Exploring Cryptographic Attacks

Abstract

Cryptographic systems are important in today's Internet age. They can ensure the confidentiality, integrity, and availability of digital communication and data storage. However, they are not immune to attacks, especially as these attacks continue to evolve in sophistication and effectiveness.

This article provides a brief introduction to the mechanics, targets, and potential consequences of different types of cryptographic attacks. Examples are also provided for each type of attack. Some examples show how attacks work in practice, while others demonstrate the impact of attacks. It covers rainbow table attacks, credential stuffing attacks, man-in-the-middle attacks, meet-in-the-middle attacks, frequency analysis attacks, padding oracle attacks, side-channel attacks, and birthday attacks.

Additionally, the study provides possible defenses against these attacks, including implementing multi-factor authentication, enhancing algorithms, hardening protocols, and maintaining strong key management procedures.

This work seeks to raise awareness of the difficulties facing modern digital systems and the significance of enhancing their security while laying the groundwork for future, in-depth research.

Keywords: attacks, targets, advantages and disadvantages, potential consequences, possible preventative measures

Introduction

Information security has become critical in the digital age. It is more important than ever to safeguard sensitive data from unwanted access as we depend more and more on digital networks for everything from personal communication to international financial transactions. The foundation of digital security is cryptographic systems, which are created to protect the availability, confidentiality, and integrity of data. These systems are not immune to attacks though. Attackers constantly create complex ways to get around cryptographic barriers thanks to technological breakthroughs, which puts the security of digital data at serious risk.

In-depth analysis of the many forms of attacks that threaten the security of cryptographic systems is provided by this project, "Exploring Cryptographic Attacks," which explores the realm of cryptographic weaknesses. This project offers a thorough explanation of each attack mechanism, its possible targets, and the consequences of successful assaults. From the well-known Rainbow Table and Credential Stuffing attacks to the Man-in-the-Middle and Meet-in-the-Middle Attacks. The project explains how these attacks work. It also provides real life examples from actual situations on how these attacks work in real-world scenarios.

Not only does the project provide information about these attacks but also potential ways on how to deter these attacks from happening may be improving the algorithm itself by using additional standards. The project's objectives are to provide readers with a basic awareness of the difficulties presented by cryptographic assaults and to emphasize the need of taking proactive steps to strengthen digital security infrastructure.

"Exploring Cryptographic Attacks" essentially provides information and a tale that highlights the persistent nature of digital dangers and provides information on measures that may be used to reduce these risks. Safeguarding the digital environment requires an awareness of the dynamics of cryptographic assaults and countermeasures, which we must acquire as we negotiate the complexity of digital security.

Discussion

1 Common Cryptographic Attacks

1.1 Rainbow Table Attacks

1.1.1 What Is Rainbow Table Attacks

Rainbow table attacks is a method of password cracking that involves precomputing many possible password hash values and their corresponding plaintext passwords, then comparing them to the password hash values stored in the target database (Rainbow Table Attack, n.d.). These precomputed hash values and password pairs are stored in a table known as a 'rainbow table.' When attackers obtain the password hash values from the database, they can quickly search and compare them against the rainbow table to find the corresponding plaintext passwords.

1.1.2 How Does Rainbow Table Attack Work

1.1.2.1 Steps of Rainbow Table Attack

The principle of rainbow table attack relies on leveraging precomputed rainbow tables to expedite the password cracking process. The steps involved in this attack method include (Beschokov, n.d.) (Jagannath, 2023):

1. Build a rainbow table: The attacker first calculates many potential plaintext passwords and their corresponding hash values based on specific rules and algorithms. These pairs are stored in the rainbow table. The correspondence between plaintext passwords and hash values can be organized using linked lists or other data structures.
2. Obtaining the target hash value: Attackers acquire the hash value of the target password stored in a database through means such as database leaks or other security vulnerabilities.
3. Matching search: Attackers compare the hash value obtained from the database with the hash values stored in the rainbow table. If a matching hash value is found, the corresponding plaintext password can be retrieved by searching for it. If it does not exist, reduce the hash value to another plaintext and hash the new plaintext. Check whether the newly obtained hash value is in the rainbow table and repeat this process until a matching hash value is found in the rainbow table or no matching hash value is found when the specified number of loops ends.

1.1.2.2 How to create a rainbow table

A rainbow table can be created according to the following steps (Jagannath, 2023):

1. Select plaintext: Depending on the situation, select plaintext, such as "password6677".
2. Hash and Reduce: Hash this plaintext and then apply a "reduction function" to get another plaintext. Continue hashing and reducing this newly obtained plaintext and repeat this process multiple times to form a chain.
3. End of chain: The process stops when a predefined number of loops is reached. Store the first and last value of the chain, i.e., store the plaintext and the last hash.

1.1.3 Types of rainbow tables

Rainbow tables can be classified according to their attributes and objectives. Here are several categories of rainbow tables (Makhene, 2024):

1. Standard Rainbow Tables: These are the most fundamental form of rainbow tables. They consist of precomputed chains of plaintext-password and corresponding hash value pairs. Each chain starts with a plaintext-password, undergoes hashing to obtain a hash value, then is transformed into another plaintext-password through a reduction algorithm, followed by hashing again, and so forth, until reaching a predetermined length.
2. Time-memory trade-off (TMTO) tables: These are optimized standard rainbow tables. They balance the time required to calculate the tables with the memory required to store the tables. These tables use less memory than standard rainbow tables but require more computing time during the cracking process.
3. Cryptanalysis-oriented tables: These tables are tailored to specific cryptographic algorithms and scenarios, and they are designed to optimize the process of reversing

cryptographic hash functions. They may exploit known patterns or vulnerabilities in hashing algorithms to create more efficient rainbow tables.

4. Specialized tables for hash algorithms: These types of rainbow tables are typically customized for specific hash functions, such as MD5, SHA-1, SHA-256, etc. Each type of table is optimized based on the characteristics of the chosen hash algorithm.
5. Alphanumeric tables: Some rainbow tables focus on alphanumeric characters, covering subsets of possible passwords. They are designed to crack passwords containing only letters and numbers (excluding symbols).
6. Case-sensitive tables: These are the type of rainbow tables that consider the case sensitivity of passwords. These tables cover variations in letter case, allowing attackers to crack passwords that may contain combinations of uppercase and lowercase letters.
7. Application-specific tables: This type of rainbow table is one created by an attacker based on the unique characteristics and requirements of certain applications or systems. This targeted approach can increase the effectiveness of attacks.
8. Multilingual tables: These rainbow tables are intended for systems that allow passwords in multiple languages or character sets. They can cover a wider range of characters beyond the standard alphanumeric set.

1.1.4 Advantages and Disadvantages

Advantages:

1. Efficiency: Rainbow tables can significantly improve the speed of password cracking because they precompute many possible password hash values and their corresponding plaintext passwords, making the lookup process extremely fast.
2. Resource saving: Once a rainbow table is created, it can be reused without the need to recalculate the hash values and plaintext password mappings each time, thus saving computational resources during the cracking process.
3. Scalability: Attackers can create rainbow tables of varied sizes and types as needed to accommodate various cracking scenarios and requirements.

Disadvantages:

1. Space Requirement: Rainbow tables require a significant amount of storage space to store precomputed hash values and password mappings, especially for long passwords and strong hash algorithms, the size of rainbow tables can be extremely large.
2. Memory Consumption: During the cracking process, the entire rainbow table needs to be loaded into memory for lookup, which may consume a considerable amount of memory resources, especially for large rainbow tables.
3. Impact on Salt Values: Salt values are a technique used to increase the strength of password hash values, but rainbow table attacks have limited effectiveness against salt values because they require precomputing rainbow tables for each salt value, increasing the cost and complexity for attackers.

1.1.5 Common Targets

Common targets of rainbow table attacks are systems that use cryptographic hash functions to store password data. These include (Swisher, 2024):

1. Systems with unsalted hashes: Systems that store hashed passwords without a unique salt (random data added to them before hashing) for each password are particularly vulnerable.
2. Legacy Systems: Older systems using outdated cryptographic hash functions like MD5, or SHA-1 are more susceptible to these attacks.
3. Systems with no two-factor authentication (2FA): Systems that use only passwords for authentication are more vulnerable to rainbow table attacks because the only obstacle is the password, and the password can be cracked through the rainbow table.

1.1.6 Examples

1.1.6.1 Example 1

Suppose there is a website with insufficient security measures and its database is easily accessible. This website uses a hash encryption algorithm to convert the password entered by the user into a fixed-length hash value and store it in the database. When an attacker obtains the website's user database (which contains hashed passwords), they compare the hashed passwords in the database with the hash values in the rainbow table to find matching plaintext passwords.

Through this method, an attacker can quickly crack the password and gain access to the user's account. Once an attacker successfully logs into a user account, they can perform unauthorized actions such as stealing personal information, tampering with data, or even perform other malicious activities.

1.1.6.2 Examples 2

If a hacker successfully breaks into a company's network and gains access to Active Directory. Through the vulnerability, he was able to retrieve hashes of user passwords stored in Active Directory without directly obtaining the clear text password. The hacker then compared the obtained hash value with the hash value of a rainbow table he created or obtained (which contains the hash values of a large number of common passwords and their corresponding plaintext passwords) to find the hash value that matches the password in Active Directory. The hashed password matches the clear text password.

