Lecture Note 6: Authentication

Sai Ho Cheung, Weimin Chen, Junjie Ma

April 11, 2023

Summary

The lecture 6 discusses three authentication scenarios such as password authentication, biometric authentication, and public key authentication. The password authentication is the most common method to manage access control. If this account's password has leakage accidentally, the attacker can access the user information in the remote system. Biometric authentication uses biological traits such as fingerprints, faces, voices, retinas, etc. This mechanism has implemented the two-factors authentication method and can prevent commonly-known attacks effectively. At last, this lecture introduces public key authentication taking the SSH authentication as an example.

1 Authentication

Authentication is a critical process that involves verifying the identity of a person or system. It is the act of proving an assertion, such as the identity of a computer system user. The core challenge for authentication is answering the question, "How do you prove to someone that you are who you claim to be?"

1.1 Authentication Factor

Authentication factors are the different types of information or credentials that are used to verify the identity of a user or system. There are three main types of authentication factors:

- Something you know (Password Authentication): This factor involves knowledge of a specific piece of information, such as a password.
- Something you are (Biometric Authentication): This factor involves biological characteristics, such as fingerprints or other biometric data.
- Something you have (Public key Authentication): This factor involves possession of a physical device or token, such as a phone, securID or cryptographic secret key.

The Shannon entropy is used for the measurement of the randomness of the authentication factors. It can be used to quantify the strength of the authentication factors. A strong authentication factor has a high entropy value, indicating that it is difficult to guess or crack, while a weak authentication factor has a low entropy value, making it vulnerable to brute force attacks. The Shannon entropy is calculated based on the number of possible values or combinations of the authentication factors used in a system. $H(x) = -\sum p(x) \log p(x)$ is the formula for Shannon entropy, where x represents a discrete random variable, and $p(x)$ is the probability of the occurrence of each possible value of x.

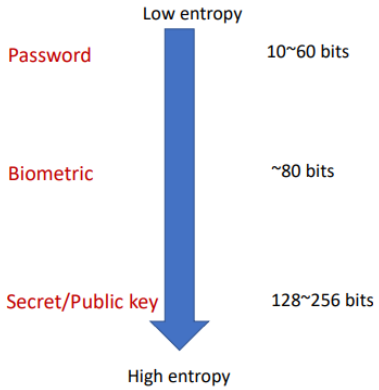

Figure 1: The Shannon entropy of the three types of authentication factors

The Figure 1 illustrates the Shannon entropy of the three main types of authentication factors. It can be observed that biometric authentication has higher entropy than password authentication due to the biometric data is unique to each individual and more difficult to replicate or guess. Public key authentication, which uses cryptographic keys generated by strong cryptographic algorithms, can have the highest entropy among the three authentication factors.

1.2 The difference between Authentication, Authorization and Access control

Authentication, authorization, and access control are three concepts in computer security that are used to restrict access to resources, each with its own distinct meaning and purpose.

- Authentication is the process of verifying the identity of a user or system. It involves proving that the user or system is whom they claim to be by requiring one or more authentication factors, such as a password, biometric data, or digital certificate. Authentication is the first line of defense against unauthorized access to a system or resource, and is critical in ensuring the confidentiality, integrity, and availability of data and services.
- Authorization is the process of determining whether a user or system has the necessary permissions or privileges to access a particular resource or perform a particular action. Authorization decisions are **based on the authentication** of the user or system. Authorization is a critical security mechanism that helps prevent unauthorized access and protects sensitive data and resources.
- Access control is the mechanism that enforces the policies and rules governing access to resources based on authorization decisions. It involves setting up rules and policies that determine who can access a resource, what actions they can perform, and under what conditions. Access control helps ensure that only authorized users or systems are able to access resources and perform actions on them, and is a critical component in ensuring the security and compliance of a system or organization.

