

Lecture note 6: Authentication

Chi Yan LEUNG, ZHANG WeiYi, Beichen HUANG, Xiaoqi WANG

April 13, 2024

Lecture 6 covers three authentication scenarios: password, biometric, and public key authentication. Password authentication is the most prevalent approach for access control. If the password for that account is mistakenly leaked, the attacker can get access to the user information on the remote system. Biometric authentication utilizes biological features such as fingerprints, faces, voices, and retinas. This method uses two-factor authentication to successfully avoid common attacks. Finally, this course discusses public key authentication using SSH as an example.

1 Authentication

Authentication is essentially the method of confirming the identity of a person or entity to a server or system. An ideal authentication system would block any attempts to mimic legitimate users and refuse entry to anyone without proper authorization. Thus, one core problem in authentication is, how to demonstrate convincingly that you are indeed the person you claim to be?

1.1 Authentication Factors

In order to achieve authentication, users must provide the system with personal information or credentials to verify their identity. These are known as authentication factors and can be categorized into three types:

- **Something you know (Knowledge Factor)** Knowledge-based authentication requires users to enter specific information they know, like a password or personal identification number (PIN), before accessing a secure system. A common way knowledge-based authentication is compromised is through phishing attacks.
- **Something you are (Inherence Factor)** The inherence-based authentication requires biometric information that is unique to the users, such as fingerprints, faces, and irises. While highly secure, a downside is that it may not always be convenient. Accessing accounts might require additional devices like fingerprint scanners or other specialized equipment for collecting biometric data.

- **Something you have (Possession Factor)** Possession factors require the user to possess specific information or devices before being granted access to the system. The device could be either hardware that could generate one-time passwords, or simply a digital secret key.

Entropy is essentially a measure of the unpredictability or randomness within a system. In the realm of cryptography, Shannon entropy is widely utilized to assess the strength of authentication factors. Consider a plaintext space $\{k_1, k_2, \dots, K_n\}$, and there is a $P(k_i)$ probability that the plaintext k_i would appear, Shannon entropy is mathematically represented as:

$$H(K) = - \sum_{k_i \in K} P(k_i) \times \log_2 P(k_i) \quad (1)$$

To maximize Shannon entropy, it's advisable to construct a password that matches the system's total possible outcomes in length, ensuring each character's occurrence probability is equal. The principle of maximum entropy also suggests avoiding any subjective assumptions when predicting a random event's probability distribution.

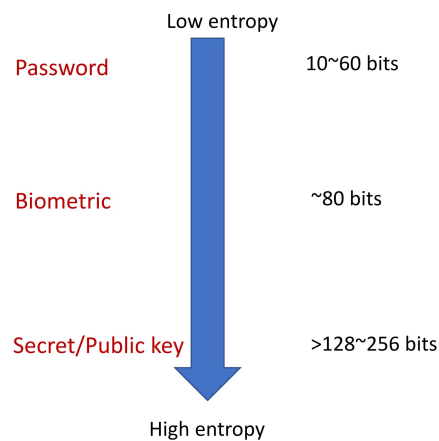


Figure 1: The Shannon entropy for different types of authentication factors.

Figure 1 shows the Shannon entropy for three major types of authentication factors. Shannon entropy is measured in bits, as it quantifies the average minimum number of bits needed to encode a set of messages, considering the probability of each message's occurrence.

Beyond Shannon entropy, conditional entropy offers a more nuanced approach to evaluating authentication factors' robustness, which could be described as:

$$H(K | c) = - \sum_{k_i \in K} P(k_i | c) \times \log_2 P(k_i | c). \quad (2)$$

The conditional entropy is more suitable for real-world scenarios, where the parameter c represents the ciphertext. The conditional entropy measures the uncertainty measurement of plaintext under the circumstance that the attacker has already obtained the ciphertext.

1.2 Difference between Authentication, Authorization and Access Control

- **Authentication** is the initial process that verifies an individual's identity, ensuring that only authenticated users gain system access. It acts as the gateway, akin to a verification checkpoint, that distinguishes between legitimate and illegitimate users. Without this process, unrestricted access could lead to significant security vulnerabilities and potential data breaches.
- **Authorization** determines the scope of actions and resources available to an authenticated user. It builds upon the foundation laid by authentication, delineating the privileges and access levels assigned to different users. For instance, within a system, the access privileges might vary significantly; standard users may only interact with their personal accounts, whereas administrators are granted comprehensive system-wide access.
- **Access Control** Compared to authentication and authorization, access control is a more general concept. It involves all the access control policies and mechanisms, including authentication and authorization. This includes not only gatekeeping functions but also monitoring and logging activities to maintain accountability. In summary, authentication decides if an entity can access the system, authorization decides how many resources the user could access and how many operations the user could perform, and access control involves all these policy and mechanisms.

In summary, authentication decides if an entity can access the system, authorization decides how many resources the user could access and how many operations the user could perform, and access control involves all these policy and mechanisms.

1.3 Authentication Paradigm

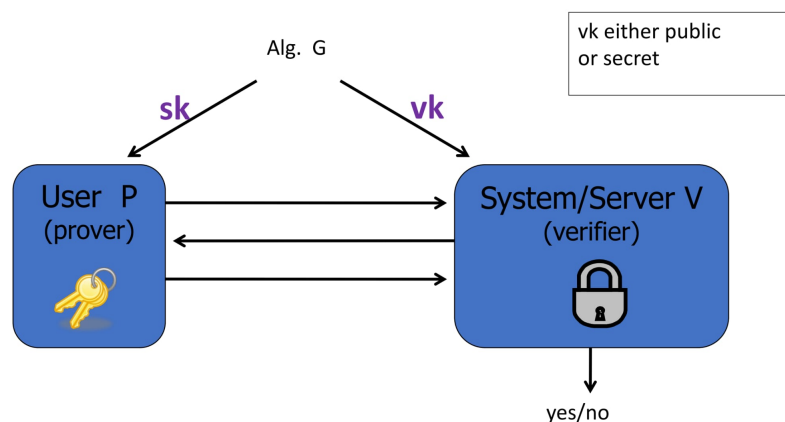


Figure 2: An illustration of authentication paradigm. The server would verify the validity of one's identity after receiving requests from the user.

Figure 2 is an overview of the authentication paradigm, which typically unfolds in four steps: First, registration. During this initial phase, the user is assigned a secret key, which could be a password they create, their biometric data, or another form of personal identifier. Concurrently, the server is assigned a verification key. This key might be identical to the user's secret key or a public key specifically intended for verification purposes.

When the user needs to access a service, they initiate the process by sending a request to the server. After receiving the request, the server evaluates the request to determine if authentication is necessary. If it is, the server responds to the user, potentially including a challenge C as part of the authentication process. The user responds to the challenge by encrypting the required information and sending it back to the server. The server then decrypts and verifies the information to confirm whether the user's account is legitimate.

For instance, the Digest Authentication scheme, outlined in RFC 7616, operates as follows:

- 1) A client initiates the process by sending a request to the server.
- 2) Should the request require authentication, the server issues a challenge back to the client. This challenge includes the authentication method to be used, a randomly generated nonce (a number used once), and other relevant details.
- 3) The client then creates an encrypted hash by combining the nonce, the requested URL, the username, and the password. This encryption is performed using MD5, a hash function that produces a 128-bit hash value. The resulting encrypted hash is known as the "response." Alongside this "response," the client sends additional information including the username, realm and nonce back to the server.
- 4) Upon receipt, the server uses the username to retrieve the corresponding password and generates a "response" using the same method as the client. If the server-generated "response" matches the one from the client, the server confirms the client's identity and grants access to the requested resources.

This method enhances security by avoiding the direct transmission of plain passwords over the network and instead using a hash value that verifies the user's identity without exposing their actual password.

2 Password Authentication

Password authentication is used to validate the identification of users, who must supply a pair of a username and a secret password as evidence of their identity. Upon entering credentials, they are cross-referenced with the stored credentials in the system's database. Access is only provided to the user if the credentials are a perfect match. Typically, the system will keep the collection of users and passwords in a database. Figure 3 shows the paradigm of password authentication.

First, an algorithm within the system generates the password, which is then distributed to a user and saved in the system's storage. In order to verify the user's identity, the user

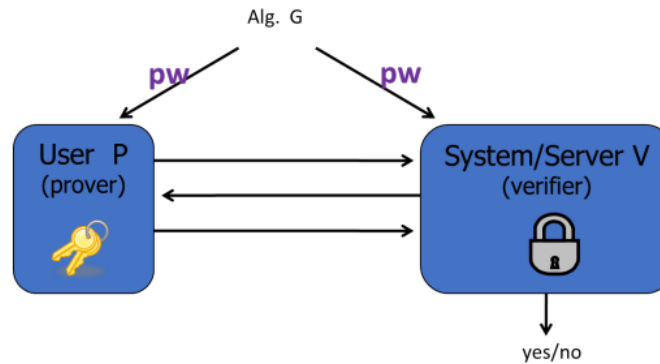


Figure 3: Password authentication paradigm.

transmits their password to the system. Subsequently, the system will ascertain whether to approve or decline the user's authentication by comparing the provided password with the stored password. However, for the purpose of addressing security concerns, such as avoiding the interception of the secret password, password authentication will be used in combination with a suitable encryption mechanism to ensure that the password is not sent directly. Figure 4 shows the flow of an Encryption-with-Password Authentication scheme.

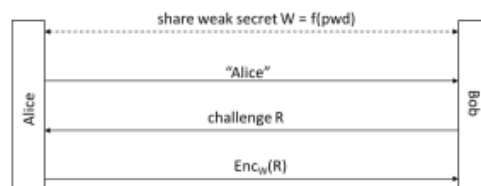


Figure 4: Encryption-with-password authentication scheme.

Alice and Bob will first share the weak password $W = f(pwd)$. Then, Alice will tell Bob her identity "Alice". After that, Bob will send a challenge R , which is a meaningless string, to Alice and ask Alice to encrypt with W by a specific encryption algorithm which only Alice and Bob know. After Alice successfully sends the encrypted message to Bob, and answers correctly. The authentication process will be completed.

2.1 Measure the Password Strength

Despite the simplicity of password authentication, many people fail to apply it correctly. Some surveys indicate that 50% of users retain their home routers' default passwords [TJYW06, MS09]. Two surveys found that 50% of users choose the default password for their routers, which leaves a potential safety hazard for hackers to break the password. Furthermore, a survey discovered that an overwhelming 99% of employees at Dixie Bank use the password "password123", an extremely vulnerable choice that exposes their accounts to hackers and results in significant financial losses for the bank. In addition, RockYou, which is a company that created widgets for MySpace and built apps for many social networks, had been compromised in December 2009 due to an attack by hackers. 32 million user passwords have

been disclosed and posted to the internet, while the passwords were in the clear. The leaked password dataset (Table 1) reveals that over 1% of Rockyou users picked “123456” as their password. Eventually, the RockYou attack awakened the attention and importance of user password composition, while password strength is one of the factors needed to attend.

Rank	Password	No. of Users with Password (absolute)
1	123456	290731
2	12345	79078
3	123456789	76790
4	Password	51622
5	iloveyou	290731
6	princess	35231
7	rockyou	22588
8	1234567	21726
9	12345678	20553
10	abc123	17542

Table 1: Top 10 Rockyou password.

Various methods exist in the real world for measuring the strength of passwords, and one of the methods is Entropy. Entropy was introduced by Claude Shannon and it is also known as Shannon Entropy. The following is the brief definition of Shannon Entropy:

- Let X denote the distribution of passwords, and passwords are selected from X .
- n is the size of support of X .
- p_1, p_2, \dots, p_n denote probabilities of passwords in decreasing order

Shannon Entropy is defined as:

$$H(X) = - \sum p_i \log_2 p_i \quad (3)$$

However, the Shannon Entropy can be a poor measure of password strength. For instance, assuming we have 1,000,00 passwords, 0.01 is the probability of the initial password, denoted as p_1 . An approximation of the probability of the remaining passwords, p_2 to p_n , can be obtained by dividing $(1 - 0.01)$ by 999,999, which is approximately $1/220$. The Shannon entropy assigns an approximate value of 19 to the entropy of these passwords, represented as $H(X)$. A potential adversary may, however, be compelled to make a rudimentary estimate regarding the initial password, which has a 0.01% chance of occurring.

Compared with Shannon Entropy, Min-entropy is a more effective method for assessing password strength as it considers the prevalence of the most popular passwords. Min-entropy is defined as:

$$H_\infty(X) = - \log_2 \max_{x \in X} p(x) \quad (4)$$

Figure 5 demonstrates that the min-entropy of the password set is approximately 6.6. This means the guessing probability for the most likely password across a population is

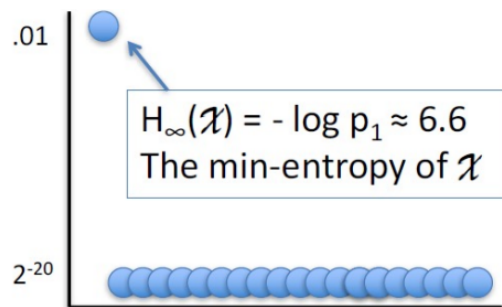


Figure 5: Min-entropy of the password set.

approximately 0.9%. Guessing Probability (GP) denotes the probability of the most probable password over a population, and can be calculated by the following equation:

$$GP_{max} = 2^{-H_{min}(X)} \quad (5)$$

Let's consider the password "123456" from the Rockyou dataset, which is the most frequently used password. The GP of "123456" is approximately 0.9%. It indicated that around 1 out of 10 users have this password. In this case, we may see that the GP can be used to assess the vulnerability of the weakest accounts, which are more likely to be targeted by attackers.

To assist users in generating more secure passwords, system administrators frequently enforce certain requirements such as a minimum length, a specified number of character classes, or the absence of passwords on a block list. Specifically, 1c12+NN10 can be used to increase password security [TBCC20]. The meanings of 1c12 and NN10 are as follows.

- **1c12:** One class with at least 12 characters.
- **NN10:** Required passwords to have password strength estimates no weaker than 10^{10} guesses.

2.2 Storage of Password

To securely store passwords, adhering to the principle of never storing passwords in plaintext is essential. Alternatively, if the attacker gains access to the passwords, they will get knowledge of all users' passwords and be capable of launching attacks on their accounts on other websites, provided that the user is likely to employ the same password for many sites. In this regard, the cryptographic hash functions can be used to cope with that. The hash function offers the characteristic of one-way, deterministic, and collisions resistant to increase the security of password storage.

However, only hashing the password remains possible security vulnerabilities. For example, some brute force attacks like the dictionary and rainbow table attacks can break the password. Because some SHA-hashing like SHA256 is quite fast to compute. Plus, the attackers can pre-compute $H(\text{word})$ for every word in the dictionary to build the Rainbow table.

To resolve these issues, the recommended approach is to combine salt and the password prior to performing the hashing operation. A salt is a predetermined, unchanging, and highly secure random value of a specific length, used in cryptographic operations. The use of varying salt values guarantees that passwords from different users, even if they are similar, remain undisclosed. In addition, it increases the complexity of pre-computed lookup attacks since an attacker would be required to pre-compute hashes for a greater range of potential passwords. Table 2 is an example of a table of hashed user-password with salt by SHA-256.

Username	Password (Plaintext)	Salt	Hash (SHA-256 with Salt)
alice	password	AAA	8dde86ea8d7c75844d166e4e5d4cfabc...
bob	123456	AAA	99c6e57451ebbbee2cdbc9505755dc4a...
charlie	hunter2	IamSalt	475c79b173e0b287ae447d1113ef690d...
dakotah	123456	IamSalt	8715c0f683e985748e9f9f24d2c6cb135...

Table 2: Sample of User Table Hashed User-Password with Salt (by SHA-256)

One other method to enhance password security is to deliberately slow down the process of hashing to impede cracking operations. As an illustration, the PKCS#5 method (Figure 6) entails repeatedly applying a hashing function to the password with a salt 100,000 times until the final hash value is obtained. To enhance the expense for the attacker, we can employ slower and memory-intensive hash methods like Scrypt or Argon2.

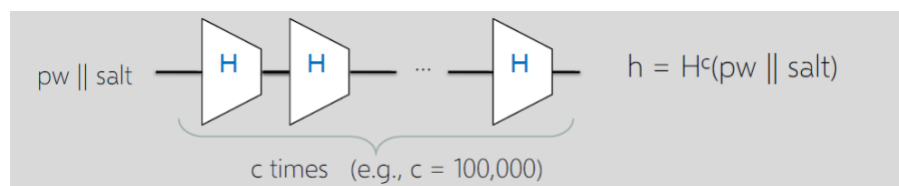


Figure 6: The PKCS#5 approach.

2.3 Attack towards Password Authentication

Generally, password authentication attacks may be categorized into two types: Online and Offline attacks.

2.3.1 Online Attack

An online attack is the act of attempting to guess passwords by logging into a live system. Usually, certain websites or login systems limit the number of attempts to guess the correct passwords to prevent online attacks. However, online attacks have shown to be more effective than previously thought because of the lack of diversity in people's password selections and the strong connection between passwords and personal information such as birthdays [WZW⁺16]. Figure 7 shows the primary target of online attacks, the most vulnerable platform. For the purpose of verifying if our password has been compromised, we can access <https://haveibeenpwned.com/> to check.

Yahoo - 3 billion	Twitter - 330 million	Canva - 137 million	Rambler - 91 million
Aadhaar - 1.1 billion	NetEase - 234 million	Apollo - 126 million	Facebook - 87 million
Verifications.io - 763 million	LinkedIn - 165 million	Badoo - 112 million	Dailymotion - 85 million
Yahoo - 500 million	Dubsmash - 162 million	Evite - 101 million	Dropbox - 69 million
Marriott/Starwood - 500 million	Adobe - 152 million	Quora - 100 million	tumblr - 66 million
Adult Friend Finder - 412.2 million	MyFitnessPal - 150 million	VK - 93 million	
MySpace - 360 million	Equifax - 148 million	MyHeritage - 92 million	
Exactis - 340 million	eBay - 145 million	Youku - 92 million	

Figure 7: Biggest data breaches from online attack.

2.3.2 Offline Attack

An offline attack involves guessing passwords from a password database, usually obtained covertly. Specific examples like the dictionary and rainbow table attack. In addition, pre-computation techniques may greatly expedite offline assaults. Table 3 shows the performance for cracking offline utilizing a Nvidia RTX 2080 Ti GPU.

Hash Type	Hashes/second	Passwords/month for 10M set	Brute-force equivalent
MD5 unsalted	≈50G	≈130,000,000G	≈8-9 characters
MD5 salted	≈50G	≈13G	≈5 characters
MD5crypt (=salted, 1000 x MD5)	≈22M	≈5.6M	≈3-4 characters
Bcrypt (= salted, work factor 8)	≈3500	≈900	≈1-2 characters

Table 3: The performance for cracking utilizing a Nvidia RTX 2080 Ti GPU [Sca19]

2.4 Multi-Factor Authentication

Multi-factor authentication is the advanced version of single-password authentication. This is also a way of verifying a user's identity by requiring them to provide two or more pieces of evidence to access a website or application. The second factor often consists of verifying ownership of a telephone number (via SMS), device (via authenticator app), or hardware token (via a one-time-password token or a universal second factor U2F token). Two-factor authentication (2FA) is a very efficient method for safeguarding users against potential attacks. According to a March 2020 analysis by Microsoft, 99.9% of hacked accounts did not employ multi-factor authentication. According to Google research in February 2022, the effective use of 2FA led to a 50% reduction in hacked accounts.

2.4.1 Short Message Service (SMS) Authentication

The fundamental concept underlying SMS authentication is shown in Figure 8. Once the user inputs their password, the system creates an authentication code and delivers it to the user's mobile device over SMS. Subsequently, the user transmits the authentication code back to the system in order to verify their identity. To compromise an account that is safeguarded by SMS-based 2FA, an assailant must possess the authentication code in order

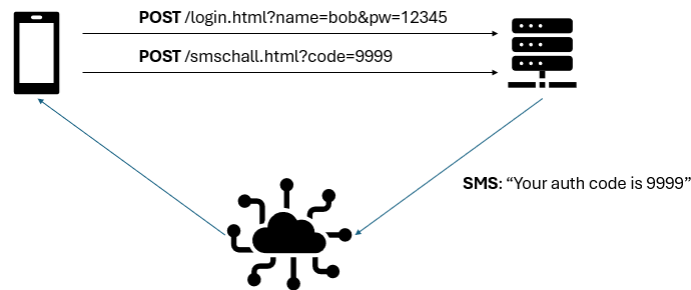


Figure 8: SMS authentication process.

to successfully gain access to the account. Nevertheless, there are still several methods to bypass SMS-based 2FA.

- **Physical attacks:** Have physical access to the gadget that is capable of receiving SMS messages.
- **SIM swap:** Employ social engineering tactics to deceive the phone supplier into registering the victim's phone as the attacker's device.
- **Phishing attacks:** Deceive or manipulate the user into revealing SMS messages to attackers.

However, the lack of usability continues to be a significant obstacle that hinders the acceptance and implementation of the 2FA.

2.4.2 Time-based One-Time Passwords (OTP)

Time-based one-time passwords include the user's device generating an OTP updated every 60 seconds or with each button click. This is done via a shared secret key. The user inputs the OTP alongside their password to finalize the authentication procedure. Figure 9 shows the process of Time-based OTP.

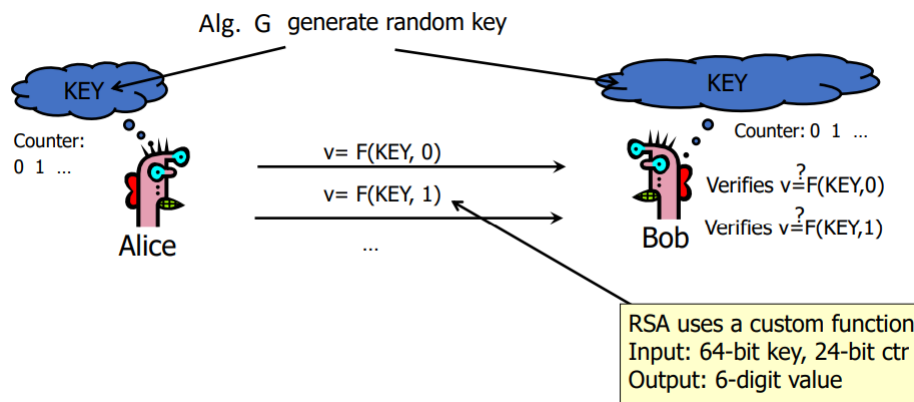


Figure 9: Time-based OTP process.

The user and the system initially possess a randomly generated KEY and a counter value. To verify the user's identity, the user must provide a 6-digit value v to the system. This value is created using the function F , which considers the KEY and the counter. The counter value is increased based on time. The system will then verify if v is equivalent to the value generated by the system. To guarantee the protocol's security against eavesdropping, the function F used to generate time-based OTP must be a secure pseudorandom function (PRF). A good example is RSA SecurID, which utilizes a PRF to combine a 64-bit key with a 24-bit counter value, resulting in a 6-digit output.

3 Biometric Authentication

Traditional authentication methods are susceptible to misuse, theft, and unauthorized access, especially with the continuous advancement of technology. To address these challenges, biometric authentication systems have emerged as a more secure and reliable alternative. Biometric authentication relies on the unique physiological and behavioral characteristics of individuals to verify their identity. Unlike traditional methods that authenticate based on what a user knows (such as a password), biometric authentication verifies based on who the user is [DTL13]. By leveraging biometric technology, individuals can be authenticated based on their distinct features, such as fingerprints, iris patterns, facial features, voice, or even behavioral traits like typing rhythm or gait. These biometric traits are inherently unique to each individual and are difficult to replicate or forge, making biometric authentication systems highly secure and resistant to unauthorized access.

3.1 Biometric Error Rates

An important aspect of evaluating biometric systems is understanding their error rates, which can be categorized into "fraud rate" and "insult rate."

Definition 1 (Fraud Rate). *Fraud rate refers to the likelihood of the system accepting a forgery or false accept. In other words, it measures the probability that an impostor is incorrectly identified as a legitimate user.*

Mathematically, the fraud rate can be represented as:

$$\text{Fraud Rate} = \frac{\text{Number of False Accepts}}{\text{Number of Impostor Attempts}}$$

Definition 2 (Insult Rate). *Insult rate pertains to the system's tendency to reject valid users or false reject. It quantifies the probability that a genuine user is incorrectly denied access.*

Mathematically, the insult rate can be expressed as:

$$\text{Insult Rate} = \frac{\text{Number of False Rejects}}{\text{Number of Genuine Attempts}}$$

It's important to note that there's a trade-off between fraud rate and insult rate, primarily determined by the acceptance threshold of the biometric system.

Increasing Acceptance Threshold: When the acceptance threshold is raised, the system becomes more stringent in accepting biometric samples as valid. This leads to a

decrease in the insult rate, as fewer genuine users are falsely rejected. However, it also results in an increase in the fraud rate, as impostors may find it easier to pass the stricter authentication criteria.

To optimize both the fraud rate and insult rate in biometric systems, it's crucial to address the underlying causes of error rates. Error rates in biometric authentication systems arise from various factors, including environmental conditions, user variability, and technological limitations [MR10]. Bio-features, such as fingerprints, iris patterns, or facial characteristics, can exhibit variability over time due to factors like aging, injury, or changes in health.

The Error rate is mainly due to **the instability of bio-feature**. The instability of bio-features can lead to fluctuations in authentication performance, resulting in higher fraud rates or insult rates [ZZJ⁺18]. For instance, variations in lighting conditions or skin conditions may affect the accuracy of fingerprint recognition systems [O'C13].

3.2 Reduce Error Rates via Fuzzy Extractors

One approach to mitigate the impact of bio-feature instability on error rates is the use of fuzzy extractors. A fuzzy extractor is a cryptographic technique that extracts a stable and reproducible representation, or template, from noisy or variable data. In the context of biometric authentication, a fuzzy extractor can generate a reference template from an initial biometric sample, such as a fingerprint image or iris scan. This reference template is designed to be resilient to minor variations in the bio-feature, allowing for consistent authentication performance over time. When a user presents their biometric sample for authentication, the system compares the presented sample with the reference template using a matching algorithm to determine authentication success.

Dodis et al. (2004) [DRS04] proposed the fuzzy extraction technique, which encompasses two fundamental components:

- **Fuzzy Extractor:** It extracts a random string with a uniform distribution from the input biometric signal R .
- **Secure Sketch:** It derives the public information P from the input biometric information x . In scenarios where the input x_0 closely resembles x , the original x can be reconstructed from P and x_0 .

Fuzzy extraction technology employs both the fuzzy extractor and secure sketch to derive a uniform random character R and public information P from the biometric information x in a robust manner. When a biometric sample x_0 exhibits minimal deviation from x , the fuzzy extractor, aided by the public information, facilitates the recovery process, yielding the random string R as output.

To streamline the fuzzy extraction process, the following formalization was proposed:

- **Random Character Generation Algorithm** ($\text{Gen}(x) \rightarrow (P, R)$): This algorithm accepts the user's biometric information x as input and generates a string R , which serves as the user's random key, along with the corresponding public information P .
- **Random Character Recovery Algorithm** ($\text{Rep}(x_0, P) \rightarrow R$): Given the user's biometric sample x_0 and the associated public information P , this algorithm outputs

R if the error between the two biometric inputs falls within the specified allowable range e , denoted by $\text{dis}(x, x_0) \leq e$.

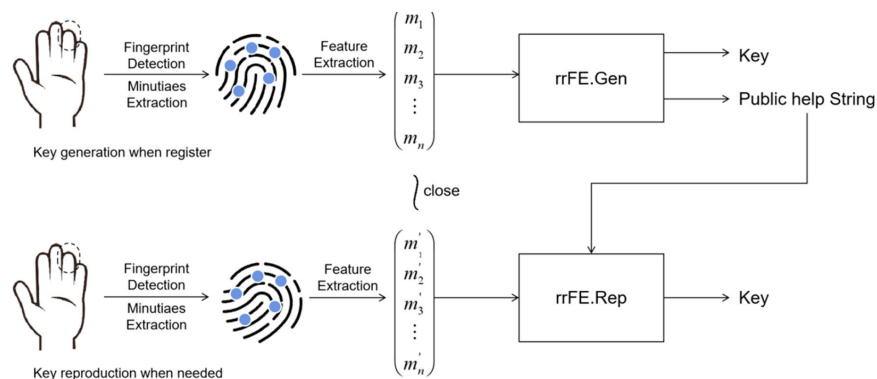


Figure 10: Flow chart of extracting a key from fingerprint based on the fuzzy extractor. [BY23]

The fingerprint-based key extraction process, illustrated in Fig.3.2, leverages both the fingerprint feature extraction scheme and the fuzzy extractor scheme. During user registration, the initial step involves fingerprint collection and subsequent feature extraction using a dedicated fingerprint collection device. The extracted feature M is then fed into the rrFE.Gen algorithm, yielding the Key and the auxiliary public string P , which is subsequently stored in the database. Notably, the Key itself need not be retained.

In scenarios where the Key is required again, the process involves collecting the characteristic M' from the same fingerprint, retrieving the previously stored P from the database, and inputting both parameters into the rrFE.Rep algorithm. If the dissimilarity between M' and M falls below the specified threshold t , the key can be successfully regenerated.

In conclusion, leveraging fuzzy extractors is a promising approach to optimize both the fraud rate and insult rate in biometric authentication systems. By addressing the challenge of bio-feature instability, fuzzy extractors contribute to improving the overall accuracy and reliability of biometric-based identity verification mechanisms.

3.3 Discussion

3.3.1 Advantages of Biometric Authentication

- **Convenience and ease of use:** One of the key advantages of biometric authentication is its convenience and ease of use. Instead of remembering complex passwords or carrying physical tokens, users can simply present their biometric traits for authentication, streamlining the authentication process and enhancing user experience.
- **Enhanced security:** Biometric authentication offers a higher level of security compared to traditional methods. Since biometric traits are unique to each individual and cannot be easily replicated, the risk of unauthorized access due to stolen credentials or identity theft is significantly reduced. This makes biometric authentication an ideal solution for protecting sensitive data and securing access to critical systems and resources.

- **Passive authentication:** Biometric authentication occurs seamlessly in the background without requiring active user input, enhancing user experience.
- **Non-sharable (mostly):** Biometric traits, such as fingerprints or facial features, are unique to individuals and difficult to replicate, reducing the risk of unauthorized sharing.

3.3.2 Faced Challenges

- **Privacy concerns:** While biometric data is inherently private, it may not be entirely secret. There is a risk of biometric data being shared between multiple systems, raising privacy concerns.
- **Difficult revocation:** Unlike passwords, which can be easily changed, altering biometric data is challenging and may even be impossible in some cases. This difficulty complicates the process of revoking access in the event of a security breach.
- **Example:** The birthday paradox demonstrates that even with a low false accept rate (e.g., 1 in a million), the probability of false matches can be surprisingly high with a relatively small number of samples. For instance, with only 1609 samples, the likelihood of a false match exceeding 50% illustrates the inherent vulnerability of biometric systems to false positives.

Given these challenges, it is advisable to use biometric authentication as a supplementary, second-factor authentication method rather than relying on it as the sole authentication factor. This approach ensures a more robust and secure authentication process.

3.3.3 Latest Developments

- **Advancements in Iris Recognition [SDNC22]:** Recent comparative studies have consistently highlighted iris recognition as a leading biometric identification method. Despite initial perceptions of being slightly intrusive and inconvenient, iris recognition stands out for its exceptional features, positioning it at the forefront of biometric technology. Already implemented in developing countries like Mexico and India, iris recognition holds significant promise for future applications.
- **Continuous Multimodal Biometric Authentication Systems [RYKH21]:** Recent research has prioritized the development of continuous multimodal biometric authentication (CMBA) systems, recognized for their enhanced effectiveness and security compared to single-modal systems. While CMBA systems primarily utilize common traits, there is growing interest in integrating additional biometrics such as ECG and BVG. Supervised machine learning techniques predominate in CMBA systems, yet the relative performance of supervised versus unsupervised approaches remains a subject for further investigation. Future research endeavors should also explore semi-supervised learning techniques and compare fusion methods to enhance system performance. Moreover, there is a need to explore different combinations of biometric modalities to optimize system usability and effectiveness in various authentication domains.

4 SSH Authentication

In this part of the lecture, we learned about SSH (Secure Shell), a popular protocol for accessing a shell securely over an unsecured network.

4.1 Design Goal

SSH authentication aims to provide a secure method for accessing remote servers over an insecure network, such as the Internet. The primary design goal is to establish a secure remote login session. Unlike web authentication (e.g. SSL), SSH does not rely on a public key infrastructure. Instead, the public/private key pair should be generated locally, and manually uploaded to the server.

4.2 Overview of SSH Authentication

SSH authentication involves a series of cryptographic protocols and techniques to verify the identity of both the client and the server. This process typically includes key exchange, encryption, and authentication mechanisms to ensure the confidentiality and integrity of the communication. This simplified authentication process is illustrated by the fig. 11. The RFC 4252 [LY06] specified the details of this authentication protocol.

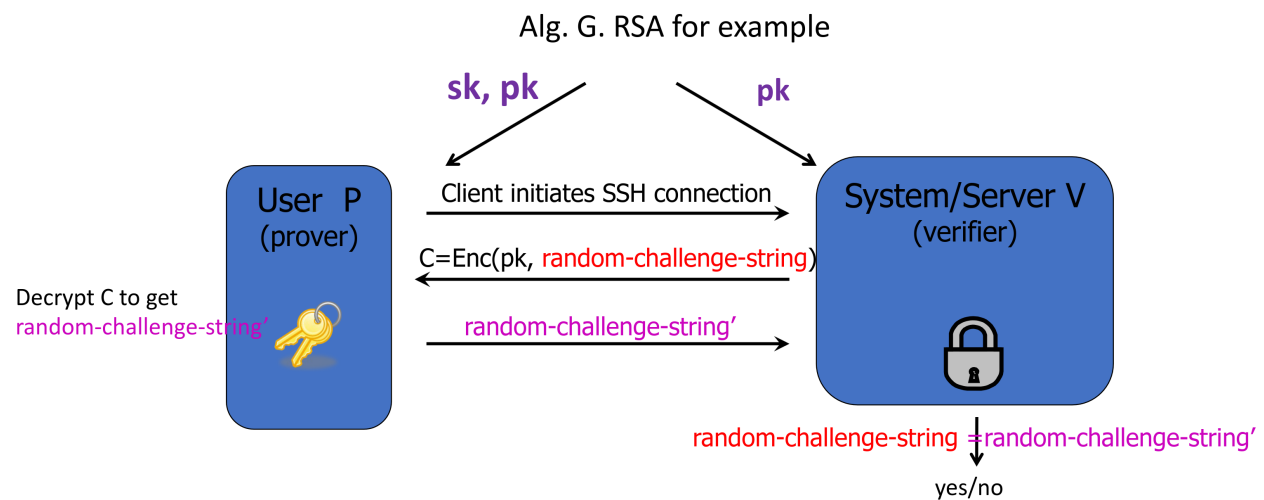


Figure 11: An illustration of the SSH authentication process in the lecture.

At first, a public/private key pair is distributed to the server and the client, where the server will hold the public key, and the client shall hold the private key and public key. Second, the client initiates an SSH connection. After that, The server will generate a random challenge string and encrypt it using the public key. Then, the client will use the private key to decrypt the challenge and return the random challenge string back to the server. Finally, The server will check if the random challenge string matches.

4.3 Pros and Cons of SSH Authentication

4.3.1 Pros

- **Difficult to hack:** SSH key authentication typically employs longer key lengths and more secure cryptographic algorithms, significantly increasing resistance against brute-force attacks.
- **Secure generation process:** SSH key pairs are generated by the system, reducing the risk of human error. Human-generated keys may be predictable or reused across multiple servers, making them susceptible to exploitation. In contrast, system-generated keys utilize true random sources and cryptographically secure random number generators, enhancing security.
- **Secure against sniffing:** Unlike password authentication, SSH key authentication does not transmit the key itself over the communication channel. This mitigates the risk of attackers intercepting the key by sniffing network traffic.

4.3.2 Cons

- **Private key is stored locally:** In SSH public key authentication, users store the public key on their local computers. However, this increases the risk of private key leakage in case the computer is lost or compromised.
- **Distribution of public key is challenging:** Deploying SSH public key authentication requires users to manually upload their keys to the server. This process often necessitates either physical access to the machine or an initial remote connection. However, securely establishing this connection can be challenging, as users typically rely on SSH for secure access, creating a bootstrap problem.
- **High education cost:** Adopting SSH public key authentication demands users to grasp its fundamentals, which can be more complex compared to password authentication. Thus, there is a higher educational barrier for users to overcome.

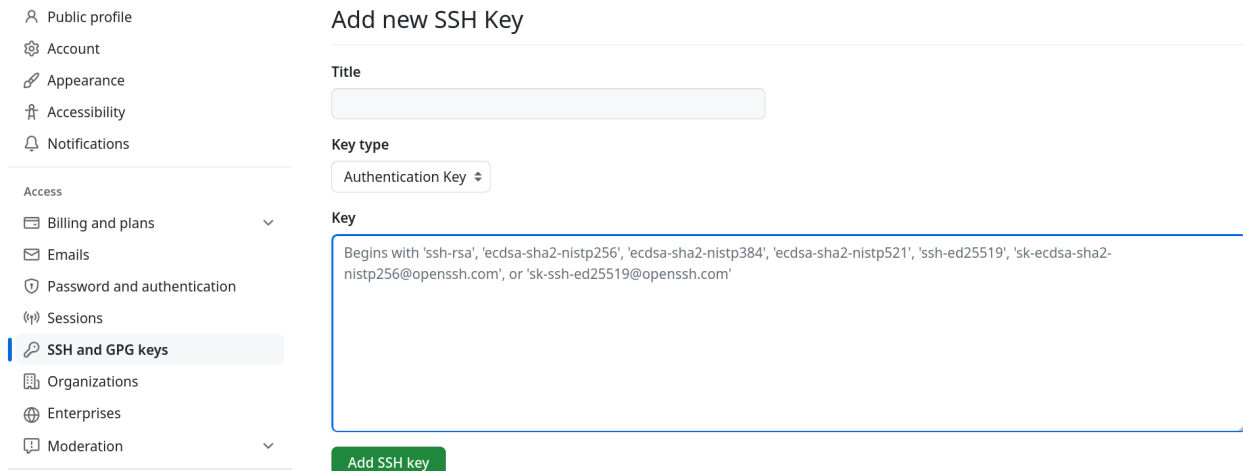
4.4 SSH Authentication with GitHub

GitHub, a popular platform for hosting Git repositories, also supports SSH authentication for secure access to repositories. Users can generate SSH keys and associate them with their GitHub accounts, allowing them to authenticate to GitHub servers securely without needing to enter their username and password each time they interact with a repository.

One can upload his/her public key in Github's web UI, as illustrated in fig. 12.

After that, one can access their git repository through an SSH link, which will use SSH authentication. This application is different from the original SSH in that it only uses SSH for authentication and a secure connection layer, but a shell is not available.

According to Github's recommendation, this SSH public key authentication is recommended over password authentication and they disable support for password authentication over HTTP in favor of SSH.



The screenshot shows the GitHub interface for adding a new SSH key. On the left is a navigation sidebar with options like 'Public profile', 'Account', 'Appearance', 'Accessibility', 'Notifications', 'Access', 'Billing and plans', 'Emails', 'Password and authentication', 'Sessions', 'SSH and GPG keys' (highlighted), 'Organizations', 'Enterprises', and 'Moderation'. The main content area is titled 'Add new SSH Key' and contains the following fields:

- Title:** An empty text input field.
- Key type:** A dropdown menu currently showing 'Authentication Key'.
- Key:** A large text area containing the text: 'Begins with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-ecdsa-sha2-nistp256@openssh.com', or 'sk-ssh-ed25519@openssh.com''.
- Buttons:** A green 'Add SSH key' button at the bottom left of the form.

Figure 12: Adding an SSH public key in Github.

References

- [BY23] Di Bao and Lin You. Two-factor identity authentication scheme based on blockchain and fuzzy extractor. *Soft Computing*, pages 1–13, 2023.
- [DRS04] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *Advances In Cryptology-EUROCRYPT 2004: International Conference On The Theory And Applications Of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004. Proceedings 23*, pages 523–540. Springer, 2004.
- [DTL13] Krishna Dharavath, Fazal A Talukdar, and Rabul H Laskar. Study on biometric authentication systems, challenges and future trends: A review. In *2013 IEEE international conference on computational intelligence and computing research*, pages 1–7. IEEE, 2013.
- [LY06] Chris M. Lonvick and Tatu Ylonen. The Secure Shell (SSH) Authentication Protocol. Request for Comments RFC 4252, Internet Engineering Task Force, January 2006. Num Pages: 17.
- [MŘ10] Vashek Matyáš and Zdeněk Říha. Security of biometric authentication systems. In *2010 International Conference on Computer Information Systems and Industrial Management Applications (CISIM)*, pages 19–28. IEEE, 2010.
- [MS09] Kevin D Mitnick and William L Simon. *The art of intrusion: the real stories behind the exploits of hackers, intruders and deceivers*. John Wiley & Sons, 2009.
- [O’C13] Kevin O’Connor. *Examination of stability in fingerprint recognition across force levels*. PhD thesis, Purdue University, 2013.

- [RYKH21] Riseul Ryu, Soonja Yeom, Soo-Hyung Kim, and David Herbert. Continuous multimodal biometric authentication schemes: a systematic review. *IEEE Access*, 9:34541–34557, 2021.
- [Sca19] ScatteredSecrets.com. How to crack billions of passwords?, 2019.
- [SDNC22] Shilpi Barman Sharma, Ishika Dhall, Soumya Ranjan Nayak, and Pushpita Chatterjee. Reliable biometric authentication with privacy protection. In *Advances in Communication, Devices and Networking: Proceedings of ICCDN 2021*, pages 233–249. Springer, 2022.
- [TBCC20] Joshua Tan, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. Practical recommendations for stronger, more usable passwords combining minimum-strength, minimum-length, and blocklist requirements. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1407–1426, 2020.
- [TJYW06] Alex Tsow, Markus Jakobsson, Liu Yang, and Susanne Wetzal. Warkitting: the drive-by subversion of wireless home routers. *Journal of Digital Forensic Practice*, 1(3):179–192, 2006.
- [WZW⁺16] Ding Wang, Zijian Zhang, Ping Wang, Jeff Yan, and Xinyi Huang. Targeted online password guessing: An underestimated threat. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 1242–1254, 2016.
- [ZZJ⁺18] Heng Zhang, Huan Zhou, Wenming Jiao, Jie Shi, Qiyan Zang, Jing Sun, and Jian Zhang. Biological features de-identification in iris images. In *2018 15th international symposium on pervasive systems, algorithms and networks (I-SPAN)*, pages 67–71. IEEE, 2018.