Once a matching plaintext password is found, the hacker can use these credentials to log into the user's account, penetrate further into the company's network and obtain more sensitive information, or perform other malicious activities. Such attacks can lead to corporate data breaches, user privacy breaches, and damage to the company's reputation.

1.1.6.3 Real Examples:

Here are two real examples of successful rainbow table attacks (Shah, 2023):

1. LinkedIn: In 2012, hackers obtained data from LinkedIn’s database containing more than 6.5 million hashed passwords. They then used rainbow tables to crack a large number of passwords and leaked them online, exposing millions of user accounts.

2. Ubuntu Forum: In 2013, hackers gained access to a database containing more than 1.8 million usernames and their corresponding hashed passwords by attacking the Ubuntu Forum network. They then used brute force and rainbow tables to crack passwords and successfully gained access to a large number of user accounts.

1.2 Credential Stuffing Attacks

1.2.1 What Are Credential Stuffing Attacks

Credential stuffing is a type of cyberattack where attackers use large sets of stolen usernames or email addresses and the corresponding passwords from one website to try to gain unauthorized access to user accounts on another website (Credential stuffing, n.d.).

1.2.2 How Does Credential Stuffing Attack Work

Credential stuffing attack involves the following steps (Lenaerts-Bergmans, 2023):

1. Collecting credentials: The attacker first obtains credentials (username and password combination) that have been compromised. These credentials may have been obtained from previous data breaches, such as those exposed in a hack or website data breach.
2. Automated attempts: Attackers use automated scripts or tools to attempt to log into multiple target websites simultaneously using obtained credentials. These scripts can simulate normal user login behavior to avoid triggering security tools that may block external or unusual addresses.
3. Check login results: The attacker monitors the results of each attempt. If the login attempt is successful, meaning the username and password combination entered is valid, the attacker can collect additional information such as personal data, stored credit card information or bank details.

1.2.3 Advantages and Disadvantages

Advantages:

1. Ease of execution: Credential stuffing attacks are easy to execute compared to other forms of cyberattacks because attackers do not require complex technical skills or expertise to carry out such attacks.
2. Fast speed: The application of automated tools significantly improves the attack efficiency and speed of credential stuffing attacks.
3. Low cost: The cost is relatively low since attackers do not need to develop custom malware or exploit system vulnerabilities.
4. Wide range of applicability: On the one hand, many websites and online services use the same authentication mechanism; on the other hand, many users tend to reuse the same username and password on multiple websites, so credential stuffing attacks may be obtained on multiple targets success.
5. Difficult to trace: Because credential stuffing attacks are often carried out through automated tools, attackers can hide their true identity and location, making it harder to trace their actions.

Disadvantages:

1. Limited success rate: Although many users reuse the same credentials, a certain percentage of users employ unique credentials or other security measures, reducing the success rate of credential stuffing attacks.
2. Countermeasures: Many websites and service providers have taken some measures to prevent credential stuffing attacks, such as using multi-factor authentication, implementing account locking mechanisms, etc. These measures may reduce the success rate of attacks.

1.2.4 Common Targets

Almost any network system that stores users' personal information, account credentials, or other sensitive data can be the target of a credential stuffing attack. Some of the most common attack targets are listed below (What is Credential Stuffing, n.d.):

1. Online services and websites: including social media, email service providers, e-commerce platforms, bank, and financial institution websites, etc. These websites often store users' personal information, account credentials, and payment information, making them targets.
2. Large organizations and enterprises: Network systems of large enterprises and organizations often contain large numbers of user accounts and sensitive data, making them targets for attackers to obtain business secrets or personal information.
3. Government Agencies: Government agencies store a large amount of sensitive information on their websites and databases, including citizens' personal data, tax information, and government secrets, and therefore become one of the targets of credential stuffing attacks.
4. Educational Institutions: Schools, universities, and other educational institutions also have large amounts of personal information and credentials within their student and employee databases, making them potential targets for attacks.
5. Healthcare organizations: Hospitals, clinics, and other healthcare organizations have large amounts of personal patient information and medical records stored in their databases, making them targets for attacks to obtain sensitive medical information or for ransom.

1.2.5 Examples

1.2.5.1 Uber

Uber suffered a data breach caused by a credential-stuffing attack in 2016 (Lake, 2024). An investigation by the UK Information Commissioner's Office (ICO) found that attackers gained access to Uber's data stores through credential stuffing. First, the attackers obtained the credentials of an Uber employee exposed on other websites and used them to access his GitHub account. After gaining access to the account, the attacker discovered the login details for the Amazon Web Service S3 bucket where Uber data was stored. This allowed them to steal the data of 57 million Uber users, including drivers and passengers. An investigation by the UK Information Commissioner's Office (ICO) found that attackers gained access to Uber's data stores through credential stuffing. First, the attackers obtained the credentials of an Uber employee exposed on other websites and used them to access his GitHub account. After gaining access to the account, the attacker discovered the login details for the Amazon Web

Service S3 bucket where Uber data was stored. This allowed them to steal the data of 57 million Uber users, including drivers and passengers.

The attackers then contacted Uber and demanded $100,000 for information on how they accessed the S3 bucket. Uber paid the fee, which makes the payment appear to be part of their bug bounty program, but Uber has not made this completely public. It was not until about a year later, in late 2017, that the company finally disclosed the breach's details. The hack and resulting cover-up resulted in Uber being punished for several reasons—poor security, delayed notifications, and more.

1.2.5.2 Reddit

In early 2019, Reddit locked out some users' accounts over suspected credential stuffing attacks (Lake, 2024).

Reddit's security team has monitored unusual activity from "a large group of accounts," which they believe is likely caused by credential stuffing. To prevent these accounts from being taken over, it locks out users and forces them to reset their passwords before their accounts can be restored.

Some affected users commented that they were using unique and strong passwords as well as two-factor authentication. Especially because with two-factor authentication in place, their accounts should theoretically not be compromised by a credential-stuffing attack. Therefore, there is some speculation that the website has some account security issues of its own and is using credential stuffing as an excuse to blame users. Another explanation is that Reddit may have decided to lock and reset the passwords of any account that experienced suspicious login attempts, not just those that were successful. If this is the case, employing two-factor authentication or a strong and unique password will be irrelevant.

1.3 Man in the Middle Attack

1.3.1 What is Man in the Middle Attack

Man in the middle attack should not be confused with meet in the middle attack, both are different types of attacks. In man in the middle-attack, the attacker eavesdrops on the conversations between two parties and the attacker might or might not alter the conversations and data between the parties. But this is not known to either of the parties and the attacker tries to gain and access as much information as possible, posing a serious security risk to both the system and the parties. Sometimes the attacker can also get unauthorized access to the system. There are three main purposes that an attacker uses the man in the middle attack, they are:

- Eavesdropping: It is done to gather information from the messages that the parties share. This might be some highly confidential information that should not be public.
- Manipulating the data: The attacker might manipulate and then send the manipulated data to the other party. This is falsification of information, and the attacker can inject malware to the receiver.
- Impersonation: The attacker can impersonate and gain unauthorized access to confidential information.

Man in the middle attack can be done in several ways like:

- Insecure Wi-Fi networks: If any user connects to Wi-Fi networks that were set up by attackers, that will easily compromise the data of the user who connected to the

network. This way the attacker can gain access to all the traffic that is generated by the user and can be rerouted to the adversary's system. The adversary can then see and use this sensitive information.

- DNS Spoofing: If the DNS cache of a user's device is changed and corrupted, redirecting the flow of traffic to the adversary's web pages, it might lead to fraud and theft because the connection between the user and the server is not secure anymore and is rather exposed to the adversary.
- Arp Spoofing: An attacker can connect their MAC address to the IP address of another host, such as the gateway, and use that link to intercept all communications sent and received by that host by sending fictitious Address Resolution Protocol (ARP) packets via a local area network (LAN).
- SSL Stripping: Converting an insecure HTTP connection from HTTPS to HTTP, which enables the attacker to view unencrypted traffic because the connection is not secure anymore.

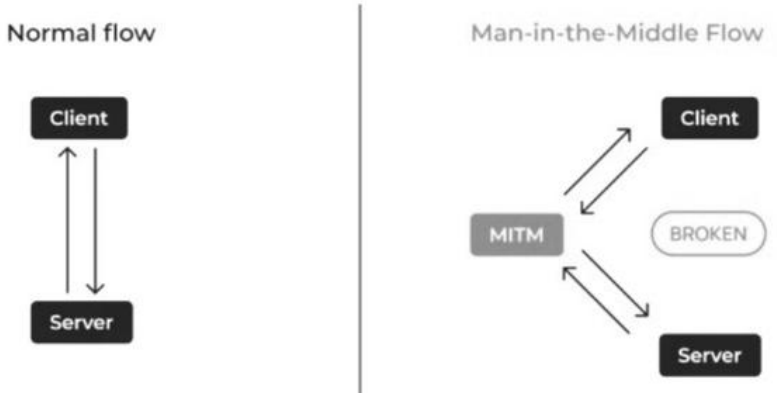

Fig 1.3.1.1 Representation of man in the middle attack

1.3.2 Advantages and Disadvantages of Man in the middle attack

Advantages:

- Makes it possible for an attacker to secretly listen in on and overhear conversations between two parties.
- Gives the attacker the ability to change the data that has been intercepted before sending it to the intended recipient, such as login passwords or financial information.
- May be executed remotely; no direct physical entry to the victim's hardware or network is required.
- Takes advantage of holes in security protocols, such as unpatched vulnerabilities, poor authentication, and unprotected wireless networks.

