Reference: Computer Security by Stallings and Brown, Chapter 2

Intro to Cryptography
Lesson Introduction

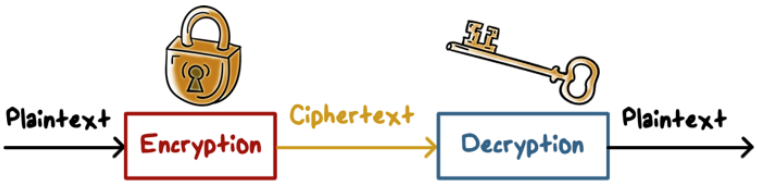
- Basics of encryption and cryptanalysis
- Historical/simple schemes
- Types of cryptography and how they are used for security

Cryptography is the foundation of security. In this lesson, we will first introduce the basics of encryption as well as attacks on encryption schemes. We will highlight several historical and simple encryption schemes. We will then introduce three types of modern-day cryptography and how they are used in security.

Instructor Notes:

NSA and the Cryptoapocalypse

Encryption/Decryption



- There is a **one-to-one mapping**
- Provides **confidentiality protection**

Encryption is the most often used cryptographic operation. It is a process of converting data into a form that is unintelligible to the unintended or unauthorized party. The authorized party can reverse this process. That is, converting the data into the intelligible form. We call the readable data, the plaintext and the unintelligible data, the ciphertext. So encryption, then, is the process of converting plaintext data into ciphertext, and decryption is the reverse operation. That is converting ciphertext into plaintext.

Notice that, in order to perform decryption, the authorized party needs to have the decryption key. There's a one-to-one mapping between a plaintext and a server text, so that the decryption process can always get back the original plaintext. Encryption protects data confidentiality, because only the authorized party with the proper secret, we call it a key, can decrypt and read the data.

Encryption/Decryption

Other services:

- **Integrity checking:** no tampering
- **Authenticity:** verified authorship
- **Authentication:** not an imposter

Encryption also provides other security services. These include integrity checking, that is to make sure that there is no tampering of data. It can also assure authenticity of message. That is it can verify the authorship of the message. It can also provide authentication. That is, to make sure that the party that you are communicating with is not an imposter.

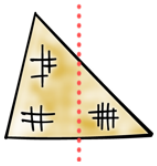
Encryption Basics

Ancient crypto:

- Early signs of encryption in Egypt in ~2000 B.C.
- **Letter-based scheme** (e.g., Caesar's cipher) ever since

cipher, as well as other similar schemes.

Encryption Basics



- **Symmetric ciphers:**
  - From ancient time to the presence
- **Asymmetric ciphers**
  - First by Diffie-Hellman-Merkle in 1976

Encryption Basics

- **Hybrid schemes -** most protocols now use both:

- **Asymmetric ciphers** for authentication, key exchange, and digital signatures

- **Symmetric ciphers** for encryption of data/traffic

authenticity.

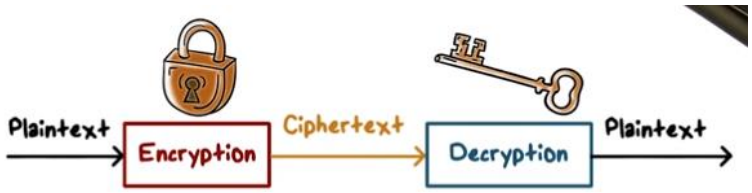
Instructor Notes:

Caesar Cipher

We often call an encryption scheme a cipher. You may not have realized this, but encryption has been used for thousands of years. For example, there is an evidence that Ancient Egyptians used some sort of ciphers. And then there is the famous Caesar's

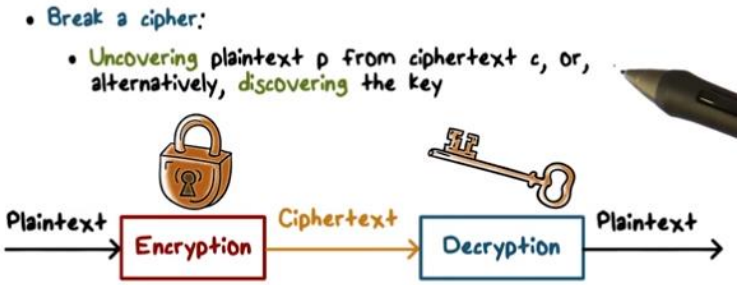
There are several types of ciphers. Symmetric ciphers, range from ancient schemes to present day algorithms. Asymmetric ciphers are relatively new, only invented in the late 70s.

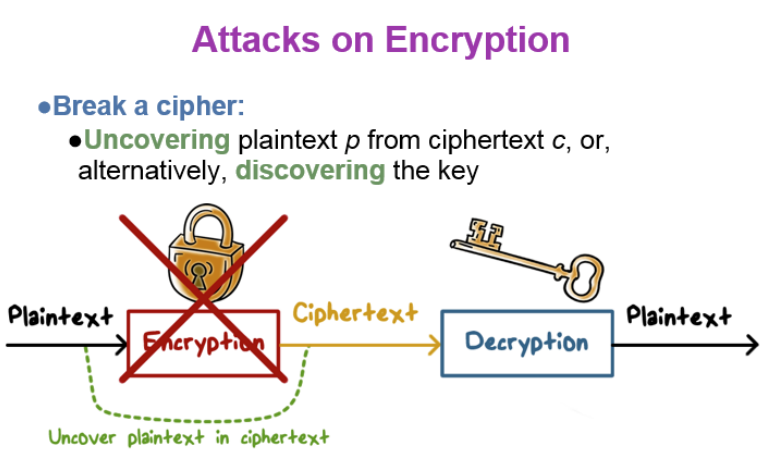
Most security protocols now use both types of schemes. Typically, they first use asymmetric ciphers to authenticate both parties involved in the communication. And then establish and exchange a encryption key for the communication. Then they use symmetric ciphers, and the established encryption key to encrypt data and traffic. They can also use asymmetric ciphers to digitally sign the data, so that they can ensure data



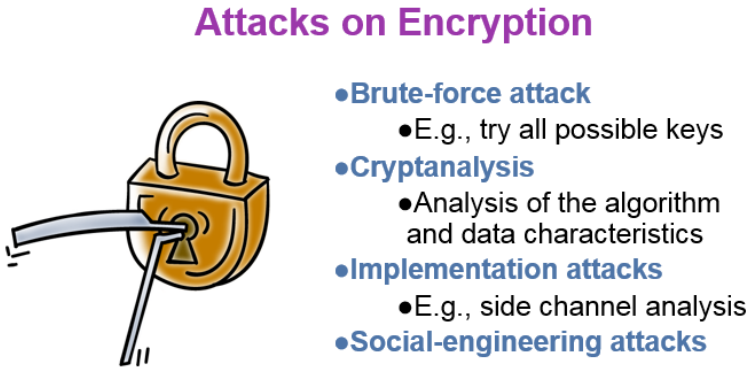
Given that encryption or cryptography in general, plays such an essential role in security, we can only expect that attackers will always try to break an encryption scheme.

