Authentication (Entity Authentication)

Authentication
Authentication is the process of assuring that the communicating entity or the origin of a piece of information is what it claims to be.

There are thus two types of authentication:

Entity Authentication
This is for connection-oriented communication, where the communicating entity is an entity involved in a connection. Some common mechanisms for this type of authentication are thus password, challenge and response.

This may be required to ensure authenticity over various communication channels:

- Phone call from the Police Department: Is this authentic?
- Logging in to LumiNUS via my account: Is the server that I'm interacting with authentic? Why would the LumiNUS' server be convinced that the entity logged in is the authentic me?
- Connecting to the "NUS" WiFi: Was that WiFi access point authentic?

Data-origin Authentication
This is for connectionless communication, where the communicating entity is the origin of a piece of information. Data-origin authenticity implies data integrity. Some common mechanisms are MAC and digital signature.

This may be required to ensure authenticity of digital data and physical documents:

- Submitted a MC for a test: Is this MC authentic?
- Obama's birth certificate: Is it authentic?
- Email from lecturer saying that the quiz is cancelled: Is the email authentic?

Authenticity vs Integrity
Authentic is an adjective to say that the claimed entity/origin is assured by supporting evidence. Authenticity is the condition of being authentic.

Authenticity and integrity are thus related. In the context of an insecure channel, we can say that a message that has been modified in transit means that it no longer comes from its original source.

In other words, a message whose integrity is compromised also means that its authenticity is compromised. As such, **data-origin authenticity implies data integrity.** But **data integrity does not imply data-origin authenticity.** Authenticity is thus a stronger requirement than integrity.

<u>Password</u>
A password is part of an authentication system that usually consists of:
- Identity (Identification)
    - The identity need not to be kept secret
    - It can be a username in a system, bank account number, customer ID etc.
- Password (Authentication)
    - The password is kept secret
    - Only the authentic user and the server knows it
    - The fact that an entity knows the password **implies that it is either the server or the authentic user**

The process can be broken down into:

Process	Provided by	To Answer	Attributes	Uniqueness Requirement
Identification	Principal	"Who are you"	Public assertion	Yes (locally)
Authentication	Principal	"How can you prove that it is you?"	Secret response	No
Authorization	System	"What can I do?"	Token/Ticket, Access Control	-

Authorization goes more into access control, which basically has the system determining what the user can or cannot do.

There are two stages to the password authentication system:

<u>Stage 1: Bootstrapping</u>
During bootstrapping, the server and the user establishes a common password. The server then keeps track of a file recording the identity (i.e. userid, username) and the corresponding password.

A quick and painless, or even no bootstrapping process can be good, but care must be taken so that the resultant password is still effective.

The password establishing can be done by either:
- The server / user chooses a password and sends it to the user / server through another communication channel. For example, NUSNET sends the password via physical mail.
- A default password is set

<u>Stage 2: Password-based Authentication</u>
This is the authentication that most of us are used to.
User → Server: Hanming is my username.
Server → User: Ok, Hanming. What is your password?
User → Server: hellohello is my password.
Server: Ok. You are indeed Hanming.

Alternatively, it could be through a SMS to a server:
userid: hanming@nus.edu.sg password: hellohello instruction: unsubscribe

What the system does is to check against its password file to find the username and password.

Weak Authentication System
The password system is classified as a weak authentication system. A weak authentication system is one that is vulnerable to **replay attack**: information sniffed from the communication channel can be used to impersonate the user at a later time.

This is in contrast with strong authentication, where information sniffed during the process cannot be used to impersonate the user.

Attacks on the Password System

1. Attack the Bootstrapping

It is possible for an attacker to attack the bootstrapping. They can target the use of default passwords (real example: IP Security Cameras).

They can also aim to intercept the password in the process of being sent via an alternative channel (e.g. stealing the postal mail containing the NUSNET password).

Why do manufacturers not use individual passwords?
For the device manufacturer – cost will increase since there is a need to:
- print the password on each equipment/manual/case
- record the passwords in a password database
- have a customer service line or Web site to help users retrieve their device's default password.