Disadvantages:

- Modern Technologies: Many modern technologies are developed in such a way that they can detect and prevent man in the middle attacks from happening. Like HTTPS

where the S stands for secure and SSL protocols where it is hard to decrypt the message.

- Countermeasures by Users: MitM (Man in the middle) attacks are less successful when simple user-level countermeasures are used, such as utilizing VPN services, checking the security certifications of websites, and staying away from dubious networks.
- Specific requirement: Man in the middle attack needs the attacker to somehow enter the conversation between two parties, which might be a difficult to do in many cases.
- Can be detected: Sometimes some traces can be left by the attacker like irregular traffic flow, etc., that may alert the parties. This might lead to a detection of the attack.

1.3.3 Common Targets

Though man in the middle attack targets many diverse types of organizations, some of the common ones are:

- Email accounts: Through social engineering or other techniques, MITM attackers can breach email accounts, after which they can intercept and alter correspondence between the victim and other parties.
- Financial institutions and e-commerce websites: One of the main reasons for man in the middle attack is to steal personal and financial information on websites and banking platforms. This data theft on the login credentials like usernames and passwords of the users or can also be sensitive confidential information like credit card details to commit fraud.
- Internet of Things (IOT): IoT devices are susceptible to MITM attacks since they frequently have internet connections and might have some form of security that is weak. An attacker can use an IOT device and disrupt the device.
- Government communications: Government communications can also be targeted by adversaries by using man in the middle attack. This can lead to the leak of secret information that should not be accessed by anyone else or made public.

1.3.4 Real world examples

Some of the real-world examples where man in the middle attack is used to commit cyber-crime are:

- Vulnerability in Equifax: Equifax, a credit reporting agency was once the target of this attack, where a data breach occurred that exposed that data of nearly 150 million Americans. This happened in the year 2017.
  The reason for this attack was due to the fact that Equifax mobile application was not always using HTTPS connection to secure its data. This made the attackers gain access to the information.
- DigiNotar Breach: An assault on the Dutch digital certificate authority DigiNotar in 2011 resulted in the acquisition of over 500 certificates for well-known websites like Google and Skype by the attacker. Then the attacker used man in the middle attack by posing themselves as legit by using the stolen certificates. This allowed the attacker to access confidential information like browsing history and information of the users.

- Belgacom Hack: It was disclosed in 2013 that GCHQ, the British government's communications center, had targeted Belgacom, a Belgian telecoms operator that is now Proximus, as part of an operation known as "Operation Socialist". The assault, which was a large-scale Man in the middle attack strategy, which involved in inserting malware into Belgacom's network with the intention of spying on conversations.
- Superfish Adware: This happened in 2015 where Lenovo computers were already pre-installed with Adware called “Superfish Visual Search”. This adware installed ads on the traffic of the users. This allowed Superfish to do man in the middle attacks because Superfish had its own SSL certificates that intercepted these communications because of the advertisements placed in the traffic of the webpages.

1.4 Meet in the Middle Attack

1.4.1 What is Meet in the Middle Attack

Meet in the middle attack is a type of known plain text attack that involves reducing the number of permutations/iterations needed to decrypt the message. It uses the plain text block and a cipher text block in an encryption scheme using multiple keys. Instead of using every possibility like a brute force attack, meet in the middle attack from either end of the encryption or decryption process. So, the entire process is broken down into two halves and the resulting intermediate result is compared through tables. Meet in the middle attack is especially effective for block ciphers that are symmetric like double DES. We will further talk about why this is the case in this report.

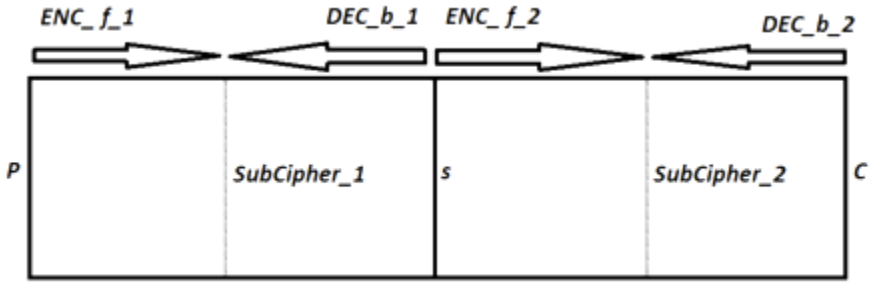

Fig 1.4.1.1 How meet in the middle attack works

1.4.2 How does meet in the middle attack work

Let us take double DES for example and see how meet in the middle attack works:

- In double DES, there are two keys K_1 and K_2 and the cipher text be C and the plaintext be P.
- Firstly, the attacker uses the plaintext P and encrypts it on the first DES and keeps the results in a table. This encryption is done by using all the possible types of keys for key K_1. The resulting encrypted text is C_{K1}.
- Then, the attacker decrypts the cipher-text C by using all possible values for key K_2 and then stores this intermediate results in a table too. Let this be C'_{K2}.

- Now, the attacker will compare these intermediate results that C_{K1} and C'_{K2}. If the attacker finds a match, it means that the corresponding keys used are the keys used to encrypt the message and those are the actual keys K_1 and K_2.

Even though this is how a meet in the middle attack works, sometimes there might be false alarms for the intermediate texts and the keys used in the actual encryption process might be different from what the attacker got because of false alarms. False alarms are nothing but false positives, where $C_{K1} = C'_{K2}$ but it really is not the case.

To further reduce the chances of false alarms, more plaintext and cipher text pairs can be selected by the attacker. With every additional plaintext and cipher text pair, the chances of false alarms are decreased.

1.4.3 Advantages and Disadvantages

Advantages:

- It reduces the number of permutations required to decrypt a message and is better than regular brute-force.
- As the attacker splits the whole encryption process into two parts, the number of operations is nearly halved from 2^{2n} to $2*2^{n}$.
- The two parts of the process can be calculated separately, so they can both be computed parallel-ly at the same time for faster execution.
- Meet in the middle attack can be used effectively on algorithms using double encryptions.

Disadvantages:

- Firstly, the meet in the middle attack algorithm has a very specific use case and is only applicable to double encryptions with multiple keys.
- It uses high computational resources because different keys and pairs should be stored in tables and there are often multiple tables generated.
- Meet in the middle attack can be prevented by using another layer of the algorithm, for example in the case of DES, triple DES can be used to prevent this.
- The attack generated false positives and to prevent this, further computation resources are needed to perform the attack.
- As the meet in the middle attack uses more resources to compute, if an attacker is trying to attack a system, the attack might get flagged because of the large number of operations performed on it by the attacker.

1.4.4 Developments on meet in the middle attack

As a simple meet in the middle attack might not work because of the current encryption standards (for example it does not work for triple DES), there is a better version of the meet in the middle attack which is the multi-dimensional meet in the middle attack.

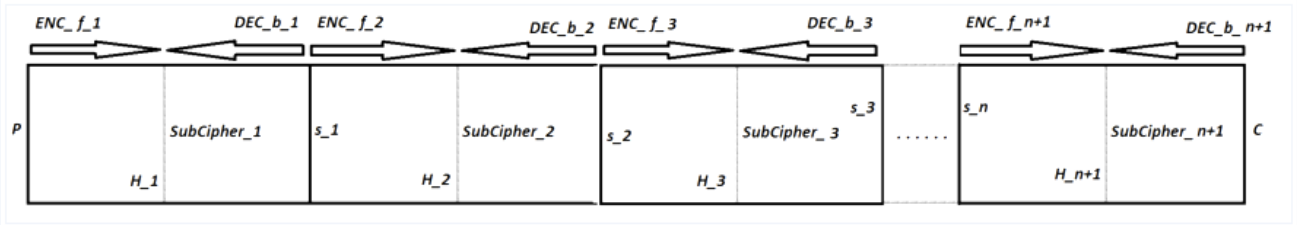

Fig 1.4.4.1 How a multi-dimensional meet in the middle attack works

Multi-dimensional meet-in-the-middle attack is a better version of the normal middle attack which can be used for complex encryption algorithms. There are multiple intermediate steps for a multi-dimensional meet-in-the-middle attack. The intermediate values should match in all of the multiple layers for the multi-dimensional-meet in the middle attack to work. There is also an increased chance of false alarms because of higher dimensions. The complexity of this attack increases as more layers are added to it. The attacker must discover a path across a highly multidimensional space that matches the right combination of the order of operations, or the keys used in the process.

Meet in the middle attacks are mostly done in theory because performing these types of attacks in the real world require a lot of computational power which might involve calculations and require some space to store all the intermediate tables that might be generated by this attack. They can be used to evaluate how secure the security systems are and to see if they are vulnerable to this attack and can help in improving the security of the system.