Typically, the goal of such an attack is to uncover the plaintext from the ciphertext or to discover the encryption key.





For example, an attacker may try to recover the plaintext from the ciphertext that is transmitted on the Internet. And you may ask, how can the attacker obtain such ciphertext that's transmitted on the Internet? The answer is, actually this is very feasible because an attacker can use a packet capturing tool to capture the traffic, and then extract the ciphertext in the traffic. And then from there, the attacker can attempt to recover the plaintext from the captured ciphertext. So as a rule of thumb, we should always assume that the ciphertext that we transmit over the Internet can be captured by an attacker. Alternatively, the attacker may try to discover the encryption key, so that he can then decrypt all the data encrypted using this key.



There are several attack methods. The first simplest and yet the most inefficient way is to use brute-force, or to search blindly. For example, an attacker can try all possible keys, one by one, until the one that can decrypt the ciphertext properly into plaintext. And you may ask, how does the attacker know that the decryption with a key has worked properly? Typically, the attacker knows what the plaintext should look like. For example, if the plaintext is an English sentence, then only the correct key can decrypt the ciphertext into data that can be read as English. This

method is very inefficient because the number of possible keys can be huge, and so brute-force or search blindly can take a long time to succeed.

Another approach is to use cryptanalysis. Here, a attacker has some knowledge of the encryption algorithm and perhaps the characteristics of data, such as, distribution of certain letters or words. With such knowledge, the attacker can do a lot better than using brute-force to search the entire key space.

Attacks on Encryption

- **Brute-force attack**
  - E.g., try all possible keys
- **Cryptanalysis**
  - Analysis of the algorithm and data characteristics
- **Implementation attacks**
  - E.g., side channel analysis
- **Social-engineering attacks**

Attackers can also exploit implementation or system's issues. For example, it was shown that by using side channel analysis, for example, by observing the power consumption used by a crypto system, an attacker can deduce values of certain bits of key, and therefore the attacker can significantly decrease the key space that he needs to search.

Finally, let's not forget that the weakest link in security is the naive users. And they can be exploited using social engineering tricks. For example, an attacker can pretend to be a sysadmin who has just forgotten the key, and he called an unsuspected user for the encryption key to a system.

Encryption Attack Quiz

If the only form of attack that could be made on an encryption algorithm is **brute- force**, then the **way to counter such attacks** would be to...

- ○ use a longer key length
- ○ use a shorter key length
- ○ use a more complex algorithm
- ○ use a harder to guess key

Now let's do a quiz. Suppose the attacker can only use brute force to attack a crypto system. Then which of the following ways can be used to counter such attacks?

- ☑ use a longer key length
- ☐ use a shorter key length
- ☐ use a more complex algorithm
- ☐ use a harder to guess key

Should we use a longer key? The answer is yes. Because a longer key length means more keys, which means the attacker has to search a lot more keys. Should we use a shorter key length? Obviously not. Should we use a more complex algorithm? The answer is no. Because the attacker is to use brute-force to blindly try the keys one by one. So, his search is independent of the complicity of the encryption algorithm. Should we use a harder to guess key? The answers no. Because the attackers is going to use brute-force again to search blindly. That is, he's going to try to key one by one, he's not going to guess the keys. So, the correct answer is to use a longer key length. Again, this would mean that there are more keys for attacker to try.

Simple Ciphers Quiz

Use Caesar's cipher to decode the message:

LQIRUPDWLRQ VHFXULWB

Enter your answer in the text box:

Let's review some simple ciphers. The most famous one is Caesar's cipher. It works by mapping a letter to another letter by always the same amount of shift. For example, if A is mapped to D, that means the shift is three, and so the letter B is then mapped to E. The shift amount is the secret. Or the key of this encryption scheme. So an attacker only needs to try 26 possible keys. Now let's decipher this message using Caesar's cipher. Here, A is meant to D and B is meant to E. Again, the fixed shift amount is three.

ABCDEFGHIJKLMNOPQRSTUVWXYZ
XYZABCDEFGHIJKLMNOPQRSTUVW

LQIRUPDWLRQ VHFXULWB

Enter your answer in the text box:

Information Security

The answer, meaning the plain text, is information security. As you can see, this cipher is not difficult to solve, because the Caesar cipher shifted letters by three. On the other hand, this code was secure for Caesar, because most of his enemies at the time, were either illiterate, or would make the assumption that it was in a different language, that they do not understand anyway. So in this case, social engineering worked to the advantage of information security.

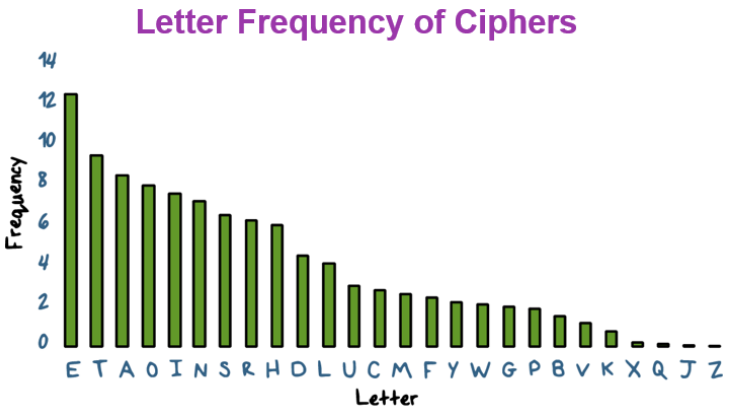
Simple Ciphers

- **Caesar's cipher (or, shift cipher):**
  - E.g., A → D, B → E
  - That is, shift by an offset *n*:
    - (letter + *n*) mod 26
  - **only 26 possible ways** of secret coding
- **Monoalphabetic cipher (or, substitution cipher):**
  - **generalization**, arbitrary mapping of one letter to another
  - 26!, ~4×10^{26} or ~2^{88}
  - Attack with statistical analysis of letter frequencies