For device users – usability will decrease if a user is unable to find or retrieve the default password. This may lead to a bad product review, and also affect manufacturer's sales.

2. Searching for the Password

<u>Guessing</u>
It is not uncommon for people to set passwords that are based off certain information about themselves and their loved ones, for the greater ease of remembering. An attacker can thus gather information about the user and attempt to guess the password.

There are two types of password guessing:

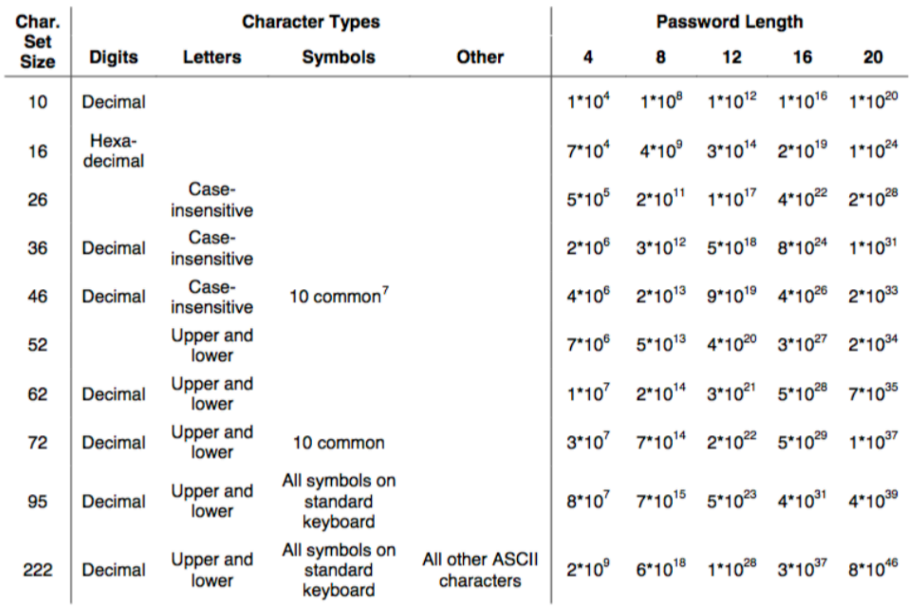
Online Password Guessing
In an online version of the password guessing attack, an adversary tries to guess a password by logging to a server. This version of the attack is less powerful than the offline version since the adversary is limited by maximum allowed login attempts, whereby no such limitations exist in the offline version.

Offline Password Guessing
In the offline version an adversary gets in the possession of some password-related data of a user (e.g. hashed password) and thus iteratively tries to guess a password and verify its hashed version with the intercepted one. In this version of the password guessing attack, the adversary is only limited with the processing power of its own machine, meaning there exists no other lock to stop him/her from trying to guess the password, since the attack is done locally on the adversary's machine. Because hardware is getting ever more powerful according to Moore's Law, the adversaries can use such enhanced power for more guessing attempts per second.

<u>Exhaustive Attacks</u>
Exhaustive search is as previously introduced. The attacker can try all possible combinations and see if any of them works. However, this can take a great amount of time, as there are passwords of different lengths and character sets.

This makes exhaustive search on passwords not very feasible. There is also no reason to try some combinations, given the low probability of the common user using certain characters.

