Misr International University

جامعة مصر الدولية

Faculty of Computer Science
CSC 350 Operating Systems II

Report

Security for Operating Systems

Abdelaziz Mohamed Shehata
2017/10803

Table of Contents

Table of Figures & Tables:

1. ***Definition of Cryptography:***

Is a technique for securing information and communications through usage of codes so that only the intended people for whom the data is supposed to reach can understand and identify it.

Availability
Confidentiality
Integrity

Figure 1: Cryptography "CIA"

2. ***Cryptography Model; The CIA Triad:***

Is a guideline for information security for an organization.

- Confidentiality: Company policies should limit access to the data to authorized people and makes sure that only those authorized individuals can see this information.
- Integrity: is accuracy, consistency, and trustworthiness of the data during its entire life cycle. Data must not be changed during transmitting and not changed by unauthorized parities.
- Availability: Maintaining equipment, performing hardware fixes, keeping operating systems and software up to date, and creating backups ensure the availability of the network and data to the authorized users.

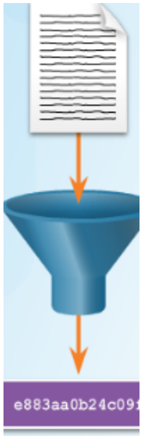
3. Algorithms Used in Cryptography:

In Cryptography the techniques which are used to protect information are obtained from mathematical concepts and a set of rule-based calculations known as ***algorithms*** to convert messages in ways that make it hard to decode it.

In order to encrypt and decrypt messages to secure communications among computer systems, devices such as smartphones, and applications.

3.1 *These algorithms are used for cryptographic:*

Table 1: Types of Cryptography.

Symmetric Key	Hash Functions	Asymmetric Key
The sender and receiver of message use a: **Single common Secret key** To encrypt and decrypt messages.	No usage of any key in this algorithm.  Plain Text Hash Function Fixed Length Hash Value	**A pair of keys** is used to encrypt and decrypt information. A public key: is used for encryption. A private key: is used for decryption.
Fast (Wire Speed)		Relatively Slow
Data Encryption System **(DES)**	Many operating systems use hash functions t**o: encrypt passwords. (MD5)**	**RSA** **ELGamal**

4. Cryptography Algorithm Used in:

Linux Operating System

“RedHat”

a. Diffie-Hellman Key Exchange

Diffie-Hellman Key Exchange is a popular mathematical key exchange algorithm.

It allows those who have never met before to safely create a shared key, even over an insecure channel that adversaries may be monitoring.

It doesn’t matter whether the intercepting party captures each piece of transmitted information, they will not be able to break the key in any way, other than the usual brute force method.

Table 2: key information of Diffi-Hellman

TimeLine	1976
Type of Algorithm	Asymmetric
Key Size (in bits)	512, 1024, 2048, 3072, 4096
Speed	slow
Time to Crack	Unknown
Resource Consumption	Medium

5. *Where is the Diffie-Hellman key exchange used?*

The main purpose of the Diffie-Hellman key exchange is to **securely develop shared secrets that can be used to derive keys.**

These keys can then be used with symmetric-key algorithms to transmit information in a protected manner.

Symmetric algorithms tend to be used to encrypt the bulk of the data because they are more efficient than public key algorithms.

Technically, the Diffie-Hellman key exchange can be used to establish public and private keys. However, in practice, RSA tends to be used instead.

Diffie-Hellman:

- Creates a shared secret between two (or more) parties, for **symmetric** cryptography
- **Key identity:** $(gen^{s_1})^{s_2} = (gen^{s_2})^{s_1} = \textit{shared secret} \pmod{\textit{prime}}$
- Where:
  - *gen* is an integer whose powers generate *all* integer in [1, *prime*)   (mod *prime*)
  - s_1 and s_2 are the individuals' "secrets", *only* used to generate the symmetric key

RSA:

- Used to perform "true" **public-key** cryptography
- **Key identity:** $(m^e)^d = m \pmod{n}$ (lets you recover the encrypted message)
- Where:
  - $n = prime_1 \times prime_2$ (n is publicly used for encryption)
  - $\varphi = (prime_1 - 1) \times (prime_2 - 1)$ (Euler's totient function)
  - e is such that $1 < e < \varphi$, and (e, φ) are coprime   (e is publicly used for encryption)
  - $d \times e = 1 \pmod{\varphi}$ (the modular inverse d is privately used for decryption)

As one of the most common methods for safely distributing keys, the Diffie-Hellman key exchange is **frequently implemented in security protocols such as TLS, IPsec, SSH, PGP, and many others**. This makes it an integral part of our secure communications.