Caesar's cipher is also called shift cipher, because each letter is map to another letter by fixed amount of shift. So if we represent letters as numbers, meaning A is 1, B is 2 and Z is 26. Then this encryption scheme or cipher can be represented as the letter, meaning the integer plus n, the fixed amount marked 26. Here the shift amount n is a secret, or the key. And obviously, there are only 26 possible keys. Therefore, it is easy to break Caesar's cipher because you only need to try 26 possible keys. A generalization of this scheme is to allow arbitrary mapping of one letter to another. That is, we no longer require that a letter is map to another letter by always the fix amount of shift. Now of course, we need to avoid that two letters are being mapped to the same letter. Demapping, meaning how each letter is map to another, it is now the secret key. Can you tell, how many possible keys are out there? Meaning, how many possible ways to map one letter to another. The answer is 26 factorial, because there are 26 factorial number of ways to map one letter to another. Please note that n factorial is much larger than 2 to the n. For example, 26 factorial is approximately 2 to the 88th, which is much larger than 2 to 26th. In other words, this is very huge number. If the attacker attempts to use brute force, or just blindly search all the possible keys, it's going

to take him long time, because there are 2 to the 88th number of possible keys. So, what should the attacker do instead? Instead of trying all of the possible keys, an attacker can analyze the statistical frequencies of letters to break the schemes. For example, in English, the most frequently used letter is E. And if in a cipher text, the letter X is the most frequent, then there's a high probability that E is mapped to X.

For substitution ciphers rather than trying all possible keys, we can use the frequency of letters. For example, here is the frequency distribution of English letters.



Instructor Notes:

Letter Frequencies

Letter Frequency of Ciphers

- What is plaintext for:

IQ IFCC VQQR FB RDQ VFLLCQ NA RDQ
CFJWHWZ HR BNNB HCC
HWWHBSQVQBRE HWQ VHLQ

WE WILL MEET IN THE MIDDLE OF THE LIBRARY
AT NOON ALL ARRANGEMENTS
ARE MADE

- In practice, also consider frequency of letter pairs, triples

Now let's do an exercise together. With the frequency distribution of English letters, can we figure out the plain text for this following cipher text? To start we notice that the letter Q here in the cipher text is the most frequent. And we know that in plain text English, the letter E is the most frequent. So, there's a high probability that Q is in fact the plain text letter E. Then we can look at the three-letter words, for example, here and here. Now we know that Q is in fact E. So both end in E, these two three-letter words. So most likely one of them would be the, T-H-E. The other one would be are, A-R-E. So here we're going to start to use the knowledge of English language. In other words, what are the common words or what are the legitimate words in English to help us uncover the plain text? For example, we can see if we say H is A and R is T, then we have a two letter word here, HR. In this case, if H is A and R is T, that would be at, which is a commonly used two letter word in English. And also, if H is A, then we have this three letter word that has two letters the same. So if H is A, then this can be all. So we can continue with this process by using our knowledge of the English language to uncover the plain text. Here we use both the frequency distribution of the English letters and our knowledge of English words to help us. If you continue with this process, you should be able to uncover this following plain text. We will meet in the middle of the library at noon, all arrangements are made. In practice, in addition to the frequency distribution of English letters, we can also consider the frequency of letter pairs, or even triples.

Monoalphabetic Cipher Quiz

Try to decipher this method using the Monoalphabetic Cipher:

WAIT IT WAS SAD

Enter your answer in the text box:

Instructor Notes:

- 500 Most Commonly Used Words in the English Language
- Relative Frequencies of Letters in the English Language

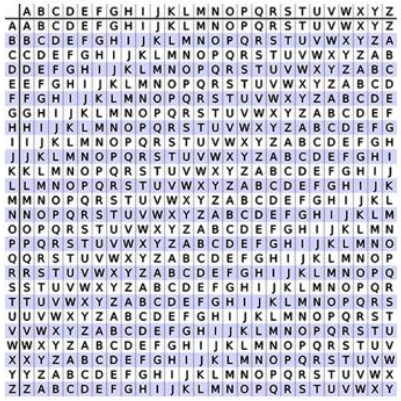
Enter your answer in the text box:

This is the end

Instructor Notes: There is a mistake in the answer to this quiz. The correct answer is "This is the ehd".

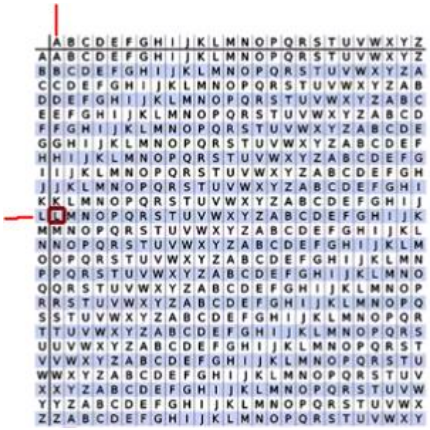
Vigenere Cipher

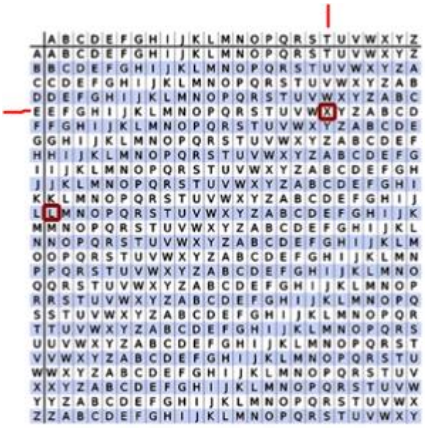
- Plaintext:
  ATTACKATDAWN
- Key:
  LEMON
- Keystream:
  LEMONLEMONLE
- Ciphertext:
  LXFOPVEFRNHR



Here's a real example of a poly alphabetic substitution cipher. It is called the Vigenere cipher. Vigenere cipher has a clever way of representing possible mappings from one letter to another as a matrix. Here we encrypt the plaintext by processing one letter at a time across the columns. We use the letters in a key to tell us which row to look at the mapping.