Table 3-1. Possible Keyspaces by Password Length and Character Set Size



Char. Set Size	Character Types				Password Length				
	Digits	Letters	Symbols	Other	4	8	12	16	20
10	Decimal				$1*10^{4}$	$1*10^{8}$	$1*10^{12}$	$1*10^{16}$	$1*10^{20}$
16	Hexa-decimal				$7*10^{4}$	$4*10^{9}$	$3*10^{14}$	$2*10^{19}$	$1*10^{24}$
26		Case-insensitive			$5*10^{5}$	$2*10^{11}$	$1*10^{17}$	$4*10^{22}$	$2*10^{28}$
36	Decimal	Case-insensitive			$2*10^{6}$	$3*10^{12}$	$5*10^{18}$	$8*10^{24}$	$1*10^{31}$
46	Decimal	Case-insensitive	10 common[7]		$4*10^{6}$	$2*10^{13}$	$9*10^{19}$	$4*10^{26}$	$2*10^{33}$
52		Upper and lower			$7*10^{6}$	$5*10^{13}$	$4*10^{20}$	$3*10^{27}$	$2*10^{34}$
62	Decimal	Upper and lower			$1*10^{7}$	$2*10^{14}$	$3*10^{21}$	$5*10^{28}$	$7*10^{35}$
72	Decimal	Upper and lower	10 common		$3*10^{7}$	$7*10^{14}$	$2*10^{22}$	$5*10^{29}$	$1*10^{37}$
95	Decimal	Upper and lower	All symbols on standard keyboard		$8*10^{7}$	$7*10^{15}$	$5*10^{23}$	$4*10^{31}$	$4*10^{39}$
222	Decimal	Upper and lower	All symbols on standard keyboard	All other ASCII characters	$2*10^{9}$	$6*10^{18}$	$1*10^{28}$	$3*10^{37}$	$8*10^{46}$

<u>Dictionary Attacks</u>
This is a form of exhaustive search where the attacker can restrict the search space to a large collection of probable passwords. These words tend to be words from the English dictionary, known compromised passwords, dictionaries from other languages etc.

It is thus possible to carry out a hybrid attack, i.e. a mix of exhaustive search and dictionary attack. For example, we could try all combinations of 2 words from the dictionary, and exhaustively try all possible capitalizations of each word, substituting ‘a’ with ‘@’, etc.

3. Stealing the Password

<u>Eavesdropping: Sniffing and Keylogging</u>
The most primitive method is the shoulder surfing, also known as the look-over-the-shoulder attack. Other ways include the interception or sniffing of communication channels, as some systems and protocols simply send the password over a public network in clear (i.e. unencrypted). For example, FTP, Telnet and HTTP do that.

It is also possible to sniff a wireless keyboard by picking up on the signals transmitted, or even via listening to the sound of the keyboard!

A keylogger captures/records the keystrokes and sends the information back to the attacker via a “covert channel”. This can be done via both software and hardware. For example, some computer viruses are designed to capture keyboard inputs. There are also hardware keyloggers that are plugged in and intercepts the transmissions by the keyboard.

<u>Phishing</u>
Phishing attacks ask for the user’s password under some false pretense. The user is then tricked to voluntarily send the password to the attacker over the network. This is typically done through emails, but can also be carried out over phone calls.

Spear phishing is the process of targeting a particular small group of users and is an example of targeted attacks. This can make the phishing attempt more realistic and believable, and much more effective.

Phishing attacks are basically social engineering attacks, which, in the context of information security, refers to the psychological manipulation of people into performing actions or divulging confidential information.

Some prevention means are to educate users via phishing drills. There are also phishing repository sites out there. However, it can be very tricky to accurately determine if an unsolicited email is a phishing attempt.

<u>Login Spoofing</u>
Attackers can also display “spoofed” login screens to trick users. To prevent this from happening, some systems have a **secure attention key** or **secure attention sequence** (e.g. Ctrl + Alt + Del for Windows). When this sequence is pressed, the system starts the trusted login processing.

<u>Password Caching</u>
When using a shared workstation, information keyed in may be cache. The next user can then see the cache. To prevent this, clear the browser’s cache and close the browser when using a shared workstation.

<u>Insider Attacks</u>
An example of this would be a malicious system admin who steals the password file. Another possibility is the compromising of a system admin’s account, leading to the loss of the password file.

Preventive Measures

1. User Side

Use a Strong Password

The most direct method to reduce the chances for an attacker is to improve the strength of the password. One way is to make sure the password is randomly chosen, perhaps using an automated password generator.

There is, however, always a trade-off between security and convenience. A password with really "high entropy" is often very difficult to remember.

Some methods to strengthen passwords:

- Mnemonic Method: Pbmbval! (Please be my best valentine!)
- Altered Passphrases: Dressed*2*tge*9z
- Combining and Altering Words: B@nkC@mera

Password Ageing

It is often recommended for users to regularly change passwords. However, many believe that frequent changes of passwords actually reduce security, due to people following some standard transformations e.g. increasing numbers, adding extra digits etc.