1.3 Authentication Paradigm

The authentication paradigm refers to the method used to verify the identity of a user. The Figure 2 provides an overview of the general authentication paradigm. The paradigm typically involves a generation algorithm $Alg.G$ that creates a pair of keys, sk and vk, which can be a password, biometric data, or a private/public key, depending on the specific authentication factor. The sk

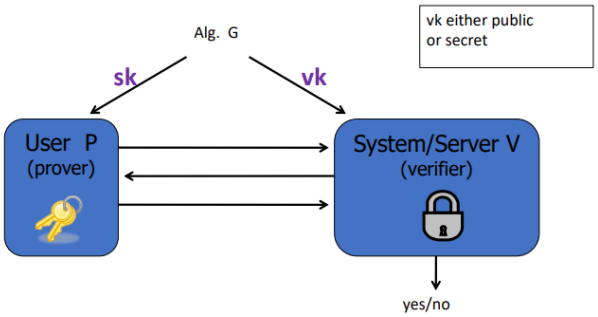

Figure 2: The overview of general Authentication Paradigm.

is assigned to the user P, while the vk is delivered to the system V. An interactive protocol is conducted between the user and the system to authenticate the identity of the user, and finally, the system V will determine whether to accept or reject the authentication of the user P.

2 Password Authentication

Password authentication is a commonly used method of authentication where a user is required to provide a secret password to prove their identity. The user selects a password, which is then stored in a database by the system. In the password authentication paradigm, as shown in Figure 3, the generation algorithm $Alg.G$ creates the password pw, which is stored by both the user P and the system V in their storage. To authenticate the identity of the user, the user sends his password pw to the system, and finally, the system will determine whether to accept or reject the authentication of the user based on whether the entered pw matches the stored pw.

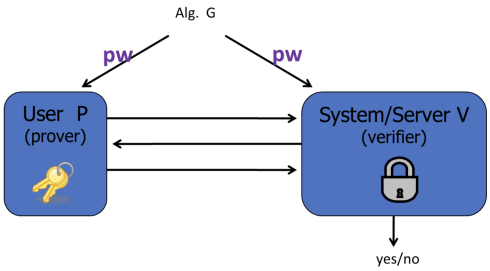

Figure 3: The overview of Password Authentication Paradigm.

2.1 Measure the Password Strength

Though password authentication is easy to use, many users do not use it properly. According to two surveys [TJYW06, MS09], half of the users use the default password of their routers and 99% of Dixie bank employees use the password "password123". Additionally, the RockYou leaked password dataset, shown in Figure 1, reveals that over 1% of Rockyou users chose "123456" as their password.

In order to evaluate what is a secure password, we need to measure the strength of the password. We first use the Shannon Entropy. Let X be password distribution and n is size of the support of X. p_1, p_2,..., p_n are probabilities of passwords in decreasing order. Then Shannon Entropy is defined as $H(X) = -\sum p(p_i) \log p(p_i)$.

Table 1: Top 10 Rockyou password

Rank	Passwd	Number of Users with Password (absolute)
1	123456	290731
2	12345	79078
3	123456789	76790
4	Password	61958
5	iloveyou	51622
6	princess	35231
7	rockyou	35231
8	1234567	22588
9	12345678	20553
10	abc123	17542

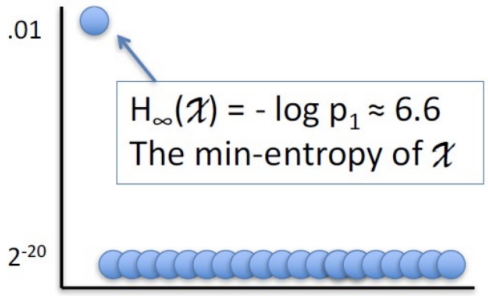

Figure 4: Min_entropy of the password

However, the Shannon entropy can be a poor measure for the password strength. For instance, assuming we have a total of 1,000,00 passwords, the first password probability p_1 is 0.01 and the rest password probability p_2 to $p_n = (1 - 0.01)/999,9999 \approx 1/220$. According to the Shannon entropy, the entropy of these passwords $H(X) \approx 19$. However, an adversary may only need to guess the first password, which has a probability of 0.01.