1.4.5 Examples

Though there are hardly any cases where meet in the middle attack was used to compromise an algorithm. New systems and encryption standards are developed in such a way to prevent the attack from happening in the first place and their security standard is also tested before using it. The reason triple DES exists may be due to the vulnerability of double DES. OBJ

1.5 Frequency Analysis Attack

1.5.1 What are Frequency Analysis Attacks?

Frequency analysis is a pivotal technique used in cryptanalysis to crack substitution ciphers. This involves observing the frequency of character letters or letter clusters within the ciphertext and comparing them to the forecasted frequencies within the target language. This technique becomes notably powerful towards classical ciphers just like the Caesar cipher, wherein every plaintext letter is shifted a set wide variety of positions up or down the alphabet. These ciphers are liable to frequency evaluation because they preserve the authentic plaintext's structure and letter patterns.

Consider the figure below, it serves as a foundation for understanding and implementing frequency analysis techniques.

Letter	Probability	Letter	Probability
A	0.082	N	0.067
B	0.015	O	0.075
C	0.028	P	0.019
D	0.043	Q	0.001
E	0.127	R	0.060
F	0.022	S	0.063
G	0.020	T	0.091
H	0.061	U	0.028
I	0.070	V	0.010
J	0.002	W	0.023
K	0.008	X	0.001
L	0.040	Y	0.020
M	0.024	Z	0.001

Fig.1.5.1 The frequencies of letters appearing in the English language, in order from most common to least.

1.5.2 Types of Frequency Analysis Attacks

There are different types of frequency analysis attacks, and their execution is based on factors like the length of the ciphertext, the encryption method used, and the characteristics of the target language:

1. Ciphertext-Only Attack: This attack is considered to be the most challenging as the attacker can access a ciphertext, but no additional information is required. Eventually, the attacker observes patterns and conjectures regarding the plaintext and key by examining the frequency distribution of individual letters or character clusters in the ciphertext.
2. Known-Plaintext Attack: In this attack, the attacker can pick up certain pairs of plaintexts and ciphertexts. Then by comparing and analyzing the frequency distributions, the attacker can gather details about the encryption technique or key.
3. Chosen-Plaintext Attack: Here, for any chosen plaintext, the attacker has access to the ciphertexts. The attacker can interpret the encryption process and retrieve the key by thoroughly selecting plaintexts and analyzing their corresponding ciphertexts.
4. Probable-Word Attack: In this attack, the attacker looks for common names, words, or phrases contained in ciphertext and analyses linguistic patterns and predicted frequency. This might provide hints to the attacker regarding the plaintext or key.

1.5.3 Working of Frequency Analysis attacks

The operational mechanics of Frequency Analysis Attack involves the following steps:

1. Firstly, the attacker calculates the frequency of each letter or symbol in the ciphertext.
2. Secondly, he/she identifies the most frequent letters and assumes that they match common plaintext letters.
3. Using this information, the attacker substitutes words and looks for recognizable patterns or well-known phrases.
4. The attacker repeats the above process, making more substitutions as more information is uncovered.

As the letter frequencies in longer ciphertexts will more closely resemble the forecasted frequencies of the language, they are easily cracked. Comparatively, shorter messages can be difficult comparatively, but enough information can be obtained for a successful attack by examining several ciphertexts that have been encrypted using the same key.

1.5.4 Advantages and Disadvantages of Frequency Analysis Attacks

Advantages:

- As frequency analysis is a ciphertext-only attack, all that an attacker needs are access to the ciphertext. Due to this, it is considered as an effective strategy when the key is unknown.
- It may provide hints about the length of the key or type of cipher being used, which can assist future cryptanalysis attempts.
- The ability to automate frequency analysis with the capability of current computing facilitates the efficient examination of large volumes of ciphertext to find patterns.
- Simple substitution ciphers are extremely susceptible to this attack because they rely on the patterns of the plaintext language compared to advanced ciphers that are meant to flatten the frequency distribution.

Disadvantages:

- It is ineffective against more advanced ciphers like AES because they are meant to flatten the frequency distribution.
- Shorter messages do not provide sufficient information for a reliable frequency analysis; hence they might make an attack less successful
- It is computationally difficult and time-consuming for polyalphabetic ciphers with large keys, especially without today's computing power.
- An unfamiliar language makes the attack much more challenging. Hence, the attacker must be aware of the expected letter frequency patterns in the plaintext language.

1.5.5 Common Targets

The common targets of Frequency Analysis Attacks are:

1. Communications Using Classical Ciphers: The most common targets are the messages that are encrypted using classical ciphers, such as the Caesar cipher, basic substitution ciphers, or the Vigenère cipher. Before advanced encryption techniques were developed, these were used in earlier diplomatic, military, and private correspondence.
2. Large sample ciphertexts: Longer ciphertexts offer more information for precise frequency analysis, making them more valuable targets when compared to relatively brief encrypted communications. The higher the sample size the more closely the letter frequencies will resemble the anticipated linguistic patterns.
3. Natural language text encryption: Frequency analysis works best when the plaintext adheres to the standard letter frequency patterns of a natural language, such as English.

1.5.6 Examples

1.5.6.1 The Cracking of the Enigma Code

Although the Enigma machine, which was employed by Nazi Germany in World War II, was not directly susceptible to frequency analysis in its most basic form and was far more complex than a simple substitution cipher, advanced forms of frequency analysis were one of the techniques used by Alan Turing and his team at Bletchley Park. They searched for repeating "cribs" and patterns that would lead to the Enigma machine's settings being figured out, which they could then use to decode communications.

1.5.6.2 The Confederate Cipher Disk

The Confederate Cipher Disk was a device used by the Confederacy to encrypt military communications during the American Civil War. Because it was polyalphabetic, it was more resistant to basic frequency analysis, but it was still prone to bad operating practices (such as not often changing the cipher parameters). These vulnerabilities were exploited by Union cryptanalysts, such as the pioneer of American cryptography, Colonel William F. Friedman.

1.6 Padding Oracle Attack

1.6.1 What is a Padding Oracle Attack?

Padding Oracle Attack uses vulnerabilities in the ciphertext's padding to decrypt the plaintext without needing the encryption key. Here, the attacker makes use of information that has been disclosed by a padding oracle (a system or service that offers commentary on the accuracy of padding in decrypted ciphertext). An attacker can decrypt the plaintext byte by byte by sending constructed ciphertexts and monitoring the oracle's responses. Attacks using padding oracles are especially dangerous to cryptographic systems that employ block ciphers with padding schemes like PKCS#5 or PKCS#7. As seen in the figure below, block ciphers frequently employ CBC decryption, which decrypts each block of ciphertext, examines the padding, eliminates the PKCS#7 padding, and provides the message's plaintext. These attacks highlight the necessity of strong protections against side-channel information leaking and the significance of applying cryptographic algorithms securely.

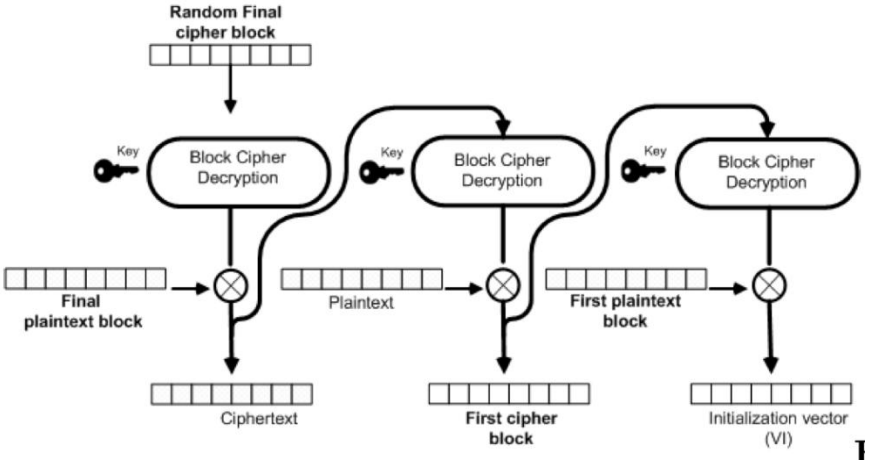

Fig 1.6.2 Padding Oracle Attack on CBC encryption

1.6.2 Types of Padding Oracle Attacks

Padding oracle attacks come in many forms, all of which take advantage of flaws in how cryptographic systems handle padding. The most common types of Padding Oracle Attacks are:

1. Classical Padding Oracle Attack: In this kind of attack, the attacker interacts with a system that encrypts messages using padding. The attacker can deduce the details about plaintext and can successfully decrypt it by submitting modified ciphertexts and observing system responses, such as timing discrepancies or error messages.
2. Plaintext Recovery Attack: In this attack, the attacker uses the encrypted message's ability to inject ciphertext blocks, then examines the system's response. The attacker modifies these injected ciphertext blocks and examines the responses again, thereby retrieving the plaintext byte by byte.
3. Adaptive-Chosen Ciphertext Attack: In this kind of attack, the attacker creates ciphertexts using responses from previous interactions with the system. An attacker can decrypt any ciphertext and obtain private data by modifying the selected ciphertexts in response to feedback from the system.