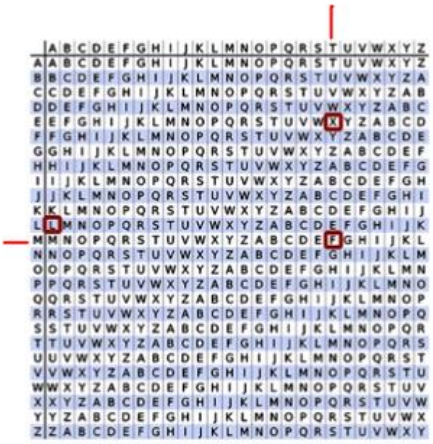
Now suppose we have a plain text, and the first three letters are A-T-T, and we have a key. In effect, the key stream should be as long as the plaintext. So this will be lemon, lemon, lemon, and go on. Here, the first three letters of the key stream are L-E-M. Now let's see how we can encrypt the plaintext according to the key. We encrypt the plaintext by processing the letters one at a time across the columns. The first letter is A, so we look at column A. We use the letter in the key, L to look at the row to decide the mapping of A. Here the mapping says L, which means in server text, A has been mapped to L.





The next plaintext letter is T. So we look at column T and we use the letter in the key to look at the row to decide the mapping. The letter is E in the key. And so we look at column T and row E. And the mapping is X. This means that the cipher text of T is X.

The next plaintext letter is T again. So look at column T again, but here the letter in the key is M, so we look at column M, and the mapping of column T, row M, is F. This means that the cipher text of plain text T is F, therefore for plain text A-T-T and key L-E-M, the corresponding cipher text is L-X-F.



Vigenere Cipher Quiz

What **weaknesses** can be exploited in the Vigenere Cipher?

- ☐ It uses a repeating key letters
- ☐ It requires security for the key, not the message
- ☐ The length of the key can be determined using frequency

For a long time the Vigenere Cipher was thought to be unbreakable, but later some attacks were discovered that can break this cipher. Can you see what weaknesses can be exploited to break the Vigenere Cipher?

Instructor Notes - Vigenere Cipher

- [x] It uses a repeating key letters
- [ ] It requires security for the key, not the message
- [x] The length of the key can be determined using frequency analysis

First it uses repeating key letters. As we have shown in the example, the key stream is as long as the plain text and the key letters will be repeated in a key stream. For example, the key stream can be lemon, lemon, lemon, and so on. Because the key letters are repeated, think even a long plaintext message. The same letters may be mapped to the same cipher text, because they have the same repeated key letters. For example, if the plaintext letters A-T-T appear multiple times in the plaintext, and the key letters L-E-M are being repeated, then there high probability that the same cipher text, L-X-F will also appears multiple times in the cipher text. Then using knowledge of the English language or the context of the communication, the attacker can guess what other words that may be repeated multiple times. Therefore, by searching for the repeated cipher text words, the attacker can then compare the cipher text words and the plaintext words to uncover the key letters. So this is a weakness that can be exploited.

Second, require security for the key, not the message. This is, in fact, a strength of the cipher.

Third, the length of the key can be determined using frequency analysis. This is also a weakness that can be exploited. For example, using knowledge of the English language, the attacker can look at the cipher text and determine that the most frequently used three letter word is the, T-H-E, and then by looking at the distance between the two occurrences of there, the calculator can then guess the length of the key. Knowing the length of a key would help the attacker uncover the whole key instead of just a few key letters. So therefore, this is a weakness that can be exploited to break the Vigenere Cipher.

What should be Kept Secret?

- **Kerckhoff's principle:**
  - A **cryptosystem** should be secure even if the attacker knows all details about the system, with exception of the secret key
- **In practice:**
  - Only use **widely known ciphers** that have been crypto analyzed for several years by good cryptographers
    - E.g., established standards

We have discussed that in the encryption key should be a secret. What about the encryption algorithm itself? In general, we should keep the algorithm open so that it can be revealed and improved by the broader community. More importantly, we don't have to rely on the secrecy of the algorithm for security. In other words, we don't have to use obscurity for security. Therefore, in practice we should always use the widely known and deployed algorithms and standards.

Types of Cryptography

Secret key cryptography:
- **one key** same key for encryption and decryption

Public key cryptography:
- **two keys**
  - Public for encryption, private for decryption
  - Private for signing and public for verification

There are several types of cryptographic algorithms.

The first is secret key cryptography. Here, the same key is used for encryption and decryption. In other words, the sender and the receiver of the confidential message, must use the same key.

Another type of algorithm is public key cryptography. Here, there are two key components that are paired or linked together mathematically. The public component is used for encryption. And a private component is used for decryption. For example, Alice can use Bob's public key to encrypt a message, that only Bob can decrypt. Because only Bob has the corresponding private key that will decrypt the message properly. The private key component is also used for signing a message. And then, the public key component can be used to verify the signature. For example, Alice can send a message using her private key, and anyone knowing her public key can verify that. Only Alice can produce this proper signature.

Hash Functions

- Compute message digest of **data of any size**
- **Fixed length output**: 128-512 bits
- Easy to compute $H(m)$
- Given $H(m)$, no easy way to find m
  - ***One-way function***
- Given m_1, it is computationally infeasible to find $m_2 \neq m_1$ s.t. $H(m_2) = H(m_1)$
  - **Weak collision resistant**
- Computationally infeasible to find $m_1 \neq m_2$ s.t. $H(m_1) = H(m_2)$
  - **Strong collision resistant**

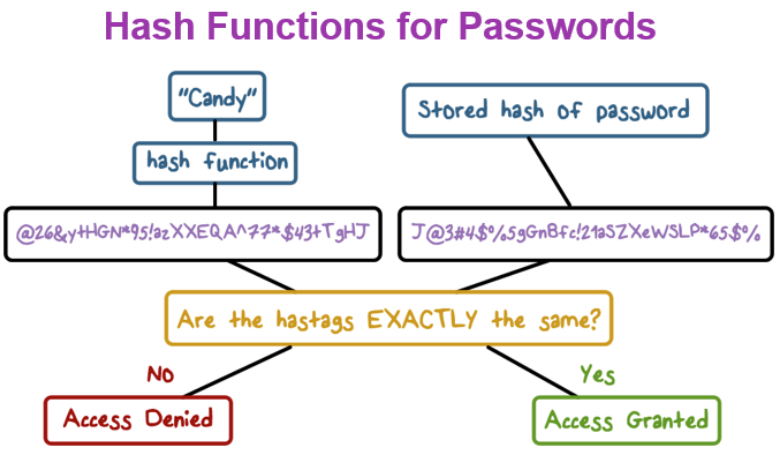
The third type of cryptographic algorithms is hash functions. Hash functions don't use keys.