A better way to measure the strength of passwords is the min-entropy, which takes into account the commonness of the most popular passwords. The min-entropy is defined as $H_\infty(X) = -log_2 \max_{x \in X} p(x)$. As shown in Figure 4, the min-entropy for the set of passwords is approximately 6.6, which means the guessing probability (GP) for the most probable password over a population is $2^{-H_\infty(X)} \approx 0.9\%$. For instance, the GP for the most common password in the Rockyou dataset, "123456," is approximately 0.9%, meaning that around 1 in 111 users have this password. Therefore, the GP can be used to assess the vulnerability of the weakest accounts, which are more likely to be targeted by attackers.

To help users create stronger passwords, system administrators often require passwords to exceed a certain length, contain at least a specific number of character classes, or not appear on a blocklist, recent paper [TBCC20] also suggests the password should meet the rule of 1c12+NN10. The 1c12 means 1 class with at least 12 characters and NN10 required passwords to have password strength estimates no weaker than 10^{10} guesses.

2.2 Storage of Password

The storage of passwords within a system is a critical aspect of password authentication. The most intuitive way to store passwords is to create a table that stores the username and its corresponding password, as shown in Table 2. However, this method is strongly not recommended, as once an attacker gains access to the table, they will learn all users' passwords and may be able to attack

Table 2: User table with different storage of password

Username	Password in Plaintext	Password in Hash	Salt	Password in Hash with Salt
alice	password	XohlmNooBHFR0OVvjcYpJ3NgPQ1qq73WKhHvch0VQtg=	ciMTj87Q5Ti/PDfSUM4jCAT6cFJWVwJFjEbMc2sqAn0=	AQAiFDIbEUk5Wdoe6tTL+bnCBOIsectOW2SfftGOje8=
bob	hunter2	9S+9MrKzuG/4jvbEkGKChfSCrxXdyylUH5S89Saj9sc=	NB9zdy/OIVnGHKPK7fK01saCclpXrWV5rdtW8i5k/XY=	uxlXXvfrQ8/gTwrbTtgnsqsZCAw/y2408nU3qlho5GE=
charlie	correct-battery-horse-staple	0mk89QsPD4FIJQv8lcHnoSe6qjOzKvcNuTevydeUxWA=	hetbWcTifseB9K3IQQPr6c/eMJyj3kVTqq/l+FqYf78=	FykuFcJV0AjBLyxMuQWrvuSTjRXyXStitVteWUJmPIM=
dakotah	hunter2	9S+9MrKzuG/4jvbEkGKChfSCrxXdyylUH5S89Saj9sc=	IZu5hPamBS/QY4ILZzTcyVY8TK17Dt9hmXW7bC4XbCc=	ydVc+vA56bKbA0oXzRfYtkABUXaxgkF4ngB0xNJRvA4=

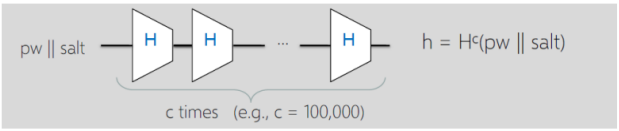

Figure 5: The PKCS#5 approach

their accounts on other sites if the user has reused their password across multiple sites.

In order to address this problem, we can hash the plaintext password and store the resulting hash value in the database. When the user provides their password pw, the system can compare the hash of pw with the stored hash value in the table, as shown in Table 2. Due to the existing cryptographic hash function has three important properties:

- One-way function: Given $y = H(M)$, hard to compute M.
- Deterministic: Function: H maps any message to a short digest (e.g., 256-bit string).
- Collisions resistant: It is hard to find M, M' that $H(M) = H(M')$.

However, simply hashing the password still poses two problems. The first problem is that users who have identical passwords will have identical hashes, which makes it easy for an attacker to spot. The second problem is dictionary attacks. Since the SHA256 hash function is quite fast to compute, an attacker can pre-compute the hash of every word in a dictionary and store them in the database known as the Rainbow table. Once the attacker finds a hash value that matches one in his Rainbow table, he can easily find the corresponding password.

The solution to these problems is to concatenate a salt and the password before hashing it. A salt is a fixed-length cryptographically strong random value. It ensures that identical passwords from different users are not revealed, since their salt values are different. Additionally, it makes pre-computed lookup attacks more difficult since an attacker would need to pre-compute hashes for a larger set of possible passwords. The example table of hashing with salt is shown in Table 2.