2. Admin/Server Side

Limited Login Attempts

The admin can add delays in the login process, security questions, and can also lock the account after a few failed attempts. All these reduce the efficiency of the attacks.

Password Checker

Check for weak passwords during registration or password changes.

Password Metering

Indicate weak, average and strong passwords.

Password Usage Policy

The organization can set rules to ensure that users use strong passwords, such as a minimum character count etc.

Protect the Password File

This is the important prevention measure for the admin/server side. As the password file stores usernames and passwords, any leakage due to insider attack, accidental leakage, system hacking etc. can be disastrous.

There have been many cases where weakly protected password files are leaked, leading to a large number of passwords being compromised. Hence, we need to add an additional layer of protection to the password file.

Password Hashing

The way to protect passwords is generally to hash passwords before storing them in the password files. This is different from encryption, as there is a way to recover the password from the ciphertext. However, for a cryptographically secure hash, it is not feasible to recover the password from its hashed value.

During authentication, the password entered by the entity is hashed, and compared with the value stored in the password file.

We also cannot have the same password being hashed to the same value for two different usernames. This allows attackers to obtain all un-hashed passwords of the same value simply by comparing hashed values. The way to prevent this is to add salt, i.e. a random string of characters to the front of the password before hashing it. This salt is randomly generated for all users and is also stored in the password file.

Security Questions
Security questions can be viewed as a mechanism for fallback mechanism or a self-service password reset. It enhances usability, as users can still login even if they have lost their passwords. It reduces costs, as it reduces the work of a helpdesk. However, this weakens security, since attackers now have another mean of obtaining access.

However, it can be difficult to provide a perfect security question. The criteria are as follows:

- Memorable
    - If users cannot remember their answers, then there is no point to having security questions
- Consistent
    - The user's answers should not change over time
    - For example, the answer to "What is your current job?" may change over time
- Nearly universal
    - The question should be applicable to a wide audience if possible
- Safe
    - The answer should not be something easily guessed or researched i.e. not publicly available information

An open-ended question can make a good question. It is also good to highlight that the user does not have to faithfully give the actual or right answer.

Nowadays, a second factor (e.g. SMS) or secondary email address is preferred over security questions for resetting passwords.

ATM and ATM Attacks
An ATM card and the PIN actually work very similarly to how a username and password would. To get authenticated, a user has to present a card and the PIN. The ATM card contains a magnetic stripe, which stores the user account ID. It essentially simplifies the process of inputting the account ID; instead of having to key it in, the user just has to insert the card. The PIN then plays the role of the password.

Data is encoded into the magnetic stripe using **well-known standards**. Given a valid card, an attacker can “copy” the card by reading the info from the card and write it to a spoofed card.

ATM Skimming
An ATM skimmer steals the victim’s account ID and PIN. A skimmer usually consists of a card-reader attached on top of an existing ATM reader, a camera overlooking the keypad or a spoofed keypad on top of the existing keypad, and some means to record and transmit the information back to the attacker.

With this method, the attacker can spoof the card and obtain the PIN.

Preventive Measures
To prevent the above from happening, we can:
- Install anti-skimming devices that prevent external card readers from being attached onto the ATM
- Shield the keypad
- Increase awareness of users
- Use newer chip-based (EMV) cards, which use encryption

Biometrics

Biometrics use unique physical characteristics of a person for authentication.

During enrollment: a reference template of a user's biometric data is constructed and stored, similar to bootstrapping in the password system.

During verification: the biometric sample data of the person-in-question is captured and compared with the template using a matching algorithm. The algorithm decides whether to accept or reject the authentication.