A hash function computes the hash, or the message digest of data of any size. The hash or the message digest is a fixed length output, typically, in the range of 128 to 512 bits. Hash functions are ,typically, useful authentication and protection of message authenticity and integrity. In order to provide these services, hash functions must satisfy these following properties.

1. First, it should be relatively easy to compute a hash of a given message m. This would make hardware and software implementations practical. In other words, we want to be able to compute hash very efficiently.
2. Second, for given hash value, it should be completely infeasible to find the original message m, such that, the hash of m is this given hash value. This so called one way property is very important. For example, suppose Alice wants to authenticate herself to Bob. Alice would hash a secret, that she shares with Bob, as with the current timestamp together. For example, she can concatenate the secret s, with the current timestamp together, and then, hash over the concatenated value, and send the hash value over to Bob. Bob would know that, this must be Alice, because he can check the current time stamp, and he has the share secret S. In other words, Bob can take the current time stamp concatenate with the shared secret s, and compute his own hash, and compare with the hash value with the received hash value, if they match, then he knows that he's been communicating with Alice.

Now, because the hash value is transmitted over the Internet, and we must assume that an attacker can obtain any value that's transmitted over the Internet. That is, the attacker may be able to obtain this hash value. If the hash function is not one way, that means the attacker can reverse the hash function, and find out the input to the hash function, which includes the share secret s. If the attacker knows the share secret s, then the attacker can impersonate Alice to Bob, or Bob to Alice. So this is an example that demonstrates that the one way property of hash function is essential.

3. Another property of hash function is that given a message m1, it should be computationally infeasible to find a different message m2, such that they have the exact same hash value. This is the weak collision resistant property. This is an essential property for message authenticity and integrity protection.
   Imagine the following situation. Alice wants to send a message, m, to Bob. And she wants to make sure that Bob knows that Alice is the real author of m. So Alice is going to send m, along with a sign hash of m, meaning that Alice would hash m and then, sign hash, using her private key. Now an attacker can look at the traffic on the Internet, and find that Alice is sending m, and assign hash to Bob. If the weak collision resistant property is not there, then the attacker may be able to find a different message m pi, such that the hash of m pi would be the same as the hash of m, therefore, although the attacker has changed the message from m to m pi, the signature will still match. And therefore, when Bob receives m pi, and the original signature of hash of m, Bob will not know that the message has been changed from m to m pi. This is a simple example to show that the weak collision of system property is essential to protect message authenticity and integrity.
4. There's a stronger version of the collision resistant property. It says that it should be computationally infeasible to find different messages, m1 and m2, such that they have exactly the same hash value. This is a stronger property because it prevents the attacker from coming up with two different messages, that have the same hash value. Again, this property is essential for protecting message authenticity and integrity.
   For example, suppose Bob can write a IOU message, and send it to Alice to sign. Again, the way that Alice will sign it is to hash the message first, and then, sign the hash, using a private key. If Bob can find two different messages, that have the same hash value, one of which requires Alice to pay a small amount, and the other requires her to pay a very large amount. And the two different messages have exactly the same hash. Suppose Bob sends the message with a small amount for Alice to sign. After Bob receives Alice's signature on the message with the small amount, Bob can then go around and claim that Alice owes her a larger amount, because the two different messages, have the same hash. And the signatures would then be the same.
   It should be obvious that if a hash function satisfies the strong collision resistant property, it automatically satisfies the weak collision system property.



The one way property of hash function is extremely useful for security.

Recall that the one way property means that it is easy to compute a hash from the message, but it is impossible to find a message from the hash. You can think of it this way. You can make hashbrown from potatoes. But you can't make potatoes out of hashbrowns.

Let's look at this example of using hash to verify password. Hash is particularly useful for password verification. We all know that when we authenticate ourself to a system, we need to supply a password.

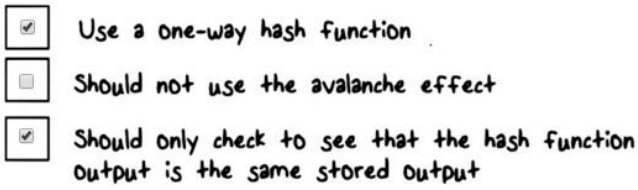
Now let's look under the hood. What happens? For example, when a user supplies the password "candy," the system would take the input "candy" and hash it, and then compare the hash value with the stored hash of a password on the system. If they match, then the access is allowed. Otherwise, access is denied. That is, the clear text password candy is never stored on the system. Only the hash of the password is stored on the system. And then whenever the user types in a password to authenticate him or herself to the system, the input will be hashed. And then the hash of the input password will be compared with the stored hash of the password. You may ask, why not just store the clear text password on the system? Think about it, if you do that then anyone who gains access to the system can look at the clear text password and then impersonate the user. This can include attackers or even other users on a system. Therefore, we never store the plain text of the passwords on system, only the has values. Of course, if an attacker gains access to system, he can steal the hash of the password. Then what the attacker can do is guess possible passwords one by one and hash each guess password and compare the hash value with the stolen hash of the password. If they match, then the attacker knows that he had correctly guessed the password. Therefore, it is important for us to use a password that's very hard to guess. For example, we should not use any word in the dictionary.

Hash Function Quiz

Which of the following characteristics would **improve password security**?

- [ ] Use a one-way hash function
- [ ] Should not use the avalanche effect
- [ ] Should only check to see that the hash function output is the same as stored output

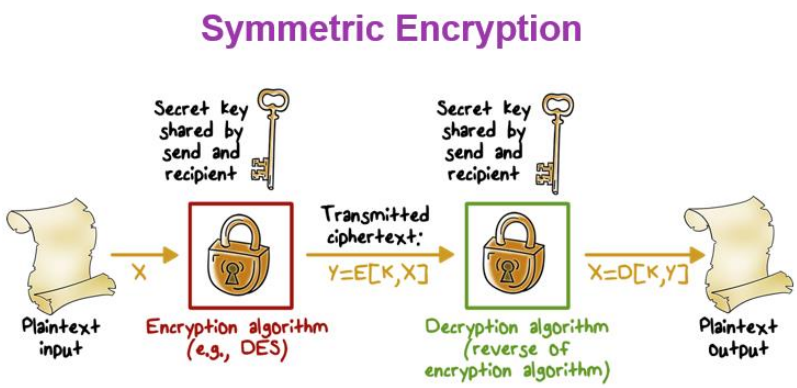
Now let's do a quiz. Since hash function is used to protect passwords let's think about what we can do to improve password security.