Another way to improve the password security is to make hashing slower to slow down cracking attacks. For example, the PKCS#5 approach (shown in Figure 5) involves hashing the password with a salt 100,000 times before outputting the final hash value. We can also use slower and memory-hard hash functions such as Scrypt or Argon2 to increase the attacker's cost.

2.3 Attack towards Password Authentication

There are two types of attacks toward the password authentication:

- Online attack: The attacker tries to guess passwords by logging to a live system.
- Offline attack: The attacker tries to guess passwords with given hash value from the password database.

In an online attack, the number of guess attempts allowed to log into the system is limited. However, a recent paper [WZW+16] shows that online attacks can be much more effective than

Figure 6: Biggest data breaches

Table 3: The compute power of a single RTX 2080Ti 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, 1,000 * MD5)	≈22M	≈5.6M	≈3-4 characters
Bcrypt (= salted, work factor 8)	≈3500	≈900	≈1-2 characters

previously thought. This is because people's password choices tend to vary greatly from one another, and passwords are often closely related to personal information (such as birthdays or other personal data). This is evident from the billions of leaked passwords shown in Figure 6. To check whether our password has been leaked, we can use the website `https://haveibeenpwned.com/`.

In the offline attack, an attacker can leverage the computing power of custom GPUs and FPGAs to build rainbow tables at a fast speed. The time it takes to build a rainbow table using a single RTX 2080Ti GPU is shown in Table3 [Sca22].

2.4 Multi-Factor Authentication

Besides using single password authentication to secure the authentication process, we can adopt multi-factor authentication by adding more factors. For example, two-factor authentication (2FA) combines passwords with another way to authenticate the user, such as proof of ownership of the user's email address, telephone number (via SMS), device pin (via an authenticator app), or hardware token (such as a one-time-password token or a universal second factor (U2F) token)

Two-factor authentication (2FA) is an effective way to protect users from being attacked. According to a Microsoft report in March 2020, 99.9% of compromised accounts did not use multi-factor authentication. A Google report in February 2022 also showed that successfully using 2FA resulted in a 50% decrease in accounts being compromised.

2.4.1 SMS (short message service) Authentication

For example, the key idea behind SMS (short message service) authentication is shown in Figure 7. After the user enters his password, the system generates an auth code and sends it to the user's phone via SMS. The user then sends the auth code back to the system to authenticate their identity. If an attacker tries to attack an account protected by SMS-based 2FA, he needs to know the authe code to log in to the account.

However, there are still some ways to circumvent SMS-based 2FA, including:

- The attacker can gain physical access to the device that receives SMS.
- SIM swap: The attacker can use social engineering to trick the phone company into registering the victim's phone as the attacker's device.

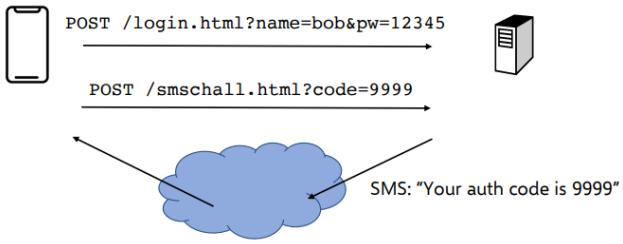

Figure 7: The process of SMS Authentication

- Phishing attacks: the attacker can confuse or trick the user into disclosing the SMS code to the attacker.

Sadly, usability remains a key issue preventing adoption, according to a report [ONG18] for over 90 percent of Gmail users still do not use two-factor authentication.

2.4.2 Time-based One-Time Passwords

In the time-based one-time passwords, the user's device generates a one-time password (OTP) that changes every 60 seconds or every button press, based on a shared secret key. The user enters this OTP along with their password to complete the authentication process. As shown in Figure 8, both the user and system hold a random KEY and a counter value at the beginning. To authenticate the user's identification, each time the user sends a 6-digit value v generated from the function F based on the KEY and the counter to the system, the counter value will be incremented based on time. The system will then verify whether v is equal to the value generated by the system.

