CS458: Introduction to Information Security

Notes 9: User Authentication

Yousef M. Elmehdwi

Department of Computer Science

Illinois Institute of Technology

yelmehdwi@iit.edu

April 1st 2024

Slides: Modified from "Cryptography and Network Security", 7/e, by William Stallings, Computer Security: Principles and Practice, 4th Ed, by: William Stallings and Lawrie Brown, & Steven Gordon

Outline

- A Story of Things Going Very Wrong
- User authentication
  - Password-based authentication
  - Token-based authentication
  - Biometrics authentication
- Remote User Authentication

The Weak Links: The case of Mat Honan August 2012[1]

MAT HONAN GEAR 08.06.12 08:01 PM

HOW APPLE AND AMAZON SECURITY FLAWS LED TO MY EPIC HACKING

Meet Mat Honan. He just had his digital life dissolved by hackers.

PHOTO: ARIEL ZAMBELICH/WIRED. ILLUSTRATION: ROSS PATTON/WIRED

[1] https://www.wired.com/2012/08/apple-amazon-mat-honan-hacking/

The Weak Links: The case of Mat Honan August 2012[2]

- 4:33 p.m., "Mat Honan" called AppleCare reporting lost me.com e-mail password. Apple issued temporary password.
  - "Mat" couldn't answer his own security questions.
  - Apple required only last four digits of a credit card and a billing address.
- 4:50 p.m.: Password reset e-mail arrived in Honan's me.com e-mail box; used to reset Honan's AppleID password
- 4:52 p.m.: GMail password recovery email arrived in Honan's me.com mailbox
- 4:54 p.m.: Honan's Google account password changed.

[2] https://www.wired.com/2012/08/apple-amazon-mat-honan-hacking/

The Weak Links: The case of Mat Honan August 2012[3]

- 5:00 p.m.: iCloud "Find My" tool used to wipe Honan's iPhone
- 5:02 p.m.: Honan's Twitter password reset
- 5:05 p.m.: Honan's MacBook wiped
- 5:10 p.m.: The real Honan calls AppleCare
- 5:12 p.m.: Hackers post message on Honan's Twitter account taking credit for the hack

[3] https://www.wired.com/2012/08/apple-amazon-mat-honan-hacking/

How did it happen?

- Attackers started by compromising Honan's Amazon account
- Needed credit card number for Honan's Amazon account. How did they learn it?
- Attackers called Amazon and **added** a new credit card number to Honan's account. (Name, email, and billing address sufficed.)
- Attackers called Amazon to reset Honan's password. For identity verification, Amazon asked for a credit card number...

How did it happen?

- Once logged into Honan's Amazon account, attackers learned last four digits of real credit card numbers
- *The very four digits that Amazon considers unimportant enough to display in the clear on the web are precisely the same ones that Apple considers security enough to perform identity verification.*

How did it happen?

- Then they called AppleCare...
- “It turns out, a billing address and the last four digits of a credit card number are the only two pieces of information anyone needs to get into your iCloud account. Once supplied, Apple will issue a temporary password, and that password grants access to iCloud.”

The result?

- Honan hadn't backed up his data.
- He lost all of it, e.g., irreplaceable photos young daughter
- Why did hackers wipe his devices?
  - Just to prevent his regaining control of accounts!
- Honan got in touch with one of the hackers, Phobia, via instant messaging?
  - Quoth Phobia: "yea i really am a nice guy idk why i do some of the things i do."
  - "idk my goal is to get it out there to other people so eventually every1 can over come hackers"
  - "even though i wasnt the one that did it i feel sorry about that. Thats alot of memories im only 19 but if my parents lost and the footage of me and pics i would be beyond sad and im sure they would be too."

How The Hackers Did It - Defeating The Hackers - BBC

- Watch: How The Hackers Did It - Defeating The Hackers - BBC
- Read: How Apple and Amazon Security Flaws Led to My Epic Hacking

Digital User Authentication

- National Institute of Standards and Technology (NIST) (Digital Authentication Guideline, October 2016) defines digital user authentication as:
  - *"The process of establishing confidence in user identities that are presented electronically to an information system."*

- *"The process of verifying a claim that a system entity or system resource has a certain attribute value".*[1]

- *Digital user authentication refers to the process of verifying the identity of a user accessing a digital system, application, or service. It is a critical security mechanism used to prevent unauthorized access and protect sensitive data and resources.*

[1] R. Shirey, "Internet Security Glossary, Version 2", IETF RFC4949

User Authentication

- User Authentication is proving your identity to a system (or another person).
  - *i.e., the process of verifying that a user is who he or she claims to be.*
- User authentication is primary line of defense in computer security; other security controls rely on user authentication

Two Steps of Authentication

- Typically two main steps in the process of digital user authentication:
- Identification Step (Identity verification):
  - In this step, the user provides some form of identification, such as a username, email address, or employee ID number, to the system or application.
  - This information is used to verify that the user is a legitimate user of the system or application.
    - Generally unique but not secret
- Verification Step (Authentication):
  - Once the user's identity has been verified, the user must provide some form of authentication to prove that they're the legitimate owner of the account.
  - This can include something the user knows, such as a password or PIN, something the user has, such as a security token or smart card, or something the user is, such as biometric data like a fingerprint or facial recognition.
    - Often secret or cannot be generated by others
- *Essentially, the user authentication process is what provides users repeat access to their own accounts while attempting to block any unauthenticated users from gaining access.*

Two Steps of Authentication

- *Think of it like entering a club. At the door (identification step), you tell the bouncer your name. Then (verification step), they check your ID to make sure it matches your name and you're on the guest list.*