1.6.3 Working of Padding Oracle Attack

Let us consider a scenario where an attacker decrypts a ciphertext using a padding method that is susceptible to oracle attacks. Blocks of encrypted information make up the ciphertext, with the final block padded to fit the block size. Decrypting this ciphertext and getting back the original plaintext is the attacker's aim. Subsequently, by manipulating the bits in the final block, the attacker makes several variations of the altered ciphertext. The target system receives these altered ciphertexts to be decrypted. As the oracle, the system examines the changed ciphertext and reports on how well the padding in the decrypted plaintext was done. The attacker observes indications from the oracle, such as timing discrepancies or error messages, that indicate whether the padding is legitimate. Through a series of repetitive adjustments to the updated ciphertext in response to the oracle's responses and input analysis, the attacker can progressively infer details about the plaintext until finally decrypting the entire message.

1.6.4 Advantages and Disadvantages of Padding Oracle Attack

Advantages:

- Effective: When used against cryptographic systems that employ weak padding techniques, padding oracle attacks can be quite successful in enabling attackers to decrypt ciphertexts and retrieve plaintext without needing to know the encryption key.
- Real-life Applicability: Padding oracle attacks are applicable to a broad spectrum of cryptographic protocols and systems that use padding, such as symmetric encryption methods like RSA and block ciphers like AES. This makes them flexible in real-world scenarios.
- Stealthy: Since padding oracle attacks usually entails interacting with a system's decryption oracle and taking advantage of responses or error messages to infer

information about the plaintext, they can frequently be carried out without leaving traceable evidence of the attack.

Disadvantages:

- Complexity: Putting a successful padding oracle attack into action can be difficult and calls for a thorough grasp of padding schemes, cryptographic principles, and the weaknesses in the target system. To efficiently create changed ciphertexts and analyze oracle responses, a great deal of trial and error may also be required.
- Resource-intensive: Padding oracle attacks can be resource-intensive, requiring the attacker to repeatedly transmit the decryption oracle modified ciphertexts and iteratively examine the responses. Longer attack times and higher computational costs may arise from this.
- Dependency on Oracle: Oracle padding attacks depend on the availability of a decryption oracle, which offers feedback on the accuracy of the padding in ciphertexts that have been decrypted. The attack might not be successful if the oracle is absent or not adequately guarded.

1.6.5 Common Targets

A wide range of cryptographic methods and systems that make use of weak padding algorithms are susceptible to padding oracle attacks. Typical goals include:

1. Web Applications: Padding oracle attacks can target web applications that use cryptographic techniques to secure sensitive data, including passwords, session tokens, or authentication tokens. Attackers may use vulnerabilities in cryptographic implementations or web application frameworks to decrypt encrypted data sent between clients and servers.
2. Cryptographic Libraries and APIs: If they employ weak padding techniques, cryptographic libraries and APIs that offer encryption and decryption capabilities could be targeted by padding oracle attacks. By taking advantage of weaknesses in these libraries, attackers can recover plaintext that has been encrypted by the library and decrypt ciphertexts.
3. Database and Cloud Storage Systems: If they use weak cryptographic algorithms or padding schemes, cloud storage and database systems that encrypt data while it is in transit or at rest could be subject to padding oracle assaults. Attackers can decrypt encrypted data sent via network connections or stored in the cloud by taking advantage of flaws in these technologies.
4. Implementations of Secure Sockets Layer/Transport Layer Security (SSL/TLS): Network traffic encrypted by SSL/TLS implementations can be decrypted using padding oracle attacks. OpenSSL Heartbleed and other SSL/TLS implementation vulnerabilities have previously been used to perform padding oracle attacks and retrieve private data sent across secure connections.

1.6.6 Examples

1.6.6.1 Microsoft's ASP.NET

The "Cryptographic Padding Oracle Attack" or the "ASP.NET Padding Oracle Vulnerability" is a well-known example of a padding oracle attack that happened in 2009. Due to this vulnerability, attackers could access and alter encrypted authentication cookies used for session management in Microsoft's ASP.NET web application framework. The plaintext contents of encrypted authentication cookies were gradually recovered by attackers by taking advantage of weaknesses in the way ASP.NET handled padding mistakes in decrypted ciphertexts. The possibility of this attack allowing attackers to take control of user sessions, get around authentication procedures, and obtain unauthorized access to private information made it a serious security concern to ASP.NET online applications.

1.6.6.2 Lucky Thirteen Attack

The 2013 Lucky Thirteen attack, which targeted the Transport Layer Security (TLS) protocol implementations used to protect network connections, is another example of a padding oracle attack in real life. The Lucky Thirteen attack recovered unencrypted data sent over encrypted network connections by taking advantage of timing discrepancies in the way TLS implementations handled padding failures in CBC (Cipher Block Chaining) mode. Attackers could use this vulnerability to extract sensitive data, like session tokens or authentication credentials, sent over secure connections by carefully timing differences in the decryption process and examining responses received from TLS servers. This attack demonstrated how crucial it is to put strong cryptographic protocols and safe padding algorithms in place to reduce the possibility of padding oracle attacks in real-world systems.

1.7 Side-channel attack

Side-channel attacks are unique cryptographic attacks since, unlike most attacks that attack the software of a system, attack the hardware of a device to find out specific information. The makeup of computers are transistors and transistors only comprehend two values, true and false. Transistors depend on only three aspects to work correctly which are the voltage, current, and resistance. Making a fixed resistance and knowing the current of the system, different devices are used to note the changes in voltage. In transistors, when a 0 signal or a 1 signal is sent, they have slightly different voltage values and attackers can exploit this knowledge to figure out private keys.

1.7.1 Simple Power Analysis

One of the main side-channel attacks used is called a Simple Power Analysis (SPA), also known as a timing attack. This attack solely looks at electric activity of hardware in cryptographic hardware devices (Randolph, 2020). It also notes the clock cycles within the transistors thus denoting its other name of a timing attack. With this attack you can view, based on electronic voltage readings, which part of the 16-round cycle in DES or 10 round cycle in AES to collect data on the encryption process and notice subtle changes to find a key. Look at image 7.1:

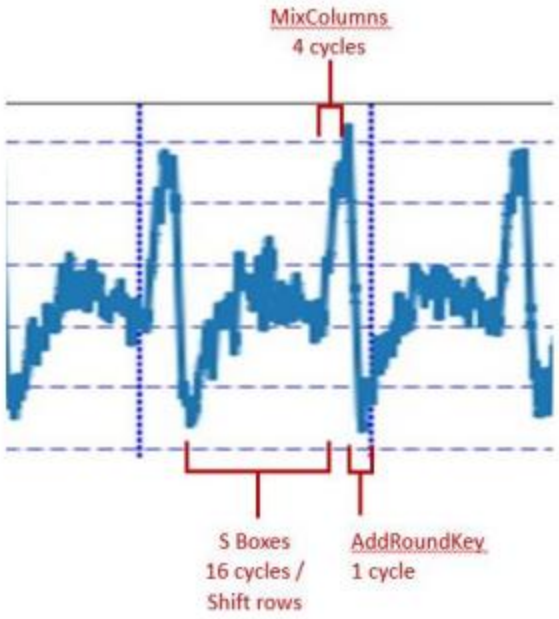

Image 7.1 AES Cycle (Randolph, 2020)

In this image you can see through many tests one could deduce which part of the AES encryption algorithm one is in by how much voltage is being picked up. This image makes sense because the most complicated part of the AES algorithm is mixed column transformation. Also, the added round key in the AES process is the easiest and should be quite close to the bottom every time, especially if the round key is not changing. One of the earliest attacks that used the SPA was published in 1996 by Paul Kocher (Randolph, 2020), who attacked the RSA encryption. With RSA being a relatively slow encryption process to other encryption methods and the public key of someone already being known, an adversary could use a “challenge phrase” to figure out someone’s private key. The encryption process for RSA equals the following equation: $(M^e \ mod \ n) = C$. The decryption process for RSA equals the following equation: $(C^d \ mod \ n) = M$. M in this equation stands for the plaintext, e corresponds to the public key, n is the product of two unknown prime numbers, d is the private key, and C stands for the ciphertext. In this process, there are secret keys being passed to one another that help compute the private key. Figure 7.2 shows how Kocher figured out the secret keys of the RSA process using a SPA and thus being able to retrieve someone’s private key through computation.