The function F used in time-based one-time passwords must be a secure pseudorandom function (PRF) to ensure the protocol's security against eavesdropping. For instance, RSA SecurID uses a custom PRF that combines a 64-bit key with a 24-bit counter value to produce a 6-digit output.

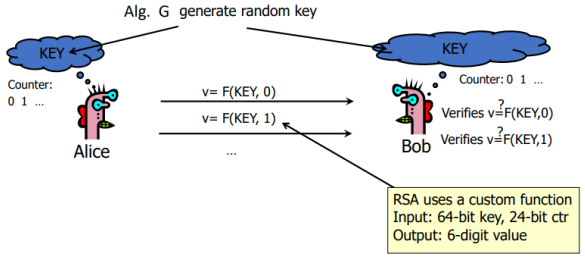

Figure 8: The process of Time-based One-Time Passwords

3 Biometric Authentication

Biometric authentication refers to a crypto process that verifies a user's identity using their unique biological traits such as fingerprints, face, voices, retinas, etc. Figure 9 shows the overview of Biometric Authentication. We assume the system stores the biometric trait before. The user P can provide his biological traits to the verfier in order to prove his identification. Then system can then decide whether to accept the user's access.

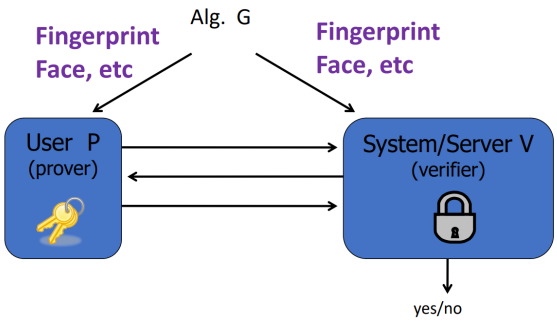

Figure 9: The overview of Biometric Authentication.

3.1 Biometric Error Rates

Biometric authentication is not widely used until twenty years ago due to the error rates. Here are two ways to define the error rate:

- Fraud error rate. system accepts a forgery (false accept)
- Insult error rate. system rejects valid user (false reject)

There is bar to balance between Fraud error rate and Insult error rate. For example, a higher acceptance threshold will increase fraud rate and decrease insult rate; a lower acceptance threshold will decrease fraud rate and increase insult rate. Next, we analyze the error source and find out the solution to optimize both fraud rate and insult rate.

Error Rate is mainly due to the instability of Bio-feature. Biometric trait must be converted into feature as the representation of the authentication system. The biometric features extracted cannot be totally the same in every time, although there are close inside the feature space. That is what we call the instability of Bio-feature.

3.2 Fix Error Rates via fuzzy extractor.

One solution to fix Error Rates is using fuzzy extractor. We assume there are two biometric features such as w and w' extracted from the Biometric trait. The fuzzy extractor function (FE) can find out whether $FE(w) = FE(w')$, when $w \neq w'$ but close to w'.

We collect the fuzzy extractor from [DRS04]. Here are the details. Definition 1. An (M, m, l, t, ϵ)-fuzzy extractor is a pair of randomized procedures, "generate" (Gen) and "reproduce" (Rep), with the following properties:

- The generation procedure Gen on input $w \in M$ outputs an extracted string $R \in \{0,1\}^l$ and a helper string $P \in \{0,1\}^*$
- The reproduction procedure Rep takes an element $w' \in M$ and a bit string $P \in \{0,1\}^*$ as inputs. The correctness property of fuzzy extractors guarantees that if $dis(w, w') \leq t$ and R,P were generated by $(R, P) \leftarrow Gen(w)$, then $Rep(w', P) = R$. If $dis(w, w') > t$, then no guarantee is provided about the output of Rep.
- The security property guarantees that for any distribution W on M of min-entropy m, the string R is nearly uniform even for those who observe $P : if (R, P) \leftarrow Gen(W)$, then $SD((R, P), (U_l, P)) \leq \epsilon$. $SD(A, B)$ is the statistical distance between A and B.