As part of these protocols, the Diffie-Hellman key exchange is often used to help secure your connection to a website, to remotely access another computer, and for sending encrypted emails.

6. *How does the Diffie-Hellman key exchange work?*

The Diffie-Hellman key exchange is complex, and it can be difficult to get your head around how it works. **It uses very large numbers and a lot of math**, something that many of us still dread from those long and boring high school lessons.

Understanding the Diffie-Hellman key exchange with an analogy.

- Think of **two people mixing paint**.
- Their names are Alice and Bob.
- **They both agree on a random color to start with**.
- they send each other a message and **decide on yellow as their common color**, just like in the diagram below
- They do not tell the other party their choice.
- Alice chooses **red**,
- Bob chooses a **slightly-greenish blue**.
- Alice and Bob mix their secret color.
- According to the diagram, Alice ends up with an **orangish mix**, while Bob's result is a **deeper blue**.
- They send the result to the other party.
- **Alice receives the deeper blue**, while **Bob is sent the orange-colored paint**.
- they then add their secret color to it. **Alice takes the deeper blue and adds her secret red paint**, while **Bob adds his secret greenish-blue to the orange mix he just received**.
- **They both come out with the same color**, which in this case is a brown known as: The **common secret**.
- **The essential part of the Diffie-Hellman key exchange is that both endpoints end up with the same result, without needing to send the entirety of the common secret across the communication channel.**
- **As a result, the attacker has no opportunity to know the secret key** because the complete shared secret is never sent over the connection.