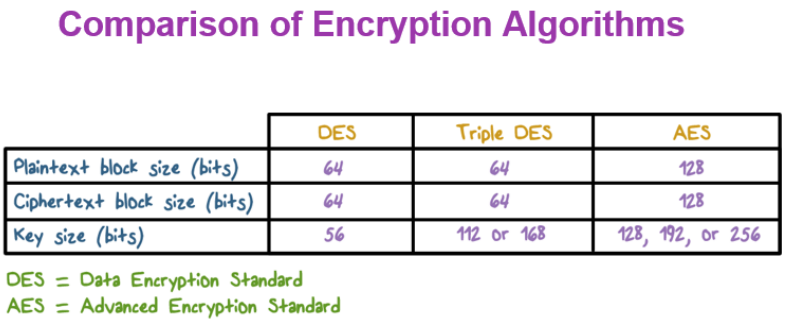
First, use a one-way hash function. This is obvious, because if the hash function does not have one-way property. Then from the hash value an attacker can easily reverse the hash function, and find the plain text password. So, we have to use a one-way hash function to protect password security.

Second, should not use the avalanche effect. The avalanche effect means that a slight change of the input may cause a large change of the output. In fact, we want our hash function to have this effect, so that two similar passwords will have very different hash values. With this in fact, even when the attacker has correctly guessed a password, he still has the same amount of difficulty to guess another similar password.

Third, should only check to see that the hash function output is the same as stored output. This obviously is yes. Because password authentication should only check that the hash function output is the same as the stored value.



Let's discuss symmetric encryption. Recall that in symmetric encryption, the same key is used for both encryption and decryption. The input is a plain text message. The encryption algorithm takes the plain text input, along with the encryption key, and processes the plain text using substitution and permutation to produce ciphertext. The decryption algorithm performs a reverse of the encryption process. It takes as input, the ciphertext, and the key, which is the same that was used for encryption, and it produced the original plain text.



The most commonly used symmetric encryption algorithms are the so called block ciphers. A block cipher takes as input a plaintext in fixed sized blocks, and it produces a block of ciphertext of equal size, for each plain text block. To process a longer plain text, we can first break the plain text into a series of fixed size blocks, and then, apply block ciphers to each block. The most important symmetric algorithms, all of which are block ciphers, are the data encryption standard, or DES, and the advanced encryption standard or AES. We will present more technical details of these algorithms in a later lecture. For now, we note that these two different algorithms have different block sizes and key length.

Comparison of Encryption Algorithms

Key size (bits)	Cipher	Number of Alternative Keys	Time Required at 10^9 descryptions/s	Time Required at 10^{13} descryptions/s
56	DES	$2^{56} \approx 7.2 \times 10^{16}$	2^{55} ns = 1.125 years	1 hour
128	AES	$2^{128} \approx 3.4 \times 10^{38}$	2^{127} ns = 5.3×10^{21} years	5.3×10^{17} years
168	Triple DES	$2^{168} \approx 3.7 \times 10^{50}$	2^{167} ns = 5.3×10^{33} years	5.8×10^{29} years
192	AES	$2^{192} \approx 6.3 \times 10^{57}$	2^{191} ns = 5.3×10^{40} years	9.8×10^{36} years
256	AES	$2^{256} \approx 1.2 \times 10^{77}$	2^{255} ns = 5.3×10^{60} years	1.8×10^{56} years

Recall that, a longer key length means that there are more keys. As a result, a brute force attack has to do more work to try all the keys. This table shows how much time is required for brute force attack for various key lengths. As we can see in the table, DES user key length of 56. And a number of possible keys, is 2 to the 56th. If a single personal computer is used to try all the possible keys, it takes about a year. If multiple PCs, for example, tens of thousands of PCs, work in parallel, then the time is drastically shortened. And today's super computers should be able to try all the possible keys, using DES, in an hour. Key sizes of 128 bits or greater are effectively unbreakable using, simply, brute force methods. For example, even with a super computer, it would take 10 to 17 years, to try all the possible keys using AES.

Symmetric Encryption Quiz

Select the correct definition for **each type of attack**:

A. A method to determine the encryption function by analyzing known phrases and their encryption
B. Analyzing the effect of changes in input on the encrypted output
C. Compare the ciphertexts with its known plaintext
D. A method where a specific known plaintext is compared to its ciphertext

- ☐ known-Plaintext attacks
- ☐ chosen-Plaintext attacks
- ☐ differential cryptanalysis
- ☐ linear cryptanalysis

Now let's do a quiz. There are various types of attacks on symmetric ciphers. Given the following attack methods, select the correct definition for each one.

The first one, known-Plaintext attacks. The answer is d. In this attack, a specific known plaintext is compared to its ciphertext.

Second, chosen-Plaintext attacks. The answer is c. In this attack method, the attacker can choose randomly some plaintext and obtain the corresponding ciphertext.

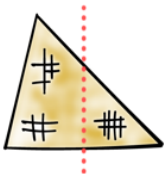
A. A method to determine the encryption function by analyzing known phrases and their encryption
B. Analyzing the effect of changes in input on the encrypted output
C. Compare the ciphertexts with its known plaintext
D. A method where a specific known plaintext is compared to its ciphertext

- [D] known-Plaintext attacks
- [C] chosen-Plaintext attacks
- [B] differential cryptanalysis
- [A] linear cryptanalysis

Third, differential cryptanalysis. The answer is b. In this attack, the attacker controls a changes of the input, meaning the input plaintext. And then observe the changes of the output, meaning the output ciphertext.

Four, linear cryptanalysis. The answer is a. In this attack method, the attacker models the relationship between plaintext, ciphertext, and the key, using linear equations. And then he uses the known ciphertext, plaintext pair to derive the key bits.