Biometrics can be used for both:
Verification: 1 to 1 comparison to determine whether the person is the claimed person
Identification: 1 to many comparison to identify the person from a database of many persons

Differences between Biometric and Password

- Biometrics cannot be changed, while a password can
- There is no need to remember anything for biometrics, while we need to remember our passwords
- There is a probability of error for biometrics, while there is zero non-matched rate for passwords
- It is not possible for a person to "transfer" the biometrics to another (in general), while users can pass their password to others

Matching Algorithm: Similarity / Inexact Matching

As there are inevitable noises in capturing biometric data, there will be errors in the matching decision. Based on these error rates, we get the following:

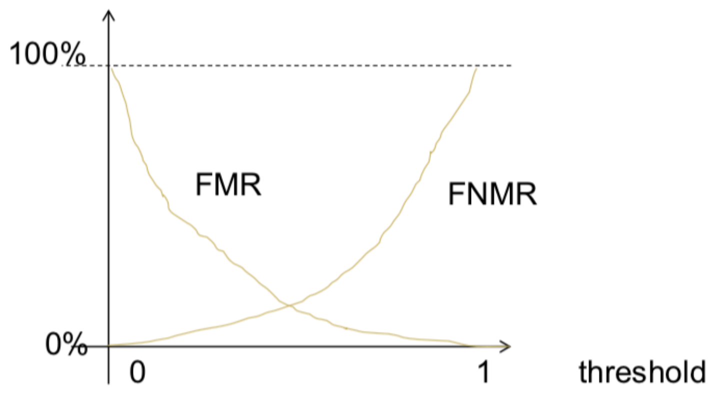
$$False\ Match\ Rate = \frac{Number\ of\ successful\ false\ matches\ (B)}{Number\ of\ attempted\ false\ matches\ (B + D)}$$

$$False\ Non - Match\ Rate = \frac{Number\ of\ rejected\ genuine\ matches\ (C)}{Number\ of\ attempted\ genuine\ matches\ (A + C)}$$

where A, B, C, D refer to the following:

	Accepted / Match	Rejected / Non-Match
Genuine Attempt	**A**	**C**
False Attempt	**B**	**D**

There is thus a threshold for the matching algorithm, which when adjusted, would adjust the FMR and FNMR as well. By having a lower threshold, our matching algorithm is more relaxed, while a higher threshold would result in a more stringent matching algorithm.



Other Types of Error and Rates
<u>Equal Error Rate (EER)</u>
The rate when FMR = FNMR

<u>Failure-to-enroll Rate (FER)</u>
Some users' biometric data may not be able to be captured for enrollment, due to reasons such as injury

<u>Failure-to-capture Rate (FTC)</u>
A user's biometric data may fail to be captured during authentication. For example, the finger of the user may be too dirty or too dry etc.

How Does Fingerprint Scanning Work?
What the scanner looks out for are feature points, which are basically edges or branches in the fingerprint. For a fingerprint, feature points are also known as minutiae.

The performance of the fingerprint as a biometric depends on the quality of the scanner, as EER can range from 0.5% to 5%.

Other Forms of Biometrics
Some other forms are palm prints, palm veins, hand geometry, face, iris, retina, DNA.

Security of a Biometric System
We assume the scanner to be secure, with no tampering possible. Even then, it may still be possible to spoof biometric data, similar to how movies show it. As such, some systems actually have "liveness detection" to verify if the entity scanned by the scanner is indeed "live" instead of being a spoofed material, e.g. photograph.

Multi-Factor Authentication

Multi-factor authentication, or n-factor authentication, requires at least 2 different authentication “factors”.

Some commonly used factors are:

1. What you know
    a. Password
    b. PIN
2. What you have
    a. Smart Card
    b. ATM Card
    c. Mobile Phone
    d. Security / OTP Token
3. Who you are
    a. Biometrics

Other possible factors are:

4. Where you are
5. What you do

It is called 2-factor authentication if 2 factors are required. The Monetary Authority of Singapore expects all banks in Singapore to provide 2-factor authentication for e-banking.

OTP Token

The One-Time Password (OTP) token is a hardware that generates a one-time password (i.e. a password that can only be used once). The server usually issues a OTP token to the user, which contains a secret key that the server knows. The user will also set a password to their account.