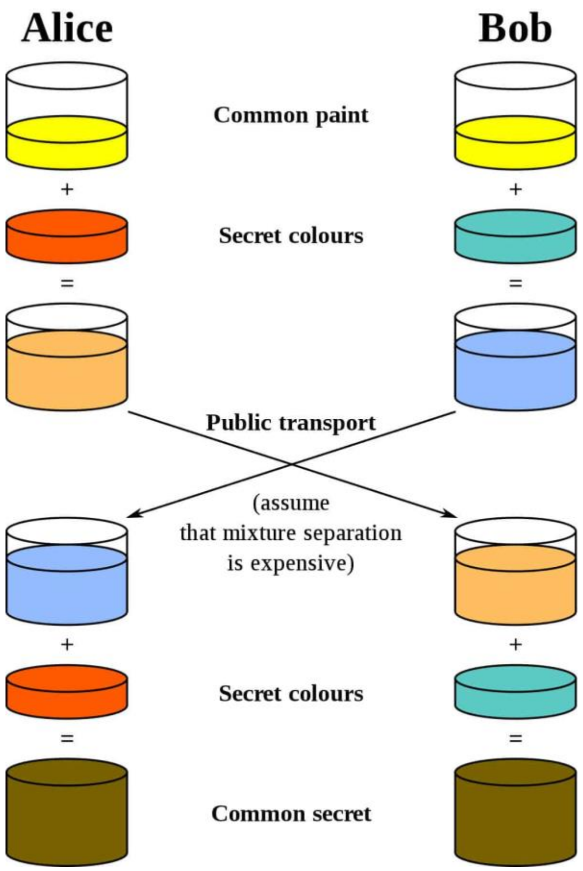
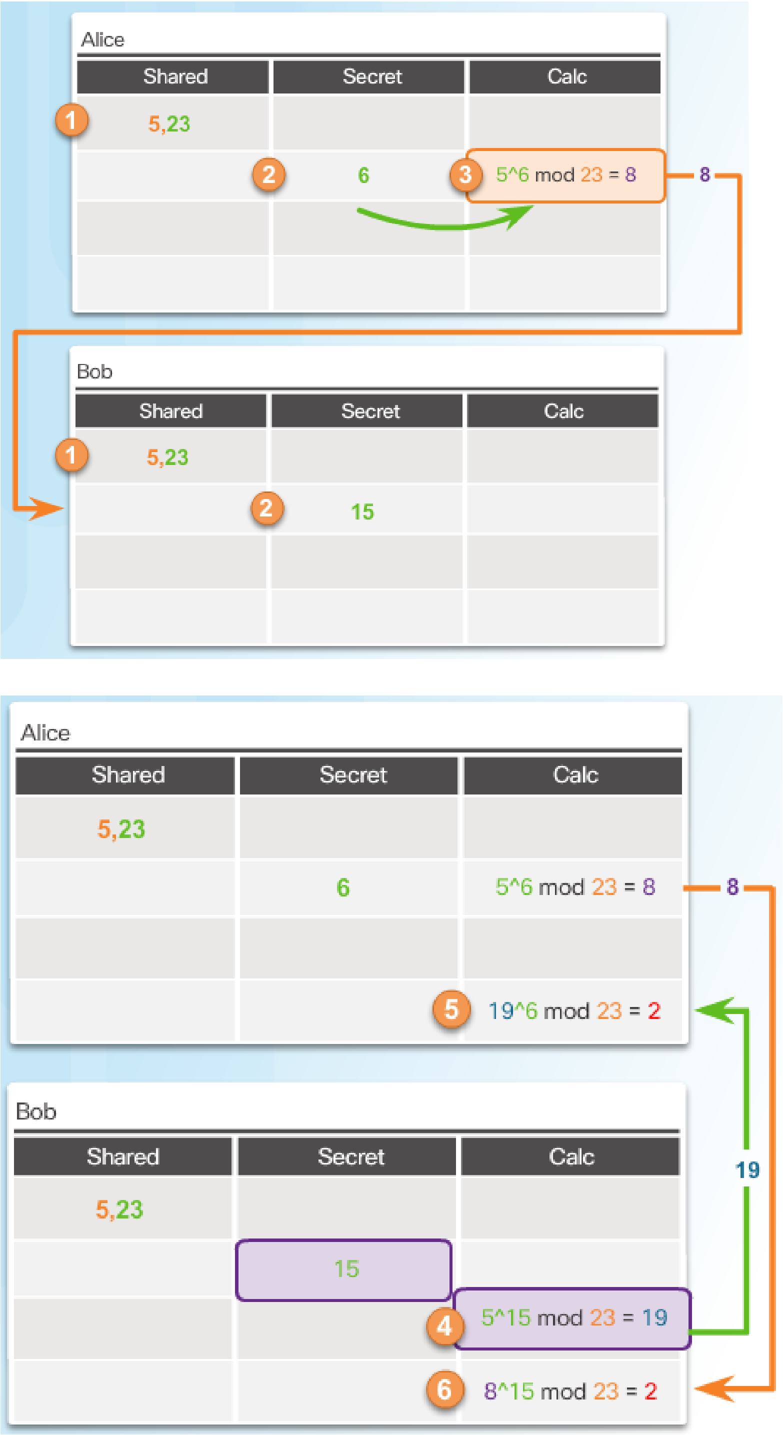


Figure 2: "Example on Diffie-Helman operation"

6.1: Technical Operation Example:



Alice

Shared	Secret	Calc
5,23		
	6	5^6 mod 23 = 8

Bob

Shared	Secret	Calc
5,23		
	15	

Alice

Shared	Secret	Calc
5,23		
	6	5^6 mod 23 = 8
		19^6 mod 23 = 2

Bob

Shared	Secret	Calc
5,23		
	15	
		5^15 mod 23 = 19
		8^15 mod 23 = 2

Figure 3: Technical operation Example

7. Establishing a shared key between multiple parties

The Diffie-Hellman key exchange can be used to set up a shared key with multiple participants.

The Diffie-Hellman key exchange, some parts of the data are being sent across insecure channels, but not enough for an attacker to be able to know the shared secret.

The standard Diffie-Hellman key exchange algorithm (or family of algorithms) works in an cyclic group with generator g, and relies on

$$y_A^{x_B}=(g^{x_A})^{x_B}=(g^{x_B})^{x_A}=y_B^{x_A},$$

where y_A and y_B are publicly transmitted, while x_A and x_B remain private.