Asymmetric Encryption

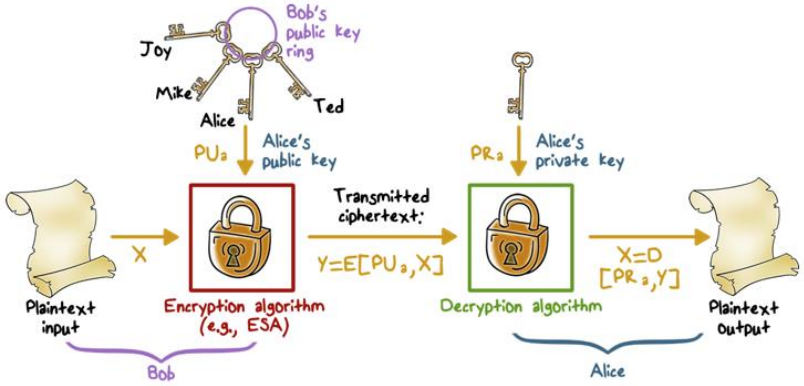


- **Plaintext**: Readable message or data that is fed into the algorithm
- **Encryption algorithm**: Performs transformations on the plaintext
- **Public and private key**: Pair of keys, one for encryption, one for decryption
- **Ciphertext**: Scrambled message produced as output
- **Decryption key**: Produces the original plaintext

Given a plaintext input, the encryption algorithm performs transformation to produce the corresponding ciphertext.

The decryption algorithm reverses the encryption process. That is, it takes as input the ciphertext and produces the original plaintext. Instead of using a single key, as in the symmetric encryption here, asymmetric encryption use a pair of keys: One is used for encryption and the other is used for decryption. The two keys are paired mathematically together. That is, if you use a key for encryption, only the corresponding paired key can decrypt a message.

Asymmetric Encryption



Let's look at the essential steps in asymmetric encryption.

First, each user generates a pair of keys that are paired together mathematically. One is the private key, the other is the public key. The public key can be published, for example, on a user's website or on a public repository. The purpose is for everybody to know your public key, but a companion private key has to be kept as a secret. That is, only the user should know his or her own private key.

Since public keys are meant to be known by others, we can imagine that in practice, a user such as Bob would have a collection of public keys of his friends. For example, in this case, Bob has the public keys of Alice, Ted, Mike and Joy. Now suppose Bob wants to send a private message to Alice. Bob will use Alice's public key, which is in his collection to encrypt a plaintext using an asymmetric encryption algorithm such as RSA and then transmit the ciphertext to Alice. It is mathematically guaranteed that only Alice can decrypt this ciphertext properly into plaintext, because only Alice has the corresponding companion private key. That is, if a public key is used to encrypt a plaintext into the ciphertext, only the companion corresponding public key can decrypt the ciphertext properly into the plaintext. Therefore, in this case, only Alice can see the plaintext.

Asymmetric Encryption Quiz

Check all tasks for which asymmetric encryption is better:

- [ ] provide confidentiality of a message
- [ ] securely distribute a session key
- [ ] scalability

Now let's do a quiz on asymmetric encryption. Check all tasks where asymmetric encryption is better than other algorithms.

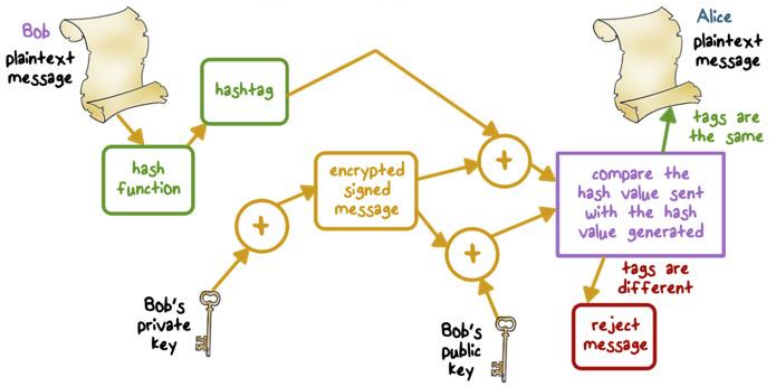
- [ ] provide confidentiality of a message
- [x] securely distribute a session key
- [x] scalability

First provide confidentiality of a message. This is typically not a strength of asymmetric encryption, because asymmetric encryption is much slower than symmetric encryption. And therefore let me try to protect the confidentiality of a message, we typically use symmetric encryption instead of using asymmetric encryption. Therefore, this is not a task that asymmetric encryption is good at.

Second, securely distribute a session key. A session key is used to encrypt all messages during a session. For example, a web browsing session. We can use a session key to encrypt and provide confidentiality of all the communications within that session. This is a task that asymmetric encryption is better at. For example, if Alice and Bob wants to establish a session key between them. It is easy for Bob for example, to create a session key and then encrypt the session key using Alice's public key and send over to Alice. It is guaranteed that only Alice can decrypt a message and extract the session key, because only Alice has the corresponding private key. Therefore, this is a task that asymmetric encryption is good at.

Third, scalability. Imagine we have N users and they need to communicate with each other securely. For example to have confidentiality in their communication. In this case, we need N square number of keys. Whereas for asymmetric encryption, regardless of how many users are involved. We only require that each user has one public key and one private key. Therefore, asymmetric encryption is much more scalable.

Digital Signatures

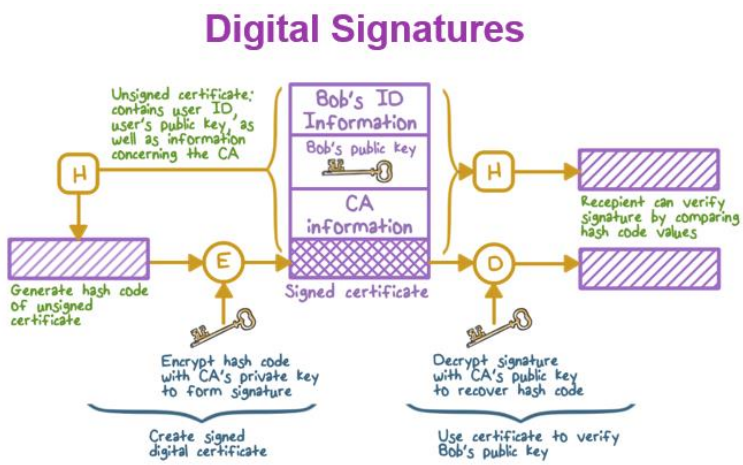


The point of public key encryption is that the public key component is really public. That is, any user can send his or her public key to any other user or just broadcast it to the world.

Although this approach is very convenient, it has a major weakness. That is, anyone can forge such a public announcement. Some user could pretend to be Bob, and send a public key to another user such as Alice, and tell Alice that this is Bob's public key. The result is that when Alice sends a private message to Bob saying she encrypts it using Bob's public key. But remember

this Bob's public key is actually forged by the attacker. Then the message can be intercepted by the attacker, and can be read by the attacker.