A fuzzy extractor is efficient if Gen and Rep run in expected polynomial time. In other words, fuzzy extractors allow one to extract some randomness R from w and then successfully reproduce

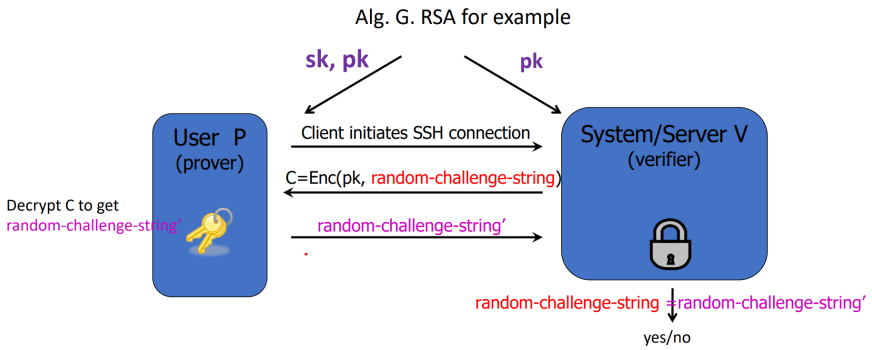

Figure 10: The overview of SSH Authentication.

R from any string w' that is close to w. The reproduction uses the helper string P produced during the initial extraction; yet P need not remain secret, because R looks truly random even given P. To justify our terminology, notice that strong extractors can indeed be seen as "nonfuzzy" analogs of fuzzy extractors, corresponding to $t = 0, P = X$, and $M = \{0,1\}^n$.

We reiterate that the nearly uniform random bits output by a fuzzy extractor can be used in any cryptographic context that requires uniform random bits (e.g., for secret keys). The slight nonuniformity of the bits may decrease security, but by no more than their distance ϵ from uniform. By choosing ϵ negligibly small, one can make the decrease in security irrelevant.

Similarly to secure sketches, the quantity $m - l$ is called the entropy loss of a fuzzy extractor. Also similarly, a more robust definition is that of an average-case fuzzy extractor, which requires that if $H(W \mid I) \geq m$, then $SD((R, P, I), (U^l, P, I)) \leq \epsilon$ for any auxiliary random variable I.

3.3 Discussion

Here are the advantages of the biometric authentication.

- Nothing to remember. Biometric Authentication uses WHAT WE ARE rather than WHAT WE KNOW.
- Passive.
- Can't share (generally).

Here are the disadvantages of the biometric authentication.

- Private, but not secret. The biometric password are shared between multiple systems.
- Revocation is difficult. It is impossible to change a biometric password. Once the biometric password leaked, we have to replace another kind of biometric password.
- Birthday paradox. Attackers may exploit the password via brute force[BEN21]. With false accept rate of 1 in a million, probability of false match is above 50% with only 1609 samples.

Due to these problems, biometric authentication primarily should be used as a second factor authentication, rather than a primary authentication factor.

4 SSH Authentication

Authenticated key exchange is a kind of public key authentication. This lecture will focus on SSH.

4.1 SSH Authentication

The SSH protocol uses encryption to secure the connection between a client and a server. All user authentication, commands, output, and file transfers are encrypted to protect against attacks in the network[Ylo22]. It was designed to protect remote login sessions rather than key exchange. Thus no Public key infrastructure is required. In particle, client should generates the public/secret key locally, and then upload public key to server and store secret key on the local device.

4.2 A Simplified SSH Authentication

Figure 10 shows the overview of a simplified SSH Authentication using a pair of private key and public key generated by a local user. There are three steps. First, the client initiates a SSH connection by contacting the server. Second, the server encrypts a random challenge using the public key and then send the encrypted challenge to the client. Third, the client decrypts the receiving challenge with the private key and sends the the plaintext to the server. Once the server validate the the challenge, the client can access the server.

4.3 Discussion

Here are the advantages of the SSH authentication.

- SSH keys are more difficult to hack than passwords and thus are more secure.
- SSH keys aren't human generated, so you'll avoid having easy-to-guess keys like "123456" or "password".
- Unlike passwords, your private SSH key isn't sent to the server.

Here are the disadvantages of the SSH authentication.