With three parties, we still have

$$((g^{x_A})^{x_B})^{x_C}=((g^{x_A})^{x_C})^{x_B}=((g^{x_B})^{x_A})^{x_C}=((g^{x_B})^{x_C})^{x_A}=((g^{x_C})^{x_B})^{x_A}=((g^{x_C})^{x_A})^{x_B}.$$

As each party wants to let its own key private, each exponentiation needs to be done at different locations, which means some parties must send their second-step results to the other parties.

One possible protocol could be the following:

1. A, B, C each generate their private keys x_A,x_B, x_C
2. A, B, C each calculate $y_A=g^{x_A}$, $y_B=g^{x_B}$, $y_C=g^{x_C}$
3. .
4. A sends y_A to B, B sends y_B to C, C sends y_C
5. to A.
6. A calculates $z_{CA}=y_C^{x_A}$, B calculates $z_{AB}=y_A^{x_B}$, C calculates $z_{BC}=y_B^{x_C}$
7. .
8. A sends z_{CA} to B, B sends z_{AB} to C, C sends z_{BC}
9. to A.
10. A calculates $k_{BCA}=z_{BC}^{x_A}$, B calculates $k_{CAB}=z_{CA}^{x_B}$, C calculates $k_{ABC}=z_{AB}^{x_C}$

The above equality means that the three parties now know a common secret $k_{ABC}=k_{CAB}=k_{BCA}$

This obviously generalizes to more than three parties, but it needs one additional group exponentiation per participant more for each additional participant (i.e. in total n^2 exponentiations).

8. *Why is the Diffie-Hellman key exchange secure?*

The Diffie-Hellman key exchanges relies on one-way functions as the basis for its security.

These are calculations which are simple to do one way, but much more difficult to calculate in reverse.

9. *Authentication & the Diffie-Hellman key exchange*

In the real world, the Diffie-Hellman key exchanges is rarely used by itself.

The main reason behind this is that **it provide no authentication, which leave users vulnerable to man-in-the-middle attacks**.

These attacks can takes place when the Diffie-Hellman key exchanges is implemente by itself, because **it has no means of verifying whether the other party in a connection is really who they say they are**.

Without any form of authentication, **users may be in connect with attackers** when they think they are communicating with a trusted party.

For this reason, the Diffie-Hellman key exchanges is generally implemented alongside some means of authentication.

This often involve using digital certificates and a public-key algorithm, such as RSA, to verify the identity of each party.

10. *Variations of the Diffie-Hellman key exchange*

The Diffie-Hellman key exchanges can be implemente in several different ways.

It also provide the basis for several other algorithms.

Some of these implementations provides authorization, while others have various cryptographic features such as perfect forward secrecy.

9.1 Example of a protocol used with Deffie- Hellman:

TLS:

TLS, which is a protocol that is used to secures much of the internet.

Use the Diffie-Hellman exchange in three different ways:

- **Anonymous Diffie-Hellman** – This version of the Diffie-Hellman key exchange does not use any authentication, leaves it vulnerable to man-in-the-middle attacks. It should not be implemented

- **Static Diffie-Hellman** – Static Diffie-Hellman uses certificate to authenticate the server. It does not authenticate the client by default, nor does it provide forward secrecy.

- **Ephemeral Diffie-Hellman** – This is considered the most secure implementation because it providesperfect forward secrecy. It is generally combine with an algorithm such as DSA or RSA to authenticate one or both of the partie in the connection. Ephemeral Diffie-Hellman uses different key pairs each time the protocol is run. This give the connection perfect forward secrecy, because even if a key is compromised in the future, it can’t be used to decrypts all of the past messages.

  In practice, only ephemeral Diffie-Hellman should be implemented, because the other options have security issues.

References:

https://crypto.stackexchange.com/questions/2867/whats-the-fundamental-difference-between-diffie-hellman-and-rsa

http://manpages.ubuntu.com/manpages/xenial/man3/diffiehellman.3bobcat.html

https://www.linuxjournal.com/content/diffie-hellman-key-exchange

https://doubleoctopus.com/security-wiki/encryption-and-cryptography/diffie-hellman-algorithm/

https://www.comparitech.com/blog/information-security/diffie-hellman-key-exchange/

https://crypto.stackexchange.com/questions/1025/can-one-generalize-the-diffie-hellman-key-exchange-to-three-or-more-parties/1034

Cisco Cybersecurity netcad course.