3. Hybrid strategies

B. In a substitution strategy, individual characters or blocks of digits are encrypted individually by applying some function that substitutes a different value for each character/block.

PROJECT Figure 29.3 from Forouzan

DEMO: Caesar, Vigenere, Stream strategies (strategy in Forouzan) using encryption strategies demo program.

(All examples are stream algorithms which use the key to specify an offset, with ‘’A’ meaning offset of 0, ‘B’ offset of 1 etc.)

C. In a transposition strategy, individual characters or bits are transposed as specified by the key

   DEMO: Transposition strategy using encryption strategies demo program.

   1. This example is a block algorithm, with the block size (in characters) equal to the length of the key.

   2. In this example, the key letters specify where the individual characters in each block go - ‘A’ means the first position in the encrypted block, ‘B’ means the second position ...

   3. In practice, transposition is more commonly done on the bit level, rather than with full characters.

D. Secret key encryption strategies are vulnerable to two kinds of cryptographic attack

   1. Brute force discovery of the key by trying all possible values.

      a) Observe that with Caesar and the Stream algorithm discussed in the text, the set of possible key values is very small - 26 if restricted to letters of the alphabet, or 256 if restricted to ASCII characters. In practice, a much broader range of key values would need to be used, of course - enough possibilities to make brute force discovery of the key infeasible even using the multiprocessor systems.

         Example: SSL and WEP use a stream algorithm known as RC4 that uses a key that can be anywhere from 8 to 2048 bits long.

         For brute force, the effort required is exponential in the number of bits in the key (i.e. the strategy requires trying each possible key to see if it produces a meaningful result.)

         PROJECT: Stallings Figure 2.2

2. Cryptanalysis. Here, the idea is to try to deduce the key from an analysis of the ciphertext.

   a) For example, an attacker may know at least the beginning of the plaintext from which a given ciphertext was produced (.eg. a standard header), and can use this information in an attempt to determine the secret key.

   b) Sometimes, an attacker has available both a ciphertext and the plaintext from which it was created - or even the encryption of a plaintext that the attacker has spoofed the target into encrypting.

   c) A key goal for an encryption algorithm is that even an attacker who knows the algorithm and who possesses several examples of ciphertext and the plaintext from which they were created will still be unable to discover the key in shorter time than what would be required for brute-force analysis.

3. One sort of secret key strategy is immune to either of these kinds of attacks: the use of a one-time pad, in which sender and receiver possess a sequence of keys, each of which is used for just one character of the message.

   PROJECT: Figure 29.12 from Forouzan

   The only vulnerability of this approach is disclosure of the key (e.g. in transit).

E. Both substitution and transposition, used alone, are vulnerable to cryptanalytic attacks. As we have noted, most practical secret key encryption strategies are hybrids that use a combination of substitution and transposition.

   1. The text discussed a hybrid algorithm known as DES (Data Encryption Standard) that was used starting in 1976.

      PROJECT Figure 29.9 from Forouzan

DES was a federal standard, widely used both in government applications and in the private sector (e.g. electronic funds transfer)

2. It is thought that algorithms like DES are safe from cryptanalytic attack and are only vulnerable to brute force attacks. (This comes from years of analysis of the algorithms and attempts to discover loopholes.)

3. At the time DES was first put into use, its 56 bit key was safe against brute force attacks using the technology of the time.

   a) There are 2^{56} possible keys. A brute force attack would need to try all possible keys until one worked - or almost 10^{17} possibilities - still a daunting number.

   b) However, in the last 1990's, brute force attacks using a special-purpose highly parallel machine succeeded in less than a day, calling into question whether its key length was still safe.

   c) DES has been replaced in practice by one of two approaches.

      (1) Triple DES does DES encryption three times on each block, using up to three different keys (yielding an effective key length of up to 168 bits).

      Recall, that, for an exponential problem, increasing the problem size by 1 doubles the effort required - so Triple DES with a 168 bit key requires 2^{112} (over 10^{33}) times the effort to crack using brute force than the effort required to brute force crack standard DES!

      (2) A new algorithm called AES (Advanced Encryption Standard) became a federal standard in 2002. There are three versions, using key lengths of 128, 192, and 256 bits.

IV. **Public Key Reversible Strategies**

A. Recall that shared key strategies have issues relative to key distribution and the need for a large number of keys.

B. A public key strategy, on the other hand, uses two keys - a public key which can be publicly disseminated, and a private key known only to the key's holder.

   PROJECT Figure 29.14 from Forouzan

C. The most widely used public key strategy is the RSA strategy, named after its inventors (Rivest, Shamir, and Adelman) who published it in 1978. A patent on it, granted in 1983, lasted 17 years, but the strategy is now in the public domain.

   1. An RSA key is generated as follows:

      a) The process uses two large (100's or 1000's of digits) prime numbers, called p and q in the literature. These are secrets known only to the key's owner.

      b) The product of the numbers (called n in the literature) is published as part of the public key.