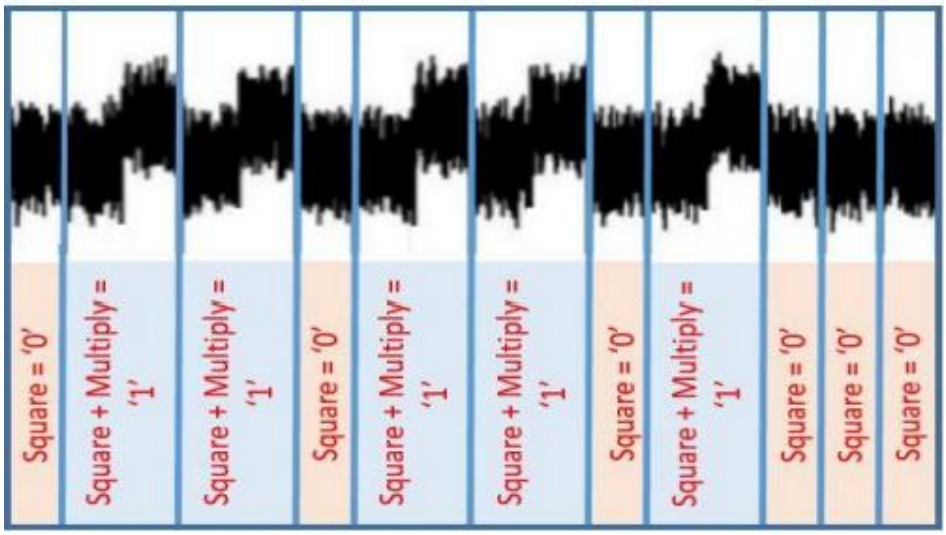

Image 7.2 RSA Secret Key Frequencies (Randolph, 2020)

1.7.2 Differential Power Analysis

Another type of side channel attack is through differential power analysis (DPA). DPA focuses on testing a device's implementation of a cryptographic algorithm rather than SPA that tests an algorithm's mathematical structure (Randolph, 2020). DPA removes the element of interpretation that SPA provides, and Kocher and his colleagues came up with a leakage mode which puts data into two classes. A static object is then chosen to test whether there are matches or not between the two classes, collecting the data on matches to help figure out the master key. To try and find the key, you place two probes on both sides of a resistor and read the voltage drop off. There will be a different amount of voltage drop for correct and incorrect data. Through the Feistel function (Randolph, 2020), one can observe power traces that help find binary values of 1, which go to one class, and 0s to another. In image 7.3, it is very easy to see how DPA clearly gives an adversary the private key without knowing the particular process of DES, AES, or RSA it is in.

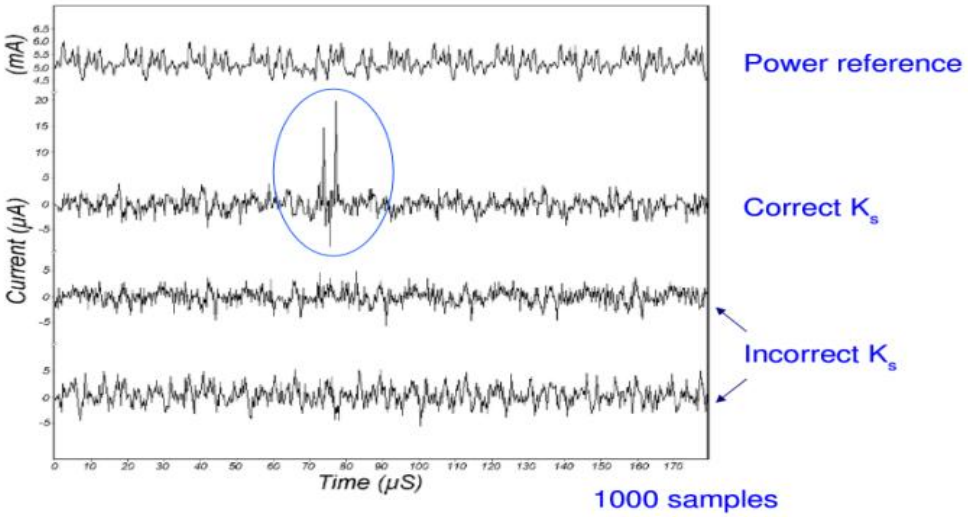

Image 7.3 DPA Example (Randolph, 2020)

As the image displays, a spike equals the correct key. Some countermeasures to take to avoid the correctness of a DPA is to make purposeful desynchronization with fake cycles, delays, and non-uniform clocking, or add a slight amount of electromagnetic radiation to disrupt the signals of the reader or add additional noise to make the jump seem less readable (Randolph, 2020).

1.7.3 Template Attacks

An additional side-channel attack is a template attack. Template attacks use a statistical commonality called probability density functions (PDF), which measures the area underneath the curve of a graph between two points. These attacks collect a small sample size of side-channel attacks to create a mold and use statistical analysis to find the secret key of an encryption process. A template attack follows (all four steps written by Randolph, 2020):

1. “Using a clone of the victim device, use combinations of plaintexts and keys and gather a large number of power traces. Record enough traces to distinguish each subkey’s value.
2. Create a template of the device’s operation. This template will highlight select points of interest” in the traces and derive a multivariate distribution of the power traces for this set of points.
3. Collect a small number of power traces from the victim device.
4. Apply the template to the collected traces. Examine each subkey and compute values most likely to be correct by how well they fit the model (template). Continue until the key is fully recovered.”

Template attacks are very effective. Once a mold of a device has been built, adversaries may do as they wish with the device. The only way to effectively stop this attack is to hide or mask your device so an adversary can’t get a mold of the device.

1.8 Birthday attack

1.8.1 Simple Hash Functions

Before delving into birthday attacks there is a key concept to understand. Hash functions are the backbone to understanding birthday attacks. Hash functions are functions that take a large value and assign it to a particular location in memory. A simple hash function would be as follows, a node-based hash function with 10 potential memory locations. Let n be any positive number from 0 to infinity. Take the equation n mod 10 and that equals the hash value. So, 77 would equal the hash value of 7, 234 equals the hash value of 4, and 1987 equals the hash value of 7 and you would be appended onto the 77 we already have. This hash function is not a good hash function because it allows for collisions to take place. A collision is where two different values equal the same hash value. For making simple data structures, the aforementioned hash function is more efficient than arrays, but when it comes to passwords, this would create a problem because two different passwords could let the same user in. There are more hash functions that get rid of this problem of collision, like linear probing and quadratic probing. This fixes the problem of collision but comes at the expense of using up a large amount of memory. Linear probing expands the hash table, in other words changing the “10” modular number to a larger value and placing the input value to the next available hash value. An example of this would be to take the values 77, 87, and 8 and mod them all by 10. The value of

77 would go into the open 7 hash value, 87 would go into the 8 values, and 8 would go into the hash value of 9. Quadratic probing is similar, except instead of finding a spot ‘x’spot away you look for a value a value “x^2” spots away, increasing x by 1 if a hash value already exists there. As you can see, this takes up memory quickly and would be simple for an attacker to test another value to see which values in a hash table are open and easily be able to put a value in that could get your password’s hash value.

1.8.2 Cryptography Hash Functions

In cryptography however, these hash functions are not used, because secure hash functions need to achieve two things: be deterministic (same output is produced for the same input) and irreversible (impossible for a given hash value to be reversed engineered back to get the plaintext) (Bellore, 2021). Some more extenuating criteria for hash functions are the input (plaintext) can be any length, the output (hash value) is finite, the hash value is easy to compute knowing the input, and they still must be collision free (Gupta, n.a.). As you can see from above, those hash functions achieve none of these requirements. The finite size of the hash function is determined by which function is used; some common ones used are MD5, SHA-1, and bcrypt, which produce 2^{128}, 2^{160} and 2^{192} respectively (Bellore, 2021). There are many more than the ones mentioned, but SHA-1 can be seen as follows:

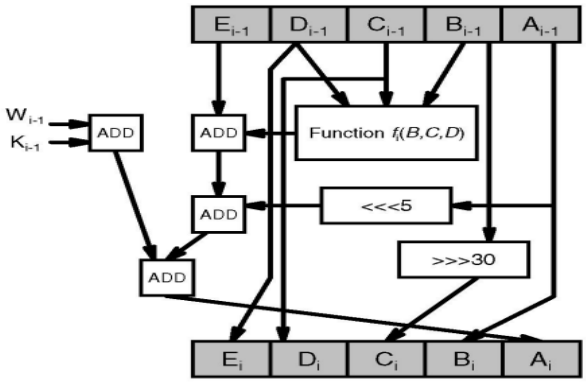

Fig 1.8.1 A round of cryptographic hash function

While the breakdown of the SHA-1 function is more in depth in (Wang, 2024), some of the key elements to understand how the SHA-1 function hashes is that it takes an input and goes through an 80-step process that processes a 512-bit block. It first goes through a process called padding, which adds 0’s to the beginning of the message, until it is 448 mod 512 long. Then a 64-bit of the original length is appended to the 448 bits. Then two buffers are added (one can be seen as the A-E above). Then, this is where the 80 steps come in and the message proceeds into 16-word blocks with 4 rounds at 20 steps each. Then after the 512 bits have been processed, the last block (the bottom row of the above image) is the hashed 160-bit message, which cannot be reversed to get the plaintext and its own special output.