There are two ways for the OTP to be generated:

1. Time-based
    a. Based on the shared secret and current time interval, a password K is generated. Now both the server and the user has a common password K.
2. Sequence-based
    a. An event (e.g. the user presses the button) triggers the change of password

The authentication process is as such:

- User “presses” the token, which then computes and displays a one-time-password
- User sends username, password and OTP to server
- Since the server has the “secret key”, the server can also compute the OTP
- Server verifies that both the OTP and the password are correct

Authenticator Application

There is also a rising trend of using a mobile application as a “soft” OTP token. This can be a custom app or an authenticator app.

SMS – Is it secure?

It is also possible to register 2FA using SMS. To register, the user gives the server his mobile phone number and password.

The authentication process is as such:

- User sends their password and username to the server
- Server verifies that the password is correct and sends a one-time-password (OTP) to the user via SMS
- User receives the SMS and enters the OTP

- Server verifies that the OTP is correct

However, SMS OTP is **not secure**. The National Institute of Standards and Technology's (NIST's) "Digital Authentication Guideline" also recommended the deprecation of SMS OTP.

Some possible security threats are:
- Interception of cellular networks' channel
- SMS messages are stored as plaintext by the Short Message Service Center (SMSC)
- Malware / Trojan on smartphones
- Singapore government has access to all of our messages

However, it is acknowledged that doing SMS OTP is still better than just using username and password.

Smartcard + Fingerprint (Door Access System)
This is usually set up by having the server issue a smartcard to the user, with the smartcard containing a secret key K. The user then enrolls their fingerprint.

The authentication process is as such:
- User inserts smartcard into the reader.
- The reader obtains the user identity and verifies if the smartcard is authentic. If so, continue.
- User presents fingerprint to the reader.
- The reader performs matching to verify that the fingerprint is authentic. If so, grant access.

Very often, the information on the user identity, the secret key K, and the fingerprint template are not stored in the reader. The reader has a secure communication channel to a server that stores these information. We thus assume that the reader and server are secure, i.e. attackers are unable to access them.

A smart card has this security feature where even if an attacker has physical access to the card, it is extremely difficult, if not impossible, to extract a secret stored in the card.

Tutorial Learnings

Keeping UserID Secret
A system that checks for the existence of a userid and prompts the user specifically about it makes it easier to be attacked. Suppose an attacker needs x guesses for the userid, and y guesses for the password, in order to log in as a valid user. The attacker will thus need x + y guesses to get in, whereas he will need xy guesses on a system that prompts for both userid and password if the combination is incorrect.

Notice that, in practice, userid is not a secret. But for a layered defense, it is good to hide it as much as possible. Ultimately, the security still relies on the password.

Using Social Information for Authentication
This can be good in that there is little to no bootstrap process, and the information needed to login are easy to remember.

However, there is a potential privacy leak. If an attacker knows the information (which is not difficult), he can access the system. If he knows all but one piece of information, he can probe the login screen to figure out the last piece.

Ultimately, it is a trade-off between usability and security.

SMS – No Information or Complete Information?
M1: No information, e.g. "Enter OTP 132373 to complete your transaction."
M2: Complete information, e.g. "You have requested to transfer $10,000 from account no 1388293-43-23 to account no 12398-234-A2. Enter OTP 132373 to complete your transaction."

Firstly, a SMS is not encrypted in an "end-to-end" fashion, and there could be multiple telco entities handling the SMS. You can also consider the scenario where the PC is in an Internet cafe, and the cafe owner could be malicious, or honest but curious.

Secondly, notice that arguing "a system M1 is more secure than M2" requires us to find an attack that M1 can prevent, but not M2.

M1 is more secure: if the mobile phone is lost or somehow an adversary can get hold of the mobile phone, and the previous messages are not deleted, then the adversary can see the undeleted transaction details. This will lead to a privacy leakage. Alternatively, if the adversary can tap into the SMS communication channel, then he will able to capture the transaction history.

M2 is more secure: if a malware resides, or is purposely planted in the PC, then the message displayed to the user can be different from the transaction actually sent to the server. M2 can help the user verify if his/her money is transferred correctly.

But if we can decide on the message format, we can show partial account information instead, which is what the industry is doing today.