         (1) The security of the algorithm rests on the known difficulty of the factoring problem - the only known way to discover p and q would be to factor n, believed to require time exponential in the number of bits in the factors using a brute-force approach.

         (2) If it a non-exponential factoring algorithm were discovered, the security of an algorithm like RSA would vanish in an

instant - since once the factors of n (which is part of the public key) are known, discovering the private key is trivial.

c) The key's owner also generates two numbers (called e and d in the literature) which have the property that they are both relatively prime to (p-1)(q-1), and e * d mod ((p-1)(q-1)) is 1. (In essence, one can choose almost any value for e that is relatively prime to (p-1)(q-1) and find a suitable d value.) The value of e is published (along with n) as the public key; the value of d (along with n) constitutes the private key.

2. To encrypt a number P

   a) P may be a binary number to begin with (e.g. an image) or may be derived from a textual message - in which case it is converted to a binary number.

   b) The number of bits in P must be < $\log_2 n$. (Longer messages are encrypted as a series of blocks.)

   c) The sender computes and sends $C = P^e \bmod n$.

3. To recover the original value of P, the receiver, computes $C^d \bmod n$.

4. Example: Suppose, we use p = 7 and q = 37 . (Very unrealistically small values, but chosen to make arithmetic simple)

   a) n = 7 x 37 = 259

   b) (p-1)*(q-1) = 6*36 = 216

   c) Suppose we choose e = 11 (which is relatively prime to 216)

d) Then we use $d = 59$, since $11 * 59 = 649$ and $649 \% 216 = 1$ and 59 is relatively prime to 216

   This yields the values we demonstrated earlier (DEMO again)

5. RSA works because (a result from number theory), for any $P < n$, $P^{ed} \bmod n = P$ when e and d have the properties indicated.

6. A public key system actually works both ways - i.e. if a message is encrypted with the sender's <u>private</u> key it can be decrypted with the sender's <u>public</u> key

   DEMO use 59 to encrypt and 11 to decrypt

7. With RSA, the security of the algorithm depends on the length of the key. A key length of 1024 is considered safe for the foreseeable future.

8. Of course, actually using RSA involves significant computation to raise a message (or block of a message) to the power e (or if encrypted to d). In general, e and d are not small numbers - in fact, the encryption is less secure if e is small. (You have a homework problem which asks you what would be the case if e is 1).

   a) The problem is not as bad as it might appear at first. It would appear that calculating P^e would require e multiplications. Actually, it can be done by $\log_2 e$ multiplications and additions:

      (1) Suppose e has b bits (where b will be ceiling($\log_2 e$)

      (2) Calculate P^2 as P x P, then P^4 as P^2 x P^2, then P^8 as P^4 x P^4 ... P^b as P^{b-1} x P^{b-1}. [b multiplications in all]

(3) Add the computed powers of P which correspond to 1's in the binary representation of e [b additions in all]

(4) Thus, the exponentiation required for encryption has complexity $O(\log_2 e)$

(The same would apply to calculating C^d, whose complexity would be $O(\log_2 d)$)

b) Nonetheless the computational effort for a large message is still significant, since the numbers being multiplied are themselves very large.

D. There are public key algorithms other than RSA, but RSA is by far the most widely used.

1. Some algorithms can only be used for a subset of the tasks RSA can be used for - e.g. Diffie-Hellman is only usable for key distribution.

2. One alternative: Elliptic Curve Cryptography (ECC) appears to offer equal security for a far smaller bit size, but is fairly new and has not been analyzed for vulnerabilities as thoroughly as RSA has.

V. **Hybrid Reversible Algorithms**

A. In practice, a public key algorithm like RSA is not generally used for encrypting entire messages - at least large ones.

B. Instead, one approach is this:

1. The sender creates a secret key.

2. The sender uses this key to encrypt the message

3. The sender encrypts the secret key using the public key of the <u>recipient</u>. (The secret key will be relatively short, so this is not nearly as complex as encrypting the entire message).

4. The sender sends both the encrypted secret key and the encrypted message to the recipient.

5. The recipient decrypts the secret key using his/her private key, then uses the decrypted secret key to decrypt the original message.

6. Note that the sender and receiver do not need to share a key in common, so this can be used even when there has been no prior interaction between the two.

C. An approach like this is part of the secure socket layer (SSL) used with, for example, with https.

1. As part of establishing a connection, the client chooses a random number and uses it to generate a symmetric session key for that one connection. This session key is then used to encrypt other transmissions between the client and server during one session.

2. The client encrypts the random number with the server's public key. The server decrypts it with its private key and generates the same session key from it.