1.8.3 Birthday Paradox

Now to explain the birthday paradox. This paradox asks the question, “How many people would need to be in a room, such that there is a 50% chance that 2 people have the same birthday?” (assuming all birthdays are equal). Well first look to see the chance that two people do not have the same birthday, which would be (365/365) (this is the first person’s birthday) and (364/365) (this is the second person's birthday). You subtract one from the numerator because there are 364 other days in a year that the second person could have a different birthday than the first. We find out that there is about a 99.7% chance that two people do not have the same birthday and about a 0.3% chance they do. Many would think this small trend would continue and you might need somewhere around 180 people before there is a 50% chance 2 of those 180 people share a common birthday, but the actual answer is only 23 people. This can be seen by the following formula, which denotes the chance that k amount of people will have different birthdays:

$$\frac{\mathbf{365} \cdot (\mathbf{365}-\mathbf{1}) \cdot \ldots \cdot (\mathbf{365}-\boldsymbol{k})}{\mathbf{365}^{\boldsymbol{k}}}$$

In this instance, our k value equals 23, and gives a value of about 49.3% chance that out of 23 people, 2 of them do not share a common birthday. This then means there is about a 50.7% chance that 2 people do share the same, and this statistical concept is especially important in breaking hash functions to get one’s digital signature, password, etc.

1.8.4 Birthday Attacks

Now, with the background on hash functions and the birthday paradox, the birthday attack is easier to understand. Two people having the same birthday have the same idea that two inputs hashing to the same value, i.e., a collision occurs. Let us look back at the size of the hash table for SHA-1 hash function. To get a 100% chance that a collision will occur, is (2^160 +1). This is also known as a brute force attack and would be exactly (1.46 * $\mathbf{10^{24}}$) attempts. With the birthday attack, this makes it twice as fast to attack. The formula k≈√n (Gupta, n.a.), is derived at the value where there is a 50% of the birthday attack working, where k equals the password, and the square root of n equals the size of the hash function. In the case of SHA-1, the value of the hash function is $\mathbf{2^{160}}$, so the 50% chance to get a desired password using the birthday attack is around 280, which is around, $\mathbf{1.2} * \mathbf{10^{24}}$ derive another formula which is $\boldsymbol{k} = \mathbf{2}^{\frac{\boldsymbol{m}}{\mathbf{2}}}$ (Gupta, n.a.), where m equals the length of the message digest. The following table shows the possibilities with stronger and weaker m values:

Bits	Possible outputs	Attempts for Desired probability of random collision	
		50%	75%
16	65,536	300	430
32	4.3×10^{9}	77,000	110,000
64	1.8×10^{19}	5.1×10^{9}	7.2×10^{9}
128	3.4×10^{38}	2.2×10^{19}	3.1×10^{19}
256	1.2×10^{77}	4.0×10^{38}	5.7×10^{38}
384	3.9×10^{115}	7.4×10^{57}	1.0×10^{58}
512	1.3×10^{154}	1.4×10^{77}	1.9×10^{77}

Fig 1.8.2 Table showing the probability of collision for the number of bits

As you can see, birthday attacks are very efficient for small message digest values, but the hash functions start to scale exponentially, so that once you get to 128 bits (recall that is MDA5) the hash becomes infeasible to execute. Something to note from the table is once 50% is achieved, 75% does not take too many more attempts to figure out. To counter birthday attacks, better hash functions need to be computed, but the 512-bit limit currently in use right now is more than enough to keep attackers away from information.

1.8.5 Yuval's Birthday Attack

An example of a birthday attack is Yuval's Birthday attack algorithm. Yuval's birthday attack uses a legitimate message (x), fraudulent message (y), m bit length, a one-way hash function, and an x' and y' which are different by H(x') = H(y') (Gupta, n.a.). The goal of this is to make minor changes from the legitimate message until a match between the hash functions is found. This is used for digital signature forgery and allows an attacker to get the victim to sign their digital signature on a document and then use the document they modified to copy that signature and since they have the same hash value, the signature would still be valid for the changed document.

To counter this Gupta suggested, "In Digital signature we can fail the Birthday attack by before signing any electronic document, make a slight change in generalization you can say that the output length of the hash function used for a signature scheme can be chosen large enough so that the birthday attack becomes computationally infeasible" (Gupta, n.a.). There is also another attack that can break down a DNS server using the birthday attack strategy. The BIND birthday attack does the following: 1) sends many quires to one nameserver 2) Attacker gives disguised answers to the quires it just made 3) when a device tries to contact that nameserver 4) one of the disguised information answers is given to the device (Gupta, n.a.). Both attacks use statistical chances to gain information on victims and make them unaware they are being attacked.

2 Potential Consequences

The risks and consequences of these cryptographic attacks are widespread, affecting individuals and organizations. Here are some potential risks:

1. Risk of User Account Compromise: If a cryptographic attack is successful, it may lead to the leakage of user data. Attackers can gain unauthorized access to sensitive information, manipulate data, or impersonate infected users to carry out malicious activities (Swisher, 2024).
2. Loss of Trust: For organizations, those affected by cryptographic attacks may lose the trust of customers, partners, and stakeholders, resulting in a decline in business and reputation. For individuals, if user account information is leaked, personal identities may be stolen for illicit activities, leading to damage to personal reputation (Swisher, 2024).
3. Financial Loss: Cryptographic attacks can lead to financial losses for organizations due to expenses related to incident response, forensic investigations, legal fees, regulatory fines, and potential lawsuits resulting from data breaches. If an individual's bank or investment accounts are compromised, financial assets may be stolen (Swisher, 2024).
4. Operational Disruption: Cryptographic attacks can disrupt normal business operations, leading to downtime, and loss of productivity (Swisher, 2024).
5. Malware and ransomware distribution: Attackers may exploit the compromised platform to propagate malicious software and ransomware, thus further amplifying the impact of the attack. This could escalate like a snowball, triggering additional security incidents within the affected organization and even among its contacts (Swisher, 2024).

3 Possible Preventative Measures

Since cryptographic attacks may cause a lot of harm, we need to prevent such attacks. There are several prevention methods:

1. Update passwords regularly: Force users to change their passwords regularly to reduce the risk of their passwords being compromised.
2. Strengthen access control (rainbow table attacks): Restrict access to the database where password hashes are stored to ensure only authorized personnel have access to this sensitive data.
3. Use Salt (rainbow table attacks): A randomly generated salt is combined with the password for hashing before storing the password hash. The salt is a random string that causes two users to have different hashes even if they use the same password, preventing rainbow table attacks (Swisher, 2024) (Makhene, 2024).
4. Use appropriate hashing algorithm (rainbow table attacks): Choose a secure hashing algorithm such as SHA-256 or SHA-3, etc. rather than one that is easily broken by rainbow table attacks (Makhene, 2024).
5. Update hash algorithms regularly: Follow the latest advances in cryptographic hash functions. If more secure algorithms become available, consider using the latest and most powerful hashing methods (Makhene, 2024).
6. Use CAPTCHA (Credential Stuffing Attack): Require users to solve the "Completely Automated Public Turing test to tell Computers and Humans Apart" (CAPTCHA) or

similar puzzles every time they log in. This can help identify automated/bot attacks and prevent automated login attempts and may also slow down credential stuffing attacks (Credential Stuffing Prevention Cheat Sheet, n.d.).

7. Educate users to use different passwords: Even if a user chooses a strong password, if they share that credential between different accounts, they are at risk of being successfully compromised by a credential stuffing attack. Educate users on the importance of avoiding password reuse and choosing strong and unique passwords. Users are advised to use password manager tools to avoid easy-to-remember passwords and use discovery tools to expose default passwords on devices that have not been changed (Lenaerts-Bergmans, 2023).
8. Increase password complexity: Enforce passwords that are long and complex enough, including a combination of uppercase and lowercase letters, numbers, and special characters. This can increase the size of the rainbow table, making it more difficult for an attacker to build and use the rainbow table (Swisher, 2024) (Makhene, 2024).
9. Enforce rate limits and account lockouts: Implementing rate limits and account lockout mechanisms effectively thwart automated password-guessing attempts. Rate limiting limits the number of logins attempts that can occur within a given period, while account lockout policies lock an account after a certain number of failed login attempts. These measures prevent brute force and rainbow table attacks and alert administrators to potential security threats (Swisher, 2024) (Makhene, 2024).
10. Use multi-factor authentication (MFA): Implement multi-factor authentication, which requires users to provide other authentication factors in addition to passwords, such as mobile phone verification codes, hardware keys, or biometric information (Swisher, 2024) (Makhene, 2024).
11. Monitor abnormal behaviors: Implement security monitoring measures, regularly check login logs and abnormal activities, and detect possible attacks in a timely manner (Swisher, 2024) (Makhene, 2024).
12. Increase in key size (Meet in the Middle attack): Usage of bigger keys might prevent or slow down a meet-in-the-middle attack because of the required complexity in calculations. This may become a time-consuming process.
13. Introduce randomness and padding: Usage of IV’s and padding in any algorithm will make the output less predictable and thus more difficult for the attackers to break into the system.
14. Using HTTPS encryption (Man in the Middle Attack): Use HTTPS protocol to secure the data that is being sent across the web to lower the chances of an attacker trying to eavesdrop or manipulate the data.
15. Using Strong credentials (Man in the Middle Attack): Using weak credentials for login purposes might make it easy for the attacker to access your information. So, it is always a good policy to use a strong password with a mixture of letters (both Lower and Upper case), numbers and special characters that are long. This provides good protection.
16. Complicated Algorithms (Frequency Analysis Attack): Make use of more advanced encryption techniques, such as AES (Advanced Encryption Standard), which have

high diffusion qualities and non-linear encryption processes that make them resistant to frequency analysis.