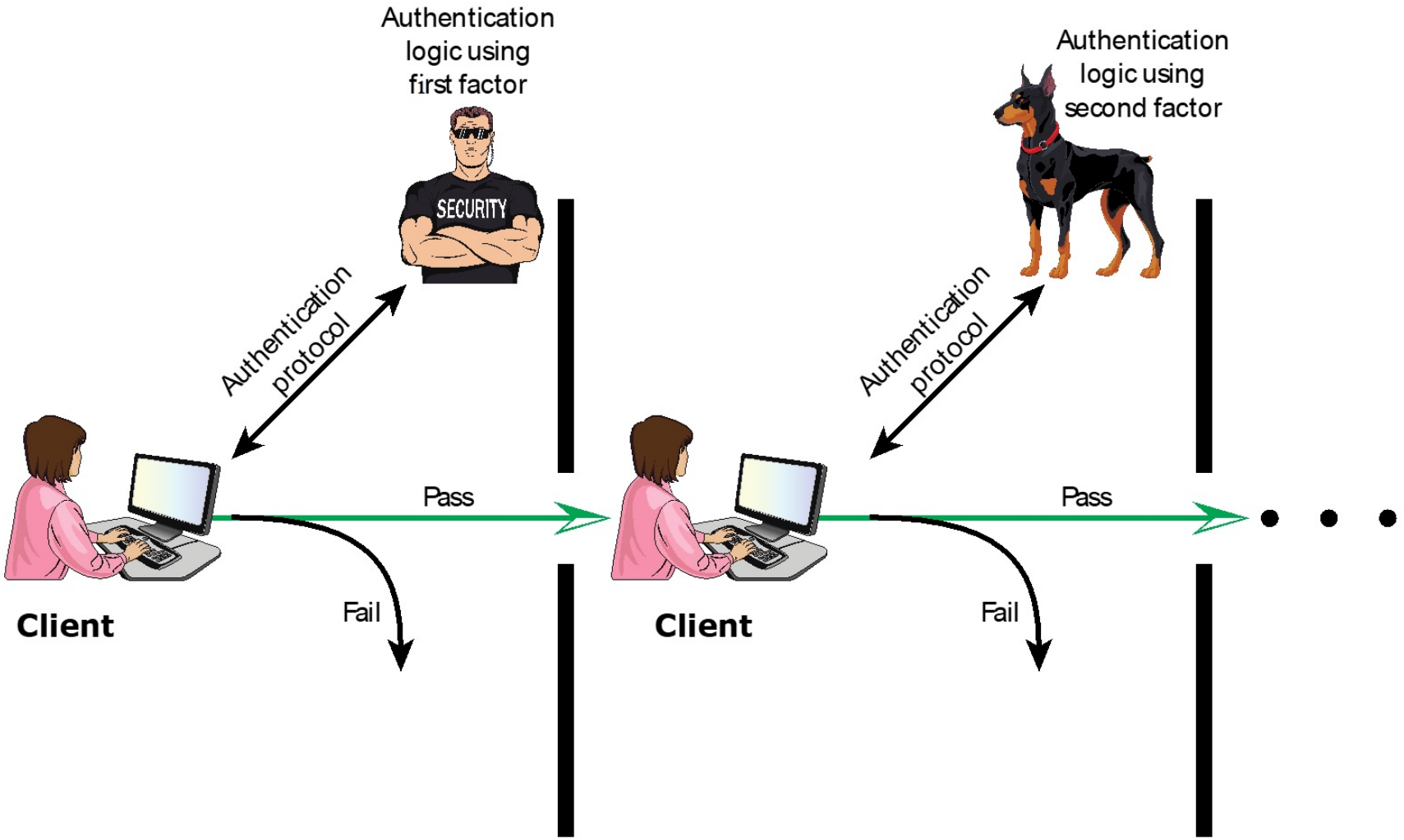
Means of User Authentication

- There are four general means of authenticating a user's identity, which can be used alone or in combination:
  - Something the individual knows (Knowledge-based)
    e.g., password, PIN, or answers to a prearranged set of questions.
  - Something the individual possesses (Possession-based)
    e.g., cryptographic keys, electronic keycards, smart cards, and physical keys
    - This type of authenticator is referred as a token
  - Something the individual is (static biometrics)
    e.g., fingerprint, retina, and face
  - Something the individual does (dynamic biometrics)
    e.g., voice pattern, handwriting characteristics, and typing rhythm
- Can use alone or combined
- Can provide user authentication
- all have issues

Multifactor authentication

- Refer to the use of more than one of the authentication means in order to authenticate a user
- The strength of authentication systems is determined by the number of factors incorporated by the system
- Two-factor authentication is an approach which requires to present two different factors for authentication
- For example:
  - A password and a USB token
  - A fingerprint and a smart card
  - A credit card and a signature
- An increasingly popular approach.

Multifactor authentication



Password-Based Authentication

Something You Know: Passwords

- Widely used line of defense against intruders
  - User provides name/login/identifier (ID) and password
  - System initially stores username and password
  - System compares password with the one stored for that specified login
- The password serves to authenticate the ID of the individual logging on to the system.
- In turn, the user ID provides security in the following ways:
  - Determines that the user is authorized to access the system
  - Determines the user's privileges, e.g., *normal* or *superuser*
  - Used in discretionary access control. e.g., by listing the IDs of the other users, a user may grant permission to them to read files owned by that user.
- Lots of things act as passwords!
  - For example: *Bad*
    - PIN
    - Social security number
    - Mother's maiden name
    - Date of birth
    - Name of your pet, etc.

Why Passwords?

- Why is “something you know” more popular than “something you have” and “something you are”?
- i.e., why are passwords so popular?
  - The initial choice for authentication.
  - Cost: They are free.
  - Convenience: They are convenient (use & management)
- It is worthwhile to study the use of passwords for user authentication in some details.

Vulnerability of Passwords

- Offline dictionary attack
  - The attacker obtains the system password file and compares the password hashes against hashes of commonly used passwords. If a match is found, the attacker can gain access by that ID/password combination.
  - **Countermeasures**: control access to database; reissue passwords if compromised; strong hashes and salts
- Specific account attack
  - The attacker targets a specific account and submits password guesses until the correct password is discovered.
  - **Countermeasure**: lock account after too many failed attempts
- Popular password attack
  - The attacker tries popular passwords with a wide range of user IDs.
  - **Countermeasures**: control password selection; block computers that make multiple attempts.
- Password guessing against single user
  - The attacker attempts to gain knowledge about the account holder and system password policies and uses that to guess password.
  - **Countermeasures**: control password selection; train users in password selection.

Vulnerability of Passwords

- Computer/Workstation hijacking
  - The attacker waits until a logged-in workstation is unattended.
  - **Countermeasure**: auto-logout.
- Exploiting user mistakes
  - Users write down password, share with friends, tricked into revealing passwords, use pre-configured passwords.
  - **Countermeasures**: user training, passwords plus other authentication.
- Exploiting multiple password use
  - Passwords re-used across different systems/accounts, make easier for attacker to access resources once one password discovered.
  - **Countermeasure**: control selection of passwords on multiple account/devices.
- Electronic monitoring
  - If a password is communicated across a network to log on to a remote system, it is vulnerable to eavesdropping.
  - Attacker intercepts passwords sent across network
  - **Countermeasure**: encrypt communications that send passwords

Password Retry

- Usernames are either public or easily guessable. It makes a brute force attack possible.
- **Common response**: Account lockout mechanism lock account after `X` tries for `Y` minutes
  - What `X` and `Y` should be?
  - Again, security vs. convenience.

Forgotten passwords

- What happens when you forget your password?
- Users forget passwords and need a way to recover them.
  - *User may be prompted to reset it using your email address, phone number, or security questions.*
  - Email reset
    - Risks exemplified by Mat Honan's story
  - Personal questions
    - Also called "security questions," "personal knowledge questions," or "life questions"
    - Another something-you-know factor
  - Sometimes a combination of these options
- Security questions
  - Might be an easier way for an attacker to obtain access to an account than guessing a password.
  - What is your favorite color? Sport? Team?
- Read: Personal Information Leakage During Password Recovery of Internet Services

Sarah Palin's password recovery[4]

- On 16 Sept. 2008, the Gov. Palin's Yahoo! account was hacked. How?
  - Password reset. *Hacker simply reset password using her birthdate, ZIP code and information about where she met her spouse*
    - Zip code?
    - *Only 2 in Wasilla*
    - Date of birth?
    - *Wikipedia: February 11, 1964*
    - Where did you meet your spouse?
    - *Wikipedia: met Todd in high school...*
- Password posted to /b/ on 4chan.
- When everyone tried to log into Palin's account, Yahoo! finally detected attack
- 22-year-old hacker David Kernell (a.k.a. Rubico) caught
  - Sentenced to one year of federal prison
- *The simplicity of the attack, of course, makes it no less illegal.*

[4] Palin e-mail hacker says it was easy

Problems with security questions

- Answers easy to guess or find out
  - Attackable using public records, LinkedIn, etc.!
- Answers hard to remember
  - People lie to increase security...then forget their answers.

Password-Based Authentication

- Password
  - What is a good password?
  - How to store the passwords?
  - How to submit the passwords?
  - How to respond (if no match)?

Storing passwords

- Upon initial usage, *ID*, *password* (or information based on it), and optionally other user information stored on system, e.g., in file or database
- To access system, user submits *ID* and *password*, compared against stored values.
- *How should passwords be stored?*
  - Plaintext: Once attacker gets access, she gets all passwords.
  - Encrypted: Where is the key stored?
  - Better solution: Use a hash function!

Storing Passwords in the Clear

`ID, P`

- Insider attack: normal user reads the database and learns other users passwords.
  - **Countermeasure**: access control on password database.
- Insider attack: admin user reads the database and learns other users passwords.
  - **Countermeasure**: none - admin users must be trusted!

- Outsider attack: attacker gains unauthorized access to database and learns all passwords.
  - **Countermeasure**: do not store passwords in the clear.

Encrypting the Passwords

$$\texttt{ID, } E_{\texttt{k}}(\texttt{P})$$

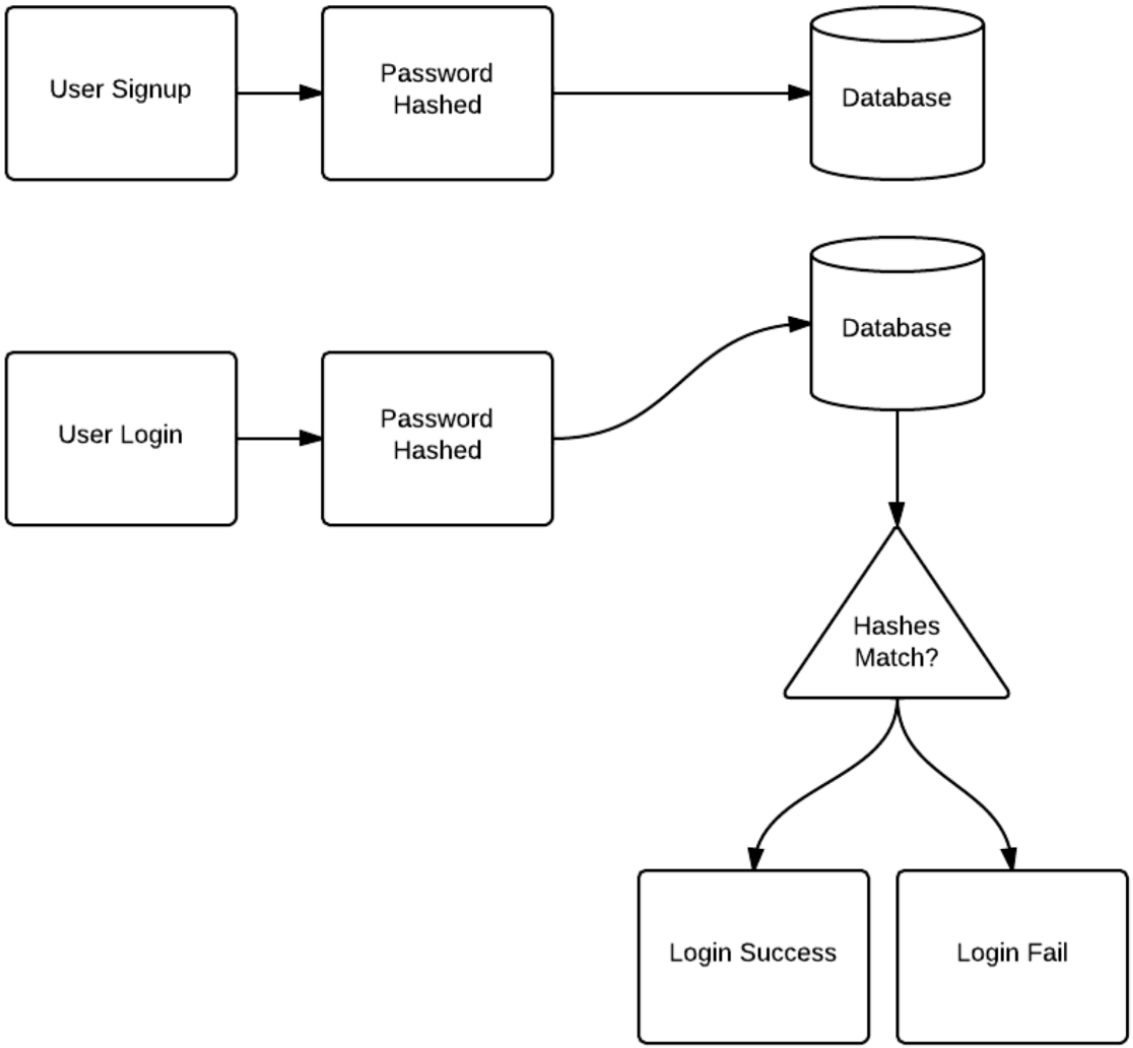
- Encrypted passwords are stored.
- When user submits ***password***, it is encrypted and compared to the stored value.
- Drawback
  - Secret key, `k`, must be stored (on file or memory);
  - If attacker can read database, then likely they can also read `k`.

Hashing the Passwords

`ID, H(P)`

- Hashes of passwords are stored.
- When user submits *`password`*, it is hashed and compared to the stored value.
- Practical properties of hash functions
  - Variable sized input; produce a fixed length, small output
  - No collisions
  - One-way function
- If attacker gains database, practically impossible to take a hash value and directly determine the original password.

Hashing the Passwords



Dictionary

- *In the context of password security, a "dictionary" is a list of common words, phrases, and character combinations that attackers can use to guess passwords through a brute force attack.*
- A dictionary can be created manually, by compiling a list of common words and phrases, or it can be generated automatically using software tools that scrape web pages, forums, and other online sources for commonly used passwords
- These lists are often referred to as "password dictionaries" or "wordlists".
- Some dictionaries can contain millions of possible password combinations, making them a powerful tool for attackers.
- Attackers can use these dictionaries to quickly generate a large set of possible passwords

Brute Force Attack on Hashed Passwords

- **Aim**
  - *given one (or more) target hash value, find the original password.*

- Start with large set of possible passwords (e.g., from dictionary, all possible n-character combinations)
- Calculate hash of possible password, compare with target hash.
  - if match, original password is found.
  - else, try next possible password.
- Attack duration depends on size of possible password set.

Pre-calculated Hashes and Rainbow Tables

- How to speed up brute force attack? Use hash values calculated by someone else.
- Possible passwords and corresponding hashes stored in database.
- Attacker performs lookup on database for target hash.
- How big is such a database of pre-calculated hashes?
  - In raw form, generally too big to be practical (`100's, 1000's of TB`).
  - Using specialized data structures (e.g., Rainbow tables), can obtain manageable size, e.g., `1TB`.
  - *Rainbow tables are a specific type of precomputed hash table that make lookups more efficient*
- **Trade-off**: reduce search time, but increase storage space.
- **Countermeasures**
  - Longer passwords.
  - Slower hash algorithms.
  - Salting the password before hashing.
    - *Salting involves adding random data to a password before hashing it, which makes it more difficult for an attacker to use pre-calculated hashes to guess the original password.*

Salting Passwords

- *Salting is a technique used to make it more difficult to crack passwords by adding random data to the password before it is hashed*
- Salts are random, unique strings that are either appended or prepended to the user's password before hashing.

`ID, Salt, H(P||Salt)`

- When *ID* and *password* initially created, generate random *s-bit* value (*salt*), concatenate with *password* and then hash.
- When user submits *password*, *salt* from password database is concatenated, hashed and compared.
- If attacker gains database, they know the *salt*; same effort to find password as brute force attack.
- BUT, pre-calculated values (e.g., Rainbow tables) are no longer feasible.
  - *The salt is unique for each user, which makes it difficult for attackers to use precomputed hash values or rainbow tables.*
  - Space required increased by factor of 2^s.

Why a salt value?

- Make a hash function look non-deterministic
- Prevent duplicate passwords from being visible in the password file
- Increase the difficulty of offline dictionary attacks
- Nearly impossible to tell if a person used the same password on multiple systems

Why called Salt?

- *The term "salt" is used in this context because just like adding salt to food, adding random data (i.e. salt) to a password makes it more complex and harder to crack. In other words, it adds a level of "flavor" or randomness to the password that makes it more secure. The term "salt" is also used because it is a short, simple and easy-to-remember word that conveys the concept of adding something extra to enhance the security of the password.*

Password Storage: Best Practice

- When storing user login information, always store a hash of a salted password

  `ID, Salt, H(P||Salt)`

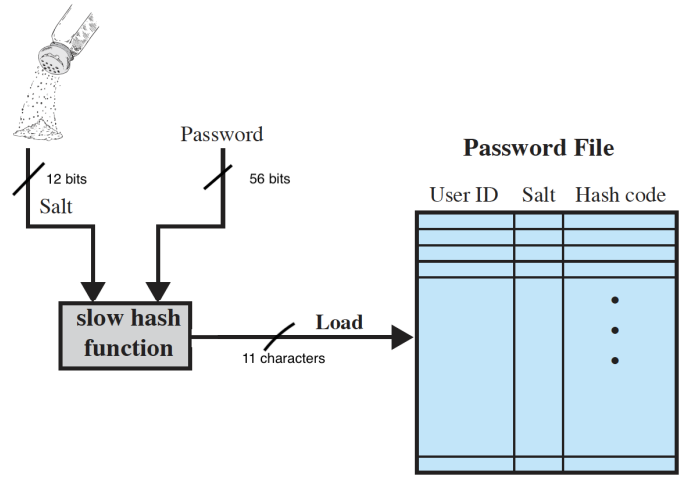
  - *`Password`*: see password policies.
  - *`Salt`*: random, generated when ID/password first stored; *`64 bits`*, *`128 bits`*, or longer.
  - *`Password Hash Functions`*: slow, adaptive speed (work factor), e.g. *`bcrypt/scrypt, PBKDF2, Argon2`*
- Design for failure: assume password database will eventually be compromised.

Secure Password Hash Functions

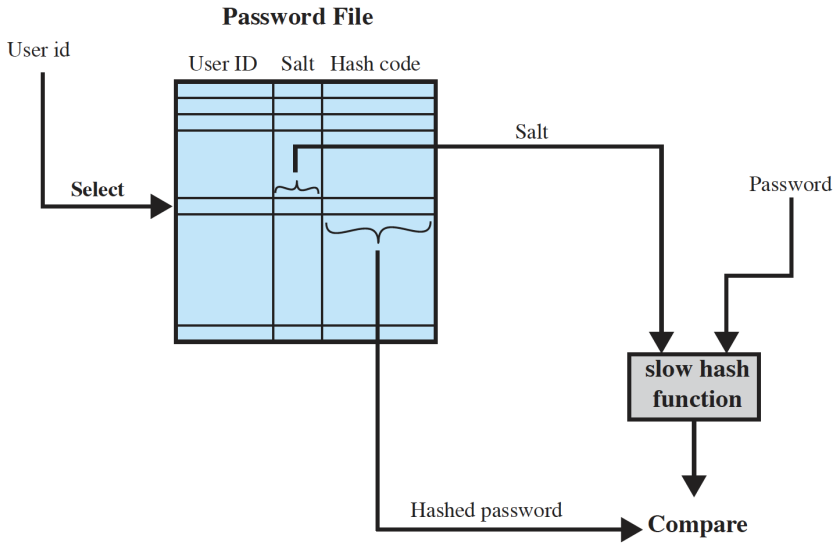
- Password hash functions are designed to be slow and computationally intensive.
- It's crucial to use secure and slow password hash functions with adaptive speed (work/cost factor) to make it computationally intensive for attackers to brute-force passwords.
- These functions allow for a configurable work factor, which determines how much computational effort is required to generate the hash.
- Some commonly recommended password hash functions
  - **bcrypt**: bcrypt is a widely used password hashing algorithm that incorporates a work factor, making it slow and adaptive.
  - **scrypt**: scrypt is designed to be memory-intensive in addition to CPU-intensive.
  - **PBKDF2 (Password-Based Key Derivation Function 2)**: is a key derivation function that can be used for securely hashing passwords.
  - **Argon2**: Argon2 is the most recent and recommended password hashing algorithm.

- *The use of slow hash functions and salting can significantly increase the time and resources required to crack passwords, making them more secure.*

Use of Hashed Passwords: UNIX Password Scheme



(a) Loading a new password



(b) Verifying a password

Password Cracking

- Password cracking is technique used by attackers to gain unauthorized access to restricted systems by guessing passwords.
  - i.e., it's an art of obtaining the correct password that gives access to a system protected by an authentication method.
- Employ a number of techniques to achieve its goals.
- The cracking process can involve either comparing stored passwords against word list or use algorithms to generate passwords that match
- *Password crackers exploit the fact that many people choose weak passwords that are easy to guess*
  - Weak passwords are often short, easy to remember, or based on personal information.
  - Attackers can also use tools to guess passwords based on common patterns, such as using the user's name or birthday.

Password Cracking

- Brute force attack
  - Attacker tries every possible password combination against the hashed password until they find the correct one.
  - This is a time-consuming process, but it can be effective for cracking weak passwords.
- Dictionary attacks
  - Attacker uses a list of common passwords to try against the target system.
  - Each password must be hashed using each salt value and then compared to stored hash values
  - This is a more efficient method than a brute-force attack, as it only tries passwords that are likely to be used.

Password Cracking Techniques

- Rainbow table attacks
  - How to speed up brute force attack? Use hash values calculated by someone else
  - Attacker uses precomputed tables of hash values for all salts to quickly find the hash of a given password.
  - This is a very efficient method of cracking passwords, but it requires a large amount of storage space.
  - Can be countered by using a sufficiently large salt value and a sufficiently large hash length
- Social engineering attacks
  - The attacker tricks the user into revealing their password.
  - This can be done through phishing emails, fake websites, or other means of deception.

Modern Approaches

- Complex password policy concept:
  - Users are doing a better job of selecting passwords, and organizations are doing a better job of forcing users to pick stronger passwords
- However password-cracking techniques have also improved
  - The processing capacity available for password cracking has increased dramatically
  - The use of sophisticated algorithms to generate potential passwords
  - Studying examples and structures of actual passwords in use
    - numerous sets of leaked password files have become available for analysis.

Password Cracking Tools

- (Some) popular password cracking (aka password auditing) tools[5]
  - Cain & Abel: Password recovery tool for Windows systems.
    - Used to recover passwords from a variety of sources, including cached passwords, logon history, and network traffic.
    - Includes a number of other features, such as a packet sniffer, a keystroke logger, and a password cracker.
  - John the Ripper: password cracking tool for Unix-like systems
    - known for its speed and its ability to crack a wide variety of password hashes.
    - Used to crack passwords in a distributed manner, using multiple computers to crack the same password hash.
- Admins should use these tools to test for weak passwords since attackers will.

  It's illegal to use a password cracking tool for hacking into another person's account or data.
- ... and they're getting faster
  - Custom GPU-based hardware
    - A 5-server rig with 25 Radeon GPUs
    - Can crack 77 million md5crypt-hashed passwords per second
  - Cloud-based cracking tools

[5] 11 Password Cracker Tools (Password Hacking Software 2022))

Password File Access Control

- One way to thwart a password attack is to deny the opponent access to the password file
- Can block offline guessing attacks by denying access to encrypted passwords
  - Make available only to privileged users
  - Often, the hashed passwords are kept in a separate file from the user IDs, referred to as a shadow password file
- Still have Vulnerabilities
  - Weakness in the OS that allows access to the file
  - Accident with permissions making it readable
  - Users with same password on other systems
  - Access from unprotected backup media
  - Sniff passwords in network traffic

Selecting Passwords

- Q: How do people pick their passwords?

Often they don't!

- In April 1994, English teenager ("Datastream Cowboy") penetrated Pentagon computers via the Air Force Rome (New York) Laboratory, started probing Korean nuclear facilities.
  - How did he do it?
    - Guessed default guest password!
- Surveys show that half of users leave the default password in place for their routers at home.
  - E.g., Warkitting: the Drive-by Subversion of Wireless Home Routers

- *Unfortunately, many users still fail to change default passwords, leaving their systems vulnerable to unauthorized access.*
- *Warwalkers drive around neighborhoods with specialized equipment to scan for wireless networks that use default passwords. Once they identify such networks, they can easily gain access and exploit the vulnerabilities of the connected devices.*

Often they don't!

- What's the most important thing in the world to prevent unauthorized access to?
  - Nuclear missiles!
    - From 1962 to 1977, the passcode for launching Minuteman missiles was 00000000[6]
      - Source of the reports was former ICBM launch officer Bruce Blair
    - Strategic Air Command was more afraid of lost passwords than of Armageddon underscores the importance of password security!

- *It's concerning to learn that sensitive systems like nuclear missile launch codes were once protected by such weak passwords.*

[6] 00000000

Good and Bad Passwords

- **Bad passwords**
  - frank
  - Fido
  - Password
  - incorrect
  - Pikachu
  - 102560
- **Good passwords**
  - jfIej,43j-EmmL+y
  - 09864376537263
  - P0kem0N
  - FSa7Yago
  - 0nceuP0nAt1m8
  - PokeGCTall150
- Q: How would you define a good password?
- SplashData releases an annual list of the worst passwords of the year based on data breaches.
- The company analyzes millions of passwords that have been leaked online and identifies the most common and easily guessable passwords.
- This information is then used to raise awareness about the importance of using strong, unique passwords.
- Top 200 most common passwords of the year 2022
  - The list details how many times a password has been exposed, used, and how much time it would take to crack it. It also compare the most common passwords of 2019-2021, highlighting how their positions have changed.

Most popular passwords 2018[7]

1. 123456 (Unchanged)
2. password (Unchanged)
3. 123456789 (Up 3)
4. 12345678 (Down 1)
5. 12345 (Unchanged)
6. 111111 (New)
7. 1234567 (Up 1)
8. sunshine (New)
9. qwerty (Down 5)
10. iloveyou (Unchanged)
11. princess (New)
12. admin (Down 1)
13. welcome (Down 1)
14. 666666 (New)
15. abc123 (Unchanged)
16. football (Down 7)
17. 123123 (Unchanged)
18. monkey (Down 5)
19. 654321 (New)
20. !@#$%^&* (New)
21. charlie (New)
22. aa123456 (New)
23. donald (New)
24. password1 (New)
25. qwerty123 (New)

[7] The 25 Most Popular Passwords of 2018 Will Make You Feel Like a Security Genius

Most popular passwords 2019[8]

1. 123456 (Unchanged)
2. 123456789 (Up 1)
3. qwerty (Up 6)
4. password (Down 2)
5. 1234567 (Up 2)
6. 12345678 (Down 2)
7. 12345 (Down 2)
8. iloveyou (Up 2)
9. 111111 (Down 3)
10. 123123 (Up 7)
11. abc123 (Up 4)
12. qwerty123 (Down 3)
13. 1q2w3e4r (New)
14. admin (Down 2)
15. qwertyuiop (New)
16. 654321 (Up 3)
17. 555555 (New)
18. lovely (New)
19. 7777777 (New)
20. welcome (Down 7)
21. 888888 (New)
22. princess (Down 11)
23. dragon (New)
24. password1 (Unchanged)
25. 123qwe (New)

[8] The Top 100 Worst Passwords of 2019

Other Common Characteristics of Passwords

- Most use only alphanumeric characters
- Most are in (password) dictionaries
- Many users re-use passwords across systems
- Some very common passwords: `123456`, `password`,`12345678`, `qwerty`, `abc123`, `...`
- When forced to change passwords, most users change a single character.

Top 200 most common passwords of the year 2019-2021)

	2019		2020		2021	
	Password	Number of users	Password	Number of users	Password	Number of users
1	12345	2,812,220	123456	2,543,285	123456	103,170,552
2	123456	2,485,216	123456789	961,435	123456789	46,027,530
3	123456789	1,052,268	picture1	371,612	12345	32,955,431
4	test1	993,756	password	360,467	qwerty	22,317,280
5	password	830,846	12345678	322,187	password	20,958,297
6	12345678	512,560	111111	230,507	12345678	14,745,771
7	zinch	483,443	123123	189,327	111111	13,354,149
8	g_czechout	372,278	12345	188,268	123123	10,244,398
9	asdf	359,520	1234567890	171,724	1234567890	9,646,621
10	qwerty	348,762	senha	167,728	1234567	9,396,813
11	1234567890	329,341	1234567	165,909	qwerty123	8,933,334
12	1234567	261,610	qwerty	156,765	000000	8,377,094
13	Aa123456.	212,903	abc123	151,804	1q2w3e	8,204,700
14	iloveyou	171,657	Million2	143,664	aa12345678	8,098,805
15	1234	169,683	000000	122,982	abc123	7,184,645
16	abc123	150,977	1234	112,297	password1	5,771,586
17	111111	148,079	iloveyou	106,327	1234	5,544,971
18	123123	145,365	aaron431	90,256	qwertyuiop	5,197,596
19	dubsmash	144,104	password1	87,556	123321	5,168,171
20	test	139,624	qqww1122	85,476	password123	4,681,010
21	princess	122,658	123	84,438	1q2w3e4r5t	4,624,323
22	qwertyuiop	116,273	omgpop	77,492	iloveyou	4,387,925
23	sunshine	107,202	123321	73,506	654321	4,384,762
24	BvtTest123	106,991	654321	69,148	666666	4,329,996
25	11111	104,395	qwertyuiop	64,632	987654321	4,239,959

Getting passwords right is difficult

- You need to balance security and convenience.
  - Weak passwords easy to remember.
  - Strong passwords difficult to remember
- Password Do's and Don'ts
  - Choose long passwords with special characters.
  - Use passphrases to remember passwords:
    "It was a dark and stormy night" $\rightarrow$ `iWadasn`
  - Don't use dictionary words or personal information.
  - Don't reuse passwords
- ... but user compliance is extremely difficult.

Using Better Passwords

- Clearly have problems with passwords
- Goal is to eliminate guessable passwords while allowing the user to select a password that is memorable
- Techniques:
  - User education
  - Computer generated passwords
  - Reactive password checking
  - Complex password policy/proactive password checking

Password Selection Strategies

- User education
  - Ensure users are aware of importance of hard-to-guess passwords; advise users on strategies for selecting passwords.
- Computer generated passwords
  - Generate random or pronounceable passwords (but poorly accepted by users).
    - Users have trouble remembering them
- Reactive password checking
  - Regularly check user's passwords, inform them if weak passwords
- Complex password policy/proactive password checking
  - Advise user on strength when selecting a password

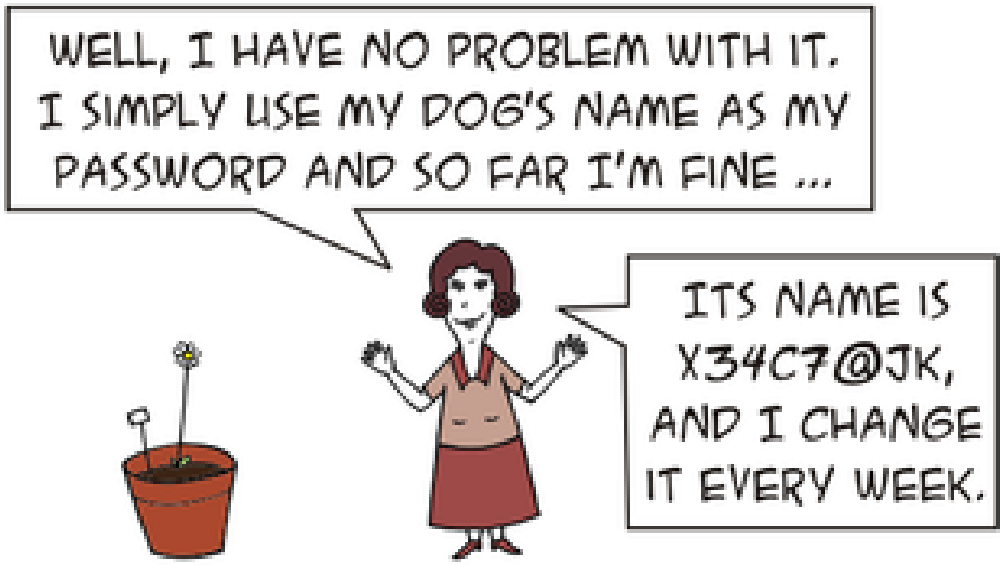
Possible Approaches to Proactive Password Checking

- Rule enforcement
  - Specific rules that passwords must adhere to
  - Can be automated by using a proactive password checker, such as the password/passphrase strength checking and policy enforcement toolset
- Password checker
  - Compile a large dictionary of possible "bad/not to use" passwords
  - When a user selects a password, the system checks to make sure that it is not on the disapproved list
  - Time and space issues. Need to have a fairly fast test of the "goodness" of a password
- Markov Model
  - Generates guessable passwords
  - Hence reject any password it might generate
- Bloom filter
  - Used to build a table based on hash values
  - Check desired password against this table

Password Defense: 1- Password expiration (password changes)

- Common interval: 90 days
- May help sometimes, but
  - **Increased help desk costs**
    - Frequent password changes can lead to an increase in help desk calls as users forget their passwords.
    - This can be a significant cost for organizations, as each password reset can cost up to $25.
  - **Reduced security**
    - Password-reset questions and other out-of-band authentication methods are often vulnerable to social engineering attacks.
    - Attackers can use social engineering techniques to trick users into revealing their passwords or other sensitive information.
  - **User inconvenience**
    - Frequent password changes can be inconvenient for users, especially if they have to remember multiple passwords for different accounts.
    - This can lead to password reuse, which is a major security risk.

Password expiration (password changes)

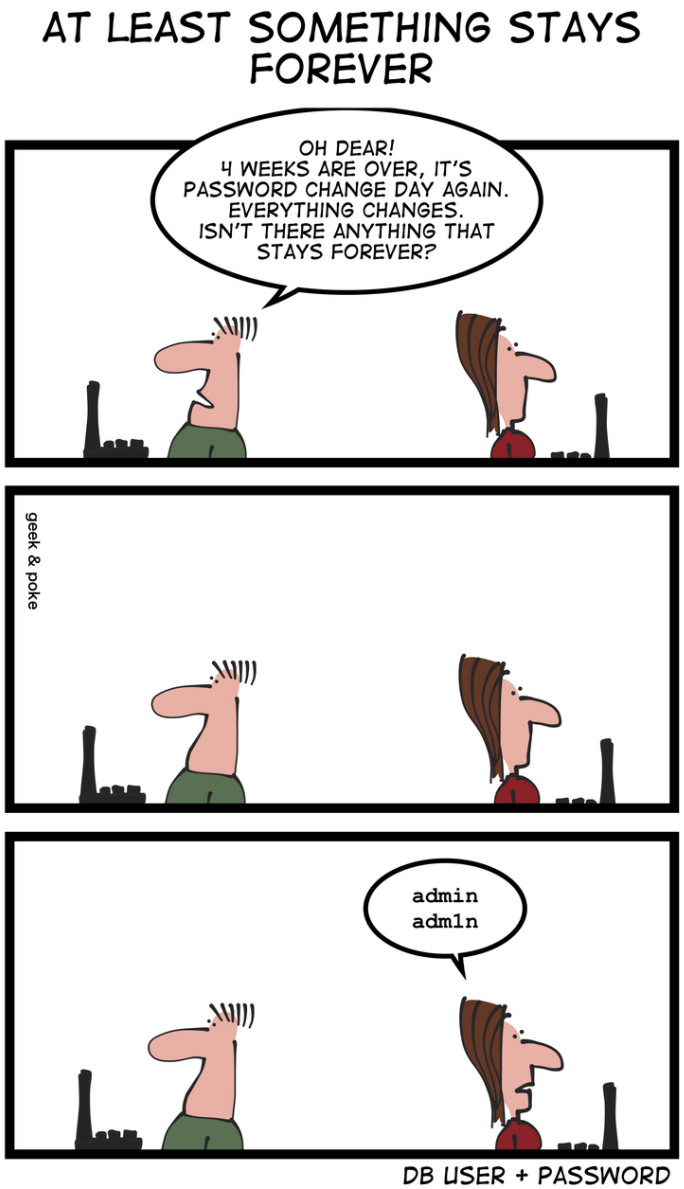


1- Password expiration (password changes)

- How do users change their passwords[9]?
  - `Password1`
  - `Password2`
  - `Password3`
  - `Pa$sword1`

[9] The security of modern password expiration: an algorithmic framework and empirical analysis

Password expiration (password changes)



Password expiration (password changes)

"Yesterday I changed everyone's password to 'password'. I sent it to everyone in a memo, put it on a big sign on the wall and printed it on all of the coffee cups. Guess how many people called me this morning because they forgot the password."

2- Password managers

- Why should users have to remember passwords?
- Password managers solve this problem.
- Password managers are a valuable tool for managing multiple passwords securely.
- They can help you create strong, unique passwords for all of your accounts, and they can store those passwords in an encrypted format so that they are not easily accessible to hackers.
  - LastPass, 1Password, Bitwarden, Dashlane, Keeper, etc.
- Idea: Encrypt all of your passwords under a single, master password
- One password to rule them all

- Best Password Managers of 2023

Other Password Issues

- Too many passwords to remember.
  - Results in password reuse.
  - Q: Why is this a problem?
- Other issues:
  - Failure to change default passwords.
  - Social engineering & phishing.
  - Error logs may contain passwords.
  - Bugs, keystroke logging, spyware, etc.
  - Users might intentionally share passwords!

Other Password Issues: ... and some exotic ones

- E.g., reflections
  - iSpy: Automatic Reconstruction of Typed Input from Compromising Reflections

Figure 1: Some example threat scenarios that we investigated. Video was recorded in both indoor and outdoor environments, using various consumer video cameras. top: shoulder surfing, bottom: reflection surfing, bottom right: key *pop-out* event.

Other Password Issues: ... and some exotic ones

- Vibrations
  - (sp)iPhone: Decoding Vibrations From Nearby Keyboards Using Mobile Phone Accelerometers

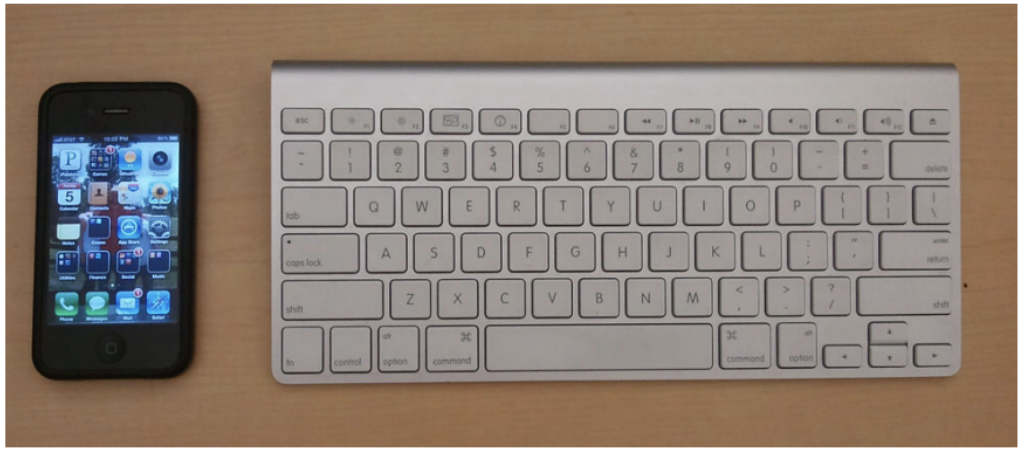


Figure 1: Our experimental placement of a mobile phone running a malicious application attempting to recover text entered using the nearby keyboard.

Passwords leaks

- 1.4 Billion Clear Text Credentials Discovered in a Single Database
- 5 Million Google Passwords Leaked Source
- 10000 Twitter User Accounts Exposed
- BREACH LEVEL INDEX
- Yahoo data breach 2017: All 3 Billion Yahoo Accounts Were Affected by 2013 Attack
- First American Financial Corp. data breach: 885 Million Sensitive Financial Records Exposed Online
- Marriott/Starwood data breach: 500 million
- RockYou2021: largest password compilation of all time leaked online with 8.4 billion entries
- Passwords Leaked in Data Breach 2021: Study Reveals Shocking Superhero Passcodes Used!

Passwords

- The bottom line... Password attacks are too easy.
  - Often, one weak password will break security.
  - Users choose bad passwords
  - Social engineering attacks, etc.
- Oscar has (almost) all of the advantages.
- Passwords are a BIG security problem.
  - And will continue to be a problem.
  - Passwords are not going to disappear anytime soon.
- Watch: What is Your Password? 2015
- Watch: What's Your Password? 2017
- Watch: I know your password! It's...
- read: How to choose a secure password

- Something You Have: Token-based Authentication

Something You Have: Token-based Authentication

- Something in your possession. You need to prove the possession of the factor in order to authenticate.
- Objects that a user possesses for the purpose of user authentication are called tokens
  - Car key
  - Laptop computer (or MAC address)
  - Cell phone
  - (One-time) Password generator
  - ATM card, smart card, etc.

Security Tokens[10]

- A security token is a physical device used to gain access to an electronically restricted resource.
- The token is used in addition to or in place of a password

[10] Source: Wikipedia: Security Tokens

Types of Cards Used as Tokens[11]

Card Type	**Defining Feature**	**Example**
Embossed	Raised characters only, on front	Old credit card
Magnetic stripe	Magnetic bar on back, characters on front	Bank card
Memory	Electronic memory inside	Phone card
Smart - Contact - Contactless	Electronic memory & processor inside - Electrical contacts on surface - Radio antenna embedded inside	Biometric ID card

[11] Credit: Table 3.2 in Stallings and Brown, Computer Security, 3rd Ed., Pearson 2015

Memory Cards

- Can store but do not process data
- The most common is the magnetic stripe card
  - Magnetic stripe can store only a simple security code, which can be read (and unfortunately reprogrammed) by an inexpensive card reader
- Can include an internal electronic memory
- Can be used alone for physical access
  - Hotel room
  - ATM
- Provides significantly greater security when combined with a password or personal identification number (PIN).
- Drawbacks of memory cards include:
  - Requires a special reader
  - Loss of token
  - User dissatisfaction

Smart Tokens

- A wide variety of devices qualify as smart tokens.
- Can be categorized along three dimensions that are not mutually exclusive
  - Physical characteristics
    - Smart tokens include an embedded microprocessor
    - A smart token that looks like a bank card
    - Other smart tokens can look like calculators, keys, or other small portable objects
  - Interface
    - Manual interfaces include a keypad and display for human/token interaction.
    - Smart tokens with an electronic interface communicate with a compatible reader/writer.
    - Contact and contactless interfaces
  - Authentication protocol

Smart Tokens

- Authentication protocol: Classified into three categories:
  - **Static**: user authenticates himself to the token and then the token authenticates the user to the computer
  - **Dynamic password generator**:
    - Token generates a unique password periodically (e.g., every minute).
    - The password is entered into the computer system for authentication, either by the user or via the token.
    - The token and computer system must be initialized and kept synchronized so that the computer knows the password that is current for this token.
  - **Challenge-response**
    - The computer system generates a challenge, such as a random string of numbers.
    - The smart token generates a response based on the challenge.
    - For example, public-key cryptography could be used and the token could encrypt the challenge string with the token's private key.

Smart Cards

- Most important category of smart token
  - Has the appearance of a credit card
  - Has an electronic interface to communicate with a compatible reader/writer (contacts/contact-less)
  - May use any of the smart token protocols to authenticate with reader/computer
- Contain
  - An entire microprocessor (including processor, memory, and I/O ports).
  - Some versions incorporate a special co-processing circuit for cryptographic operation to speed the task of encoding/decoding messages or generating digital signatures to validate the information transferred.
- Typically include three types of memory
  - Read-only memory (ROM)
    - Stores data that does not change during the card's life
  - Electrically erasable programmable ROM (EEPROM)
    - Holds application data and programs, e.g., the protocols that the card can execute. It also holds data that may vary with time.
  - Random access memory (RAM)
    - Holds temporary data generated when applications are executed

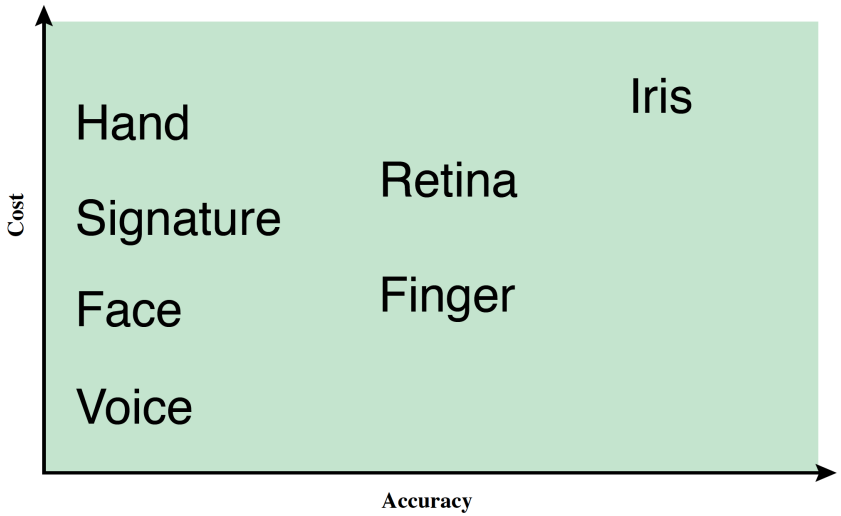
Two-Factor Authentication in practice

- Two-Factor == Two-Step
- Two most popular options: email and cell phone (apps or text msgs)
- Pretty much all major websites offer two-factor authentication.
  - LinkedIn, Twitter, Microsoft, Apple, Google, Dropbox, Tumblr, Snapchat, Instgram, etc.

Something you are: Biometric Authentication

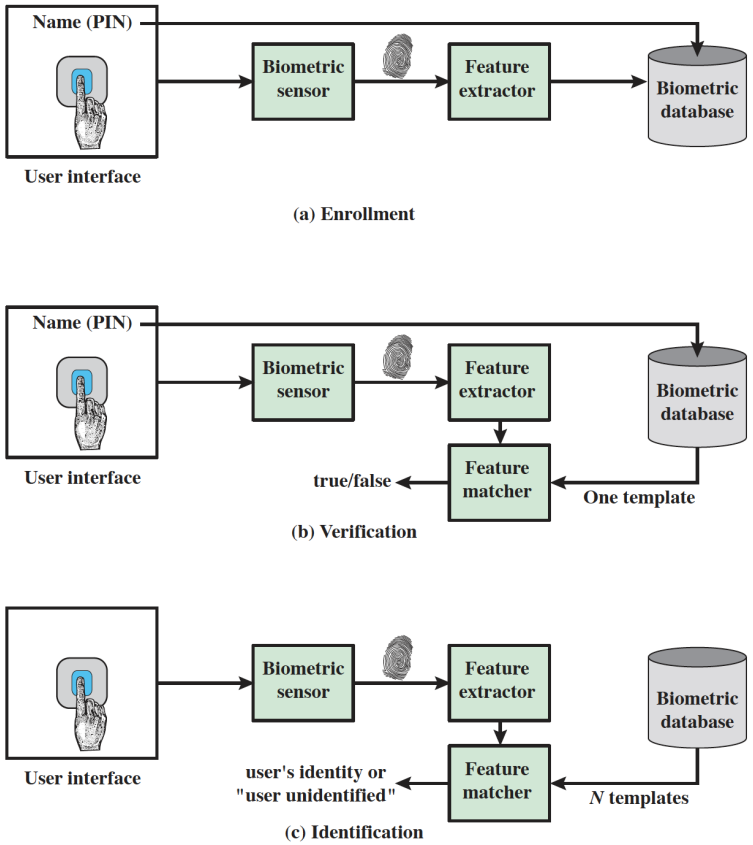
- Attempts to authenticate an individual based on unique physical characteristics
- Based on pattern recognition
- Is technically complex and expensive when compared to passwords and tokens
- Physical characteristics used include:
  - Facial characteristics
  - Fingerprints
  - Hand geometry
  - Retinal pattern
  - Iris
  - Signature
  - Voice

Cost vs Accuracy for Biometric Authentication



- The concept of accuracy does not apply to user authentication schemes using smart cards or passwords.
  - For example, if a user enters a password, it either matches exactly the password expected for that user or not.
- In the case of biometric parameters, the system instead must determine how closely a presented biometric characteristic matches a stored characteristic.

Generic Biometric System



- Enrollment creates an association between a user and the user's biometric characteristics.
- Depending on the application, user authentication either involves verifying that a claimed user is the actual user or identifying an unknown user.

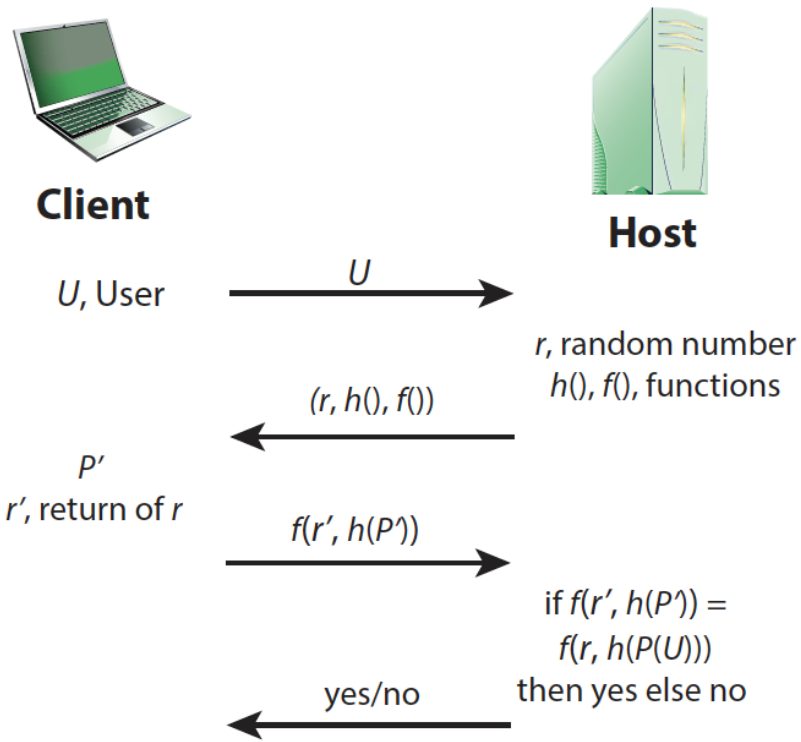
Remote User Authentication

- Simplest form of user authentication is local authentication, in which a user attempts to access a system that is locally present.
- Authentication over a network, the Internet, or a communications link is more complex
- Additional security threats such as:
  - Eavesdropping, capturing a password, replaying an authentication sequence that has been observed
- Generally rely on some form of a challenge-response protocol to counter threats

Remote User Authentication: Password protocol

- User transmits identity to remote host
- Host generates a random number r (**nonce**) and returns it to the user
- In addition, host specifies two functions $h()$ (hash function) and $f()$ to be used in the response
- This transmission from host to user is the challenge
- User's response: $f(r',h(P'))$, where $r'=r$ and P' is the user's password
- Host stores a hash function of each registered user's password
  - $h(P(U))$ for user U
- Host compares $f(r',h(P'))$ to the calculated $f(r,h(P(U)))$
  - If match, the user is authenticated.

Password protocol



Password protocol

- This scheme defends against several forms of attacks:
  - The host stores not the password but a hash code of the password.
    - secure the password from intruders into the host system.
  - Transmit a function in which the password hash is one of the arguments.
    - for a suitable function f, the password hash cannot be captured during transmission.
  - Use of a random number as one of the arguments of f defends against a replay attack
    - in which an adversary captures the user's transmission and attempts to log on to a system by retransmitting the user's messages.

Kerberos: The Network Authentication Protocol[12]

- There are a number of approaches that organizations can use to secure networked servers and hosts.
- Kerberos' name comes from Greek mythology, the three-headed guard dog of Hades.
  - *It's pretty fitting since it takes a third-party (a Key Distribution Center) to authenticate between a client and a service or host machine*
- In security, Kerberos is an authentication protocol based on symmetric key crypto
  - Originated at MIT
  - Has been issued as an Internet standard and is the de facto standard for remote authentication
  - Relies on a Trusted Third Party (TTP)
  - Uses tickets to authenticate
  - Avoids storing passwords locally or sending them over the internet
  - *Allows two users (or client and server) to authenticate each other over an insecure network*

[12] https://web.mit.edu/kerberos/

Motivation for Kerberos

- Authentication using public keys
  - `N` users $\rightarrow$ `N` key pairs
- Authentication using symmetric keys
  - `N` users requires (on the order of) $\texttt{N}^2$ key pairs
- Symmetric key case does not scale
- Kerberos based on symmetric keys but only requires `N` keys for `N` users

  \- Security depends on TTP

  \+ No PKI is needed

Kerberos

- Users wish to access services on servers:
  - Assume an open distributed environment in which users at workstations wish to access services on servers distributed throughout the network.
  - We would like for servers to be able to restrict access to authorized users and to be able to authenticate requests for service
- *Security risk: (impersonation) An opponent can pretend to be another client and obtain unauthorized privileges on server machines.*
- A workstation cannot be trusted to identify its users correctly to network services
  - User pretend to be another user.
    - A user may gain access to a particular workstation and pretend to be another user operating from that workstation
  - User alter the network address of a workstation.
    - A user may alter the network address of a workstation so that the requests sent from the altered workstation appear to come from the impersonated workstation
  - User eavesdrop on exchanges and use a replay attack.
    - A user may eavesdrop on exchanges and use a replay attack to gain entrance to a server or to disrupt operations

Kerberos

- To counter this threat, *servers must be able to confirm the identities of clients who request service.*
  - Each server can be required to undertake this task for each client/server interaction,
  - but in an open environment, this places a substantial burden on each server.
- Kerberos is a computer-network authentication protocol that works on the basis of tickets to allow nodes communicating over a non-secure network to prove their identity to one another in a secure manner.

Kerberos

- Kerberos represents an authentication service based on the symmetric key encryption and on a Key Distribution Center (KDC) which is a trusted third part.
- Provides a centralized authentication server to authenticate users to servers and servers to users.
- Relies on conventional encryption, making no use of public-key encryption
- Two versions: version 4 and 5
  - Version 4 makes use of DES for encryption
  - Version 5 uses any encryption techniques as the cipher text is tagged with an encryption identifier.

Kerberos Functionality

- Instead of client sending password to application server:
  - Request ticket from Key Distribution Center (KDC)
  - Ticket and encrypted request sent to application server

Kerberos: Basic idea

- Key Distribution Center (KDC) consists of two parts logically separated:
  - Authentication Server (AS)
    - AS knows the passwords of all users
    - AS is responsible for handling a login request from a user
  - Ticket Granting Server (TGS)
    - Setting up secure channels is handled by TGS
- When requesting access to a service or host, three interactions take place between user and:
  - the Authentication Server
  - the Ticket Granting Server
  - the Service or host machine that user wants access to
- Read Chapter 23

Kerberos Version 4

- Makes use of DES to provide the authentication service
  - Authentication Server (`AS`)
    - Knows the passwords of all users and stores these in a centralized database
    - Issues ticket-granting tickets (TGTs) that are good for admission to the ticket-granting service (TGS).
    - Before network clients can get tickets for services, they must obtain a `TGT` from the authentication service
  - Ticket-Granting Server (`TGS`)
    - Shares a unique secret key with each server. These keys have been distributed physically or in some other secure manner
    - Issues tickets to users who have been authenticated to `AS`.
    - The user first requests a ticket-granting ticket from the `AS`.
    - Each time the user requires access to a new service, the client applies to the `TGS` using the ticket to authenticate itself
    - The `TGS` then grants a service-granting ticket for the particular service.
    - The client saves each service-granting ticket and uses it to authenticate its user to a server each time a particular service is requested.
  - Ticket
    - A ticket is an unforgeable, non replayable, authenticated object.
    - It is an encrypted data structure naming a user and a service that the user is allowed to obtain.
    - It also contain a time value and some control information.
  - Client/Server Exchange
    - The client presents the ticket for admission to a service.

Kerberos: Version 4

- C: Client
- AS: Authentication Server
- V: Application Server
- ID_c: identifier of user on C
- ID_v: identifier of V
- P_c: password of user on C
- AD_c: network address of C
- K_c: secret encryption key derived from the user's password
- $K_{c,tgs}$: session key for C and TGS
- K_{tgs}: secret encryption key shared by AS and TGT
- $K_{c,v}$: session key for C and V
- K_v: secret encryption key shared by TGS and V
- TS: timestamp
- ||: concatenation

Authentication Dialogue: Intuitive Description

- The client authenticates itself to the Authentication Server and receives a ticket. (All tickets are time-stamped.)
- It then contacts the Ticket Granting Server, and using the ticket it demonstrates its identity and asks for a service.
- If the client is eligible for the service, then the Ticket Granting Server sends another ticket to the client.
- The client then contacts the Service Server, and using this ticket it proves that it has been approved to receive the service

Authentication Dialogue: Problem 1:

- The lifetime of the ticket-granting ticket in Kerberos is an important factor in balancing security and usability.
- The lifetime associated with the ticket-granting ticket creates a problem:
  - If very short (e.g., minutes), the user will be repeatedly asked for a password
  - If long (e.g., hours), then an opponent has a greater opportunity for replay
    - This would give the opponent unlimited access to the resources and files available to the legitimate user
- Similarly, if an opponent captures a service-granting ticket and uses it before it expires, the opponent has access to the corresponding service.
- *The threat is that an opponent will steal the ticket and use it before it expires*
- *Solutions to this problem include using renewable tickets, session keys, and timestamp/sequence number mechanisms to prevent replay attacks.*
- A network service (the `TGS` or an application service) must be able to prove that the person using a ticket is the same person to whom that ticket was issued

Authentication Dialogue: Problem 2:

- *The Kerberos protocol also includes mutual authentication between the user and the network service, ensuring that the user is communicating with the intended service and not an attacker who has intercepted the traffic.*
- Servers need to authenticate themselves to users
  - Without such authentication, an opponent could sabotage the configuration so that messages to a server were directed to another location.
  - The false server would then be in a position to act as a real server and capture any information from the user and deny the true service to the user.

Summary of Kerberos Version 4 Message Exchanges

- Authentication Service Exchange to obtain Ticket-Granting Ticket
- Authentication Service Exchange to obtain Service-Granting Ticket
- Client/Server Authentication Exchange to obtain Service

Authentication Service Exchange

- To obtain Ticket-Granting Ticket
  - Client sends a message to the AS requesting access to TGS.

1) C → AS: $ID_c || AD_c || ID_{tgs} || TS_1$

  - AS looks up the user in its database, gets the associated user's password.
  - The AS then responds with a TGT and a session key $K_{c,tgs}$, encrypted with a key derived from the user's password.

2) AS → C: $E_{K_c}[K_{c,tgs} || ID_{tgs} || TS_2 || Lifetime_2 || Ticket_{tgs}]$

$Ticket_{tgs} = E_{K_{tgs}}[K_{c,tgs} || ID_c || AD_c || ID_{tgs} || TS_2 || Lifetime_2]$,

  - Ticket contains user's ID, server's ID, a timestamp, a lifetime, and a copy of the same session key sent in outer message to client. The entire ticket is encrypted using a secret key shared by the AS and the TGS.

6. When the client receives the reply, it prompts the user for his/her password, generates the key, then attempt to decrypt the incoming message. If the password is correct, the ticket and session key are successfully recovered.
  - The session key has been securely delivered to both C and the TGS

Ticket-Granting Service Exchange

- To obtain Service-Granting Ticket
  - The client C sends the TGS a message that includes the ticket plus the ID of the requested service

3) $C \rightarrow TGS: ID_v || Ticket_{tgs} || Authenticator_c$

$Ticket_{tgs} = E_{K_{tgs}}[K_{c,tgs} || ID_c || AD_c || ID_{tgs} || TS_2 || Lifetime_2]$,

$Authenticator_c = E_{K_{c,tgs}}[ID_c || AD_c || TS_3]$

4) $TGS \rightarrow C: E_{K_{c,tgs}}[K_{c,v} || ID_v || TS_4 || Ticket_v]$,

$Ticket_v = E_{K_v}[K_{c,v} || ID_c || AD_c || ID_v || TS_4 || Lifetime_4]$

  - The TGT message is encrypted with the session key shared by the TGS and C and includes a session key to be shared between C and the server V, the ID of V, and the timestamp of the ticket. The ticket itself includes the same session key.
  - Client C now has a reusable service-granting ticket for V.

Client/Server Authentication Exchange

- To obtain Service
  - When C presents this ticket, it also sends an authenticator.

5) C → V: $Ticket_v || Authenticator_c$

$Ticket_v = E_{K_v}[K_{c,v} || ID_c || AD_c || ID_v || TS_4 || Lifetime_4]$