Now, at some point hopefully, Bob can discover that there's a forgery going on and a fake public key of his was being used. But then what can Bob do? Bob can send Alice another message saying that, hey, this is my real public key. But how could Alice tell? That is, how could Alice tell that the previous key was a forgery and this key, that Bob just sent, is real.

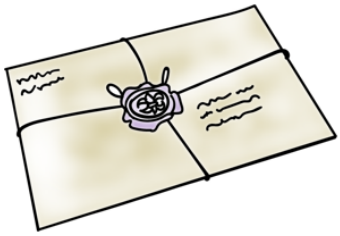


The solution to this problem of public key forgery is to use a public key certificate. In essence, a certificate consists of Bob's public key and Bob's information such as the user ID, let's say his name and address and so on. The certificate authority's information. And the whole blog is signed using the certificate authority's private key. The certificate can also include other information, such as the period of validity of this certificate, that is, for how long this certificate is valid for this public key, say, one year.

Now let's see how certificate is created, and how it is verified, and how it is being used to distribute public key. Suppose Bob wants the certificate authority CA to create a certificate for his public key. Bob would contact the CA and provide authentication information such keys driver's license and so on, and then he will send his public key to CA. The CA will then put his ID, his public key and other information such as the period of validity together and then hash it. And then the CA will use his private key to sign the hash. So that creates the certificate of Bob's public key. Now Bob can send this public key certificate to anybody such as Alice. When Alice receives this public key certificate, she can first extract the key types of information of Bob's idea, public key, and all the information. And then she will hash this data, and then Alice will also use the certificate authorities public key to decrypt the signature or verify the signature and compare these two hash values. If they match, that means this public key has been properly signed by the CA. In other words, this public key of Bob's has been validated by the CA. So this is how public key certificate works.

Now of course, the underlying assumption is that. The CA is a trusted party by everybody involved. In practice, the CA is a well-known company such as Verisign, Microsoft, Google, or Apple, and the public keys are already stored in, for example your web browser. That is with these public keys already configured on your system, they can automatically validate public key certificates signed by these entities.

Digital Envelopes

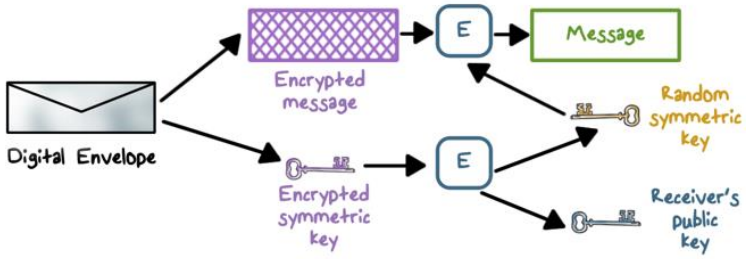


- Protects a message **without needing** to first arrange for sender and receiver to have the same secret key
- Equates to the same thing as a **sealed envelope containing an unsigned letter**

Recall we discussed that public key encryption is typically used to establish a symmetric key for encrypting messages for say between Alice and Bob. That is, before Alice and Bob can communicate securely they would use public key's encryption to exchange a sure and secret key. Of course, Alice and Bob can first establish this key before Alice can send Bob the first message encrypted using this key.

But with Public Key Encryption we can do better, meaning that we can do it in a more efficient way with so called digital envelopes. Alice can send Bob a message encrypted using a share key that she just created, and this share secret key itself is encrypted using Bob's public key, so that only Bob can decrypt and extract the share secret key. This is similar to the situation where only the intended recipient can open the sealed envelope.

Digital Envelopes



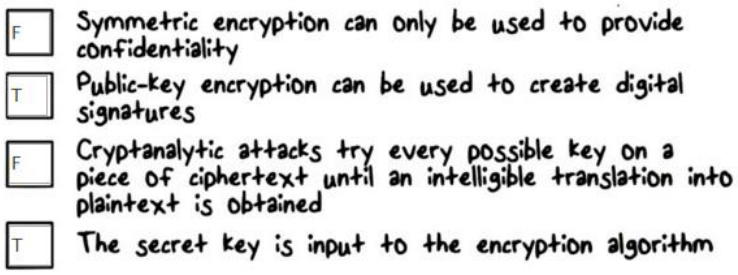
To illustrate, Alice first creates a symmetric key that she wants to share with Bob. She can encrypt a message using this share key, then she also uses Bob's public key to encrypt this share key. And then she can put the encrypted message and the encrypted key in envelope. Then she can put the encrypted message and the encrypted key together and send them to Bob. On the receiving end, Bob can use his private key to decrypt the encrypted share key. Once he gets a shared key, he can now decrypt the encrypted message and get a plain text message.

Encryption Quiz

Mark each of the statements either **T for True or F for False**:

- ☐ Symmetric encryption can only be used to provide confidentiality
- ☐ Public-key encryption can be used to create digital signatures
- ☐ Cryptanalytic attacks try every possible key on a piece of ciphertext until an intelligible translation into plaintext is obtained
- ☐ The secret key is input to the encryption algorithm

Now let's do a quiz. Decide if each of these statements is true or false.



First. Symmetric encryption can only be used to provide confidentiality. This is false because symmetric encryption can be used for other security services. For example it can be used for authentication. Suppose Alice and Bob share a secret. Then Alice can use the shared secret as the key and encrypt message using symmetric encryption algorithm and send the message to Bob to prove that she's Alice.

Second, public-key encryption can be used to create digital signatures. This is true. Given a message we can first hash the message and then encrypt the message using our public-key. The encrypted hash value becomes the digital signature of this message.

Third, cryptanalysis attacks try every possible key on a piece of ciphertext until an intelligible translation into plaintext is obtained. This is false, because what's described here is actually the brute force attack, or the spy research all possible keys until the ciphertext is translated into a plain text. Well as cryptanalytic attacks would use knowledge of the algorithm or the plain text such as the frequency of letters in order to break a scheme. In other words a cryptanalytic attack typically does not need to try every possible key.

Fourth, the secret key is input to the encryption algorithm. This is true. An encryption algorithm takes as its input, the plain text and a key.

Intro to Cryptography
Lesson Summary

- Encryption schemes and attacks on encryption have been around for thousands of years.
- Hash: no key, no encryption
- Secret key cryptography: same key for encryption and decryption
- Public key cryptography: public key for encryption and signature verification and private key for decryption and signins

Encryption schemes and attacks on encryption schemes have been around for thousands of years. The main attack approaches are brute force and cryptanalysis. One other use of cryptography includes some combination of hash, secret key cryptography, and public key cryptography.