17. Polyalphabetic Ciphers (Frequency Analysis Attack): The efficacy of frequency analysis can be diminished by using polyalphabetic ciphers as opposed to monoalphabetic ones. A well-known example of adding difficulty using numerous alphabets is the Vigenère cipher.
18. Random Padding (Frequency Analysis Attack): To complicate analysis, random padding involves adding random characters to the plaintext prior to encryption, which disrupts the characters' typical frequency distribution.
19. Encryption Mode (Padding Oracle Attack): Avoid encryption modes like CBC (Cipher Block Chaining) with predictable error messages that are susceptible to padding oracle attacks. Rather, employ verified encryption modes such as GCM (Galois/Counter Mode), which offer both integrity and confidentiality.
20. Error Message Handling (Padding Oracle Attack): To stop attackers from differentiating between different error types, such as padding errors versus decryption errors, use consistent error messages for all encryption error types.
21. MAC-then-Encrypt (Padding Oracle Attack): Prior to encryption, apply a Message Authentication Code (MAC) to the plaintext. This reduces the risk by enabling the server to validate the MAC first during decryption and avoiding the requirement of checking padding.

Conclusion

As we come to the end of this thorough examination of cryptographic attacks, we have covered the nuances of a variety of techniques, from Birthday Attacks to Rainbow Table Attacks, explaining their workings, advantages, disadvantages, and common targets in addition to providing real-world examples that highlight their importance in the field of cybersecurity. This study makes it clear that the field of digital security is a dynamic battleground where the ingenuity of attackers constantly poses a challenge to the development of defensive tactics. The vulnerabilities present in password-based systems were brought to light during the discussion of the Rainbow Table and Credential Stuffing Attacks, highlighting the urgent need for strong password restrictions and sophisticated authentication techniques. Analyzing Man-in-the-Middle and Meet-in-the-Middle Attacks demonstrated how vulnerable data is while it is being sent, supporting the need for end-to-end encryption and stringent key management. Moreover, Frequency Analysis and Padding Oracle Attacks highlighted the significance of algorithmic complexity and cautious implementation by exposing potential weaknesses in the cryptographic algorithms themselves. The analysis of Side-Channel and Birthday Attacks shed light on the frequently disregarded avenues via which data leaks can transpire, indicating a comprehensive security strategy that considers both the digital and physical domains.

In addition to demonstrating the complex nature of cryptographic weaknesses, this study makes a compelling case for the ongoing development of cybersecurity defenses. It emphasizes how important it is for organizations and individuals to be on guard, take the initiative to strengthen and update security procedures, and promote a security-aware culture. The creation

of innovative defensive technology and the training of an educated and equipped cybersecurity workforce have become essential components in defending the integrity of our digital world against these always changing dangers. The knowledge gleaned from the analysis of these cryptographic attacks will surely help to improve cybersecurity theory and practice as we traverse this constantly changing terrain, strengthening our collective defenses against the ceaseless tide of cyberattacks.

References

Bellore, Arthur. "Birthday Attacks, Collisions, and Password Strength." Auth0 - Blog, 23 Mar. 2021, auth0.com/blog/birthday-attacks-collisions-and-password-strength/

Beschokov, M. (n.d.). *Rainbow Table Attack*. Retrieved from wallarm: https://www.wallarm.com/what/rainbow-table-attack

BIRTHDAY ATTACK. Colombia University https://www.math.columbia.edu/~goldfeld/BirthdayAttack.pdf

Cheng, L., & Meng, F. (2023). Public key authenticated searchable encryption against frequency analysis attacks. Information Sciences, 640, 119060. https://doi.org/10.1016/j.ins.2023.119060

CODE BOOK - THE SCIENCE OF SECRECY : SIMON SINGH : Free download, borrow, and streaming : Internet Archive. (2017, April 6). Internet Archive. https://archive.org/details/CodeBook-TheScienceOfSecrecy

Credential stuffing. (n.d.). Retrieved from wikipedia: https://en.wikipedia.org/wiki/Credential_stuffing

Credential Stuffing Prevention Cheat Sheet. (n.d.). Retrieved from OWASP Cheat Sheet Series: https://cheatsheetseries.owasp.org/cheatsheets/Credential_Stuffing_Prevention_Cheat_Sheet.html

Crypto-IT. (n.d.). https://www.crypto-it.net/eng/attacks/meet-in-the-middle.html

Datta, S., & Datta, S. (2024, March 18). How does Meet-in-the-Middle Attack work? | Baeldung on Computer Science. Baeldung on Computer Science. https://www.baeldung.com/cs/security-meet-in-the-middle-attack

Duong, T., & Rizzo, J. (2011). Cryptography in the Web: The case of cryptographic design flaws in ASP.NET. Padding Oracle Attacks. https://doi.org/10.1109/sp.2011.42

Fardan, N. J. A., & Paterson, K. G. (2013). Lucky Thirteen: Breaking the TLS and DTLS record protocols. Lucky Thirteen. https://doi.org/10.1109/sp.2013.42

Gupta, Ganesh. What Is a Birthday Attack?? Research Gate, www.researchgate.net/profile/Ganesh-Gupta-7/publication/271704029_What_is_Birthday_attack/links/54cfbdcc0cf24601c0958a1e/What-is-Birthday-attack.pdf

Hodges, A. (1985). Andrew Hodges. Alan Turing: the enigma. Burnett Books, London, and Simon and Schuster, New York, 1983, ix + 587 pp. the Journal of Symbolic Logic/the Journal of Symbolic Logic, 50(4), 1065–1067. https://doi.org/10.2307/2273992

Jagannath, S. (2023, September 12). *Rainbow Table Attack – Types, Examples & Preventing it*. Retrieved from DebugPointer: https://debugpointer.com/security/rainbow-table-attack

Katz, J., & Lindell, Y. (2014). Introduction to modern Cryptography. In Chapman and Hall/CRC eBooks. https://doi.org/10.1201/b17668

Lake, J. (2024, January 26). *Credential stuffing attacks explained (and some recent examples)*. Retrieved from comparitech: https://www.comparitech.com/blog/information-security/credential-stuffing-attacks/#Recent_credential_stuffing_attacks

Larew, K. G., & Kahn, D. (1968). The Codebreakers: The Story of Secret Writing. the American Historical Review, 74(2), 537.https://doi.org/10.2307/1853680

Lenaerts-Bergmans, B. (2023, November 08). *Credential Stuffing*. Retrieved from Crowdstrike: https://www.crowdstrike.com/cybersecurity-101/credential-stuffing/

Makhene, T. (2024, Februrary 06). *Understanding a rainbow table attack*. Retrieved from Paubox: https://www.paubox.com/blog/understanding-a-rainbow-table-attack

Man In the Middle (MITM) Attacks - Definition & Prevention | Rapid7. (n.d.). Rapid7. https://www.rapid7.com/fundamentals/man-in-the-middle-attacks/

Paterson, K. G., Ristenpart, T., & Shrimpton, T. (2011). Tag size does matter: Attacks and proofs for the TLS Record Protocol. In Lecture notes in computer science (pp. 372–389). https://doi.org/10.1007/978-3-642-25385-0_20

Paterson, K. G., & Yau, A. K. L. (2004). Padding Oracle attacks on the ISO CBC Mode Encryption Standard. In Lecture notes in computer science (pp. 305–323). https://doi.org/10.1007/978-3-540-24660-2_24

Rainbow Table Attack. (n.d.). Retrieved from Byond Identity: https://www.beyondidentity.com/glossary/rainbow-table-attack

Randolph, Mark, and William Diehl. “Power Side-Channel Attack Analysis: A Review of 20 Years of Study for the Layman.” Cryptography, vol. 4, no. 2, 19 May 2020, p. 15, https://doi.org/10.3390/cryptography4020015

Shah, J. (2023, April 13). *What is a Rainbow Table Attack? How To Protect Against It?* Retrieved from 1kosmos: https://www.1kosmos.com/authentication/rainbow-table-attack/

Spreitzer, Raphael, et al. “Systematic Classification of Side-Channel Attacks: A Case Study for Mobile Devices | IEEE Journals & Magazine | IEEE Xplore.” Ieeexplore.ieee.org, 2018, ieeexplore.ieee.org/abstract/document/8141882

Swisher, J. (2024, February 23). *What is a Rainbow Table & How to Prevent These Attacks*. Retrieved from Jetpack: https://jetpack.com/blog/rainbow-table-attack/

Veracode. (n.d.). Man in the Middle (MITM) attack | VeraCode. https://www.veracode.com/security/man-middle-attack

Wang, Guoping. An Efficient Implementation of SHA-1 Hash Function. Department of Engineering Indiana University Purdue University Fort Wayne, ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4017767

What is Credential Stuffing. (n.d.). Retrieved from silverfort: https://www.silverfort.com/glossary/credential-stuffing/#how-to-detect-and-prevent-credential-stuffing-attacks

Yasar, K., & Cobb, M. (2022, April 28). man-in-the-middle attack (MitM). IoT Agenda. https://www.techtarget.com/iotagenda/definition/man-in-the-middle-attack-MitM