3. No one else is able to learn the session key; hence, communication encrypted by it is secure.

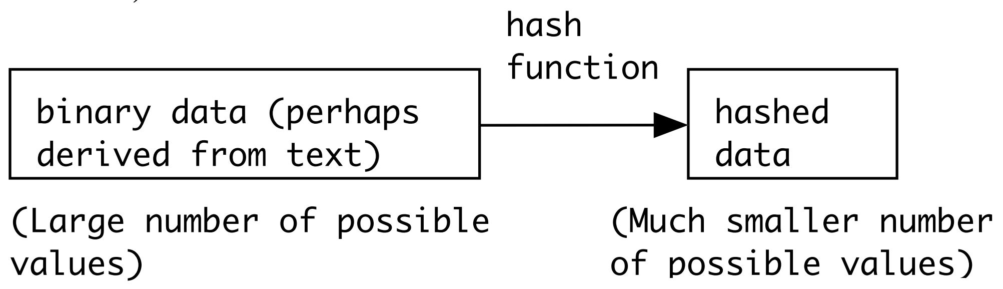
VI. Non-Reversible (One-Way) Encryption

A. The encryption strategies we have considered thus far have been reversible - given the encrypted text and the appropriate key, it is possible to reconstruct the original message.

B. There is also a category of encryption algorithms that are not reversible - i.e. there is no way to extract the original message from the encrypted form.

1. In fact, in general, the encrypted form is smaller than the encrypted message. Thus, several different messages will encrypt to the same value - so necessarily there is no way of recovering the correct original value.

2. Strategies used are variants of hashing (which you will study in CPS222)


hash function
binary data (perhaps derived from text)
hashed data
(Large number of possible values)
(Much smaller number of possible values)

a) Hashing strategies are used in search structures (e.g. Java HashSets and HashTables). In this case, irreversibility is not an issue.

b) When used cryptographically, we want to ensure that there is no feasible way to reconstruct the original data given the hashed data, or to construct an arbitrary message that hashes to a given value.

3. When hashing is applied to a message to produce a much smaller hashed value, it is commonly called a digest.

C. So what is this sort of encryption good for? It can be used for authentication and non-repudiation.

1. You have already seen an example of this. When you downloaded a disk image of Ubuntu, you used MD5 to create a digest of the downloaded image which you then compared to a cryptographic digest posted on the Ubuntu site. This was intended to assure you that the image you downloaded had not been tampered with. The hashing algorithm used (MD5) is designed so that it is virtually impossible to change the contents of the disk image it protects in such a way as to yield the same md5 checksum as the original, unaltered version.

a) The name "MD5" stands for "Message Digest Algorithm 5".

b) Actually, MD5 has been shown to have significant weaknesses Hence, another algorithm is preferred in many cases.

(1) One that is widely used is SHA-1 (Secure Hash Algorithm 1 - a successor to an algorithm that was shown to have weaknesses.)

(2) Because SHA-1 produces just a 160 bit hash, newer variants that produce longer hashes are taking its place (SHA-256, SHA-384, or SHA-512)

2. In general, the sender of a message can enable the receiver to be sure the message has been transmitted correctly as follows:

   a) Create a digest of the message using a well-known scheme such as MD5 or SHA-1.

   b) Send both the message and the digest to the recipient.

   c) The recipient can ensure the message's integrity by

      (1) Creating a digest of the message by using the same algorithm as the sender used.

      (2) Comparing the two digests. If they are the same, the recipient can be confident that the message received is the same as the message sent

         PROJECT: Figure 29.16 from Forouzan

      (3) Of course, this assumes that the digest is secure. If this is an issue, the digest can be encrypted using the sender's <u>private</u> key - in which case the ability to decrypt it using the sender's public key ensures its authenticity.

3. Non-reversible encryption is also typically used for storing passwords.

   a) When a user chooses (or is assigned) a password, the password itself is not stored - rather, a one-way hash is stored.

   b) When a user presents a password for login purposes, the system encrypts the password supplied by the user with the same algorithm used to encrypt the stored password.

c) If the encrypted password submitted matches the stored encrypted password, the user is assumed to have supplied the correct password.

d) Use of non-reversible encryption means that password recovery requires generating a new password - there is no way the original password can be recovered.

4. Non-reversible encryption can be used to ensure the integrity of a file (or group of files). If a digest is made of the file(s) to be protected and saved, it can be compared to a digest created later to ensure that the file has not been changed.

5. Non-reversible encryption can also be used to guarantee not only that a message has not been corrupted but also can prevent the sender of a message from later claiming to not have sent it (digital signature).

   a) The sender creates a digest of the message (using a well-known algorithm)

   b) The sender then encrypts the digest with the <u>sender's</u> <u>private</u> key.

   c) The sender sends the message (encrypted if necessary) and the encrypted digest to the recipient.

   d) The recipient can prove that the sender actually sent the message by demonstrating that the encrypted digest created by the sender, when decrypted with the <u>sender's public</u> key, is identical to a digest created using the same algorithm from the received (and decrypted if necessary) message.

   PROJECT Figure 29.19 from Forouzan