- the private key needs to be stored on the device.
- distribution of public keys and education of staff on how to use SSH keys can be more cumbersome.

4.4 SSH Demo on Github

Here we show how to use SSH to login Github.

Generating a new SSH key. As shown in Figure 11, we generate a new SSH key on the local machine. `ssh-keygen -b 256 -t ecdsa -f id_ecdsa` I leave a blank passphrase. Then I can obtain the public key via `cat id_ecdsa.pub`.

Adding the SSH key to Github. As shown in Figure 12, in the upper-right corner of any Github page, we click the profile photo, then click Settings. In the "Access" section of the sidebar, we open the "SSH and GPG keys", where I can add a new SSH key. That is the public key we generate in previous step, i.e., "id_ecdsa.pub".

Accessing Github with SSH Key. Then I can download the Github repository via `git clone git@github.com:haiyangxc/hyxue.git`, if I loaded my SSH key to haiyangxc's Github account.

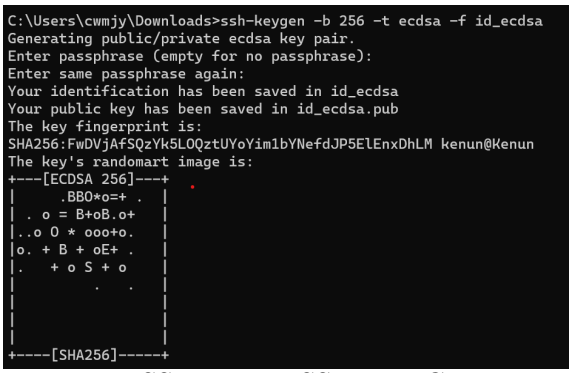

Figure 11: SSH Demo: SSH Key Generation.

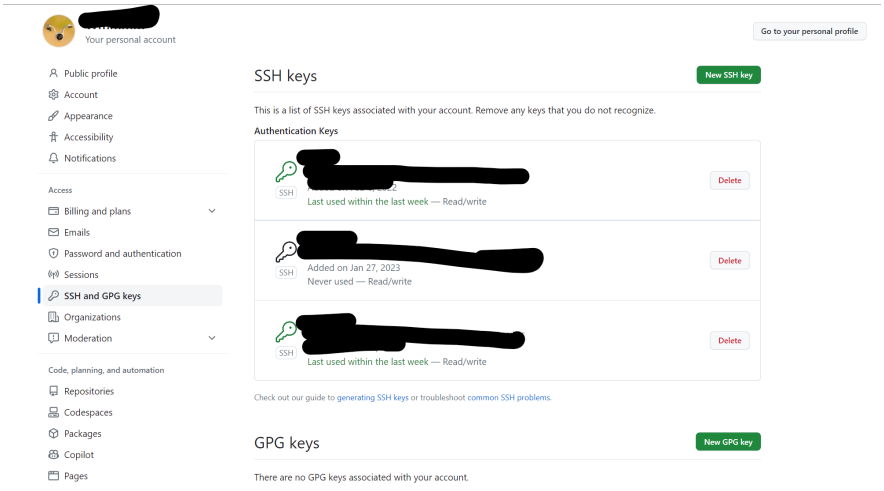

Figure 12: SSH Demo: Adding SSH Key. Sensitive information are hidden.

References

[BEN21] Toras Pangidoan Batubara, Syahril Efendi, and Erna Budhiarti Nababan. Analysis performance bcrypt algorithm to improve password security from brute force. In *Journal of Physics: Conference Series*, volume 1811, page 012129. IOP Publishing, 2021.

[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.

[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.

[ONG18] THUY ONG. Over 90 percent of gmail users still don't use two-factor authentication, 2018. [Online]. Available: https://www.theverge.com/2018/1/23/16922500/gmail-users-two-factor-authentication-google.

[Sca22] ScatteredSecrets.com. How to crack billions of passwords?, 2022. [Online]. Available: https://scatteredsecrets.medium.com/how-to-crack-billions-of-passwords-6773af298172.

[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 Wetzel. 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.

[Ylo22] Tatu Ylonen. Ssh authentication, 2022. [Online]. Available: https://www.ssh.com/academy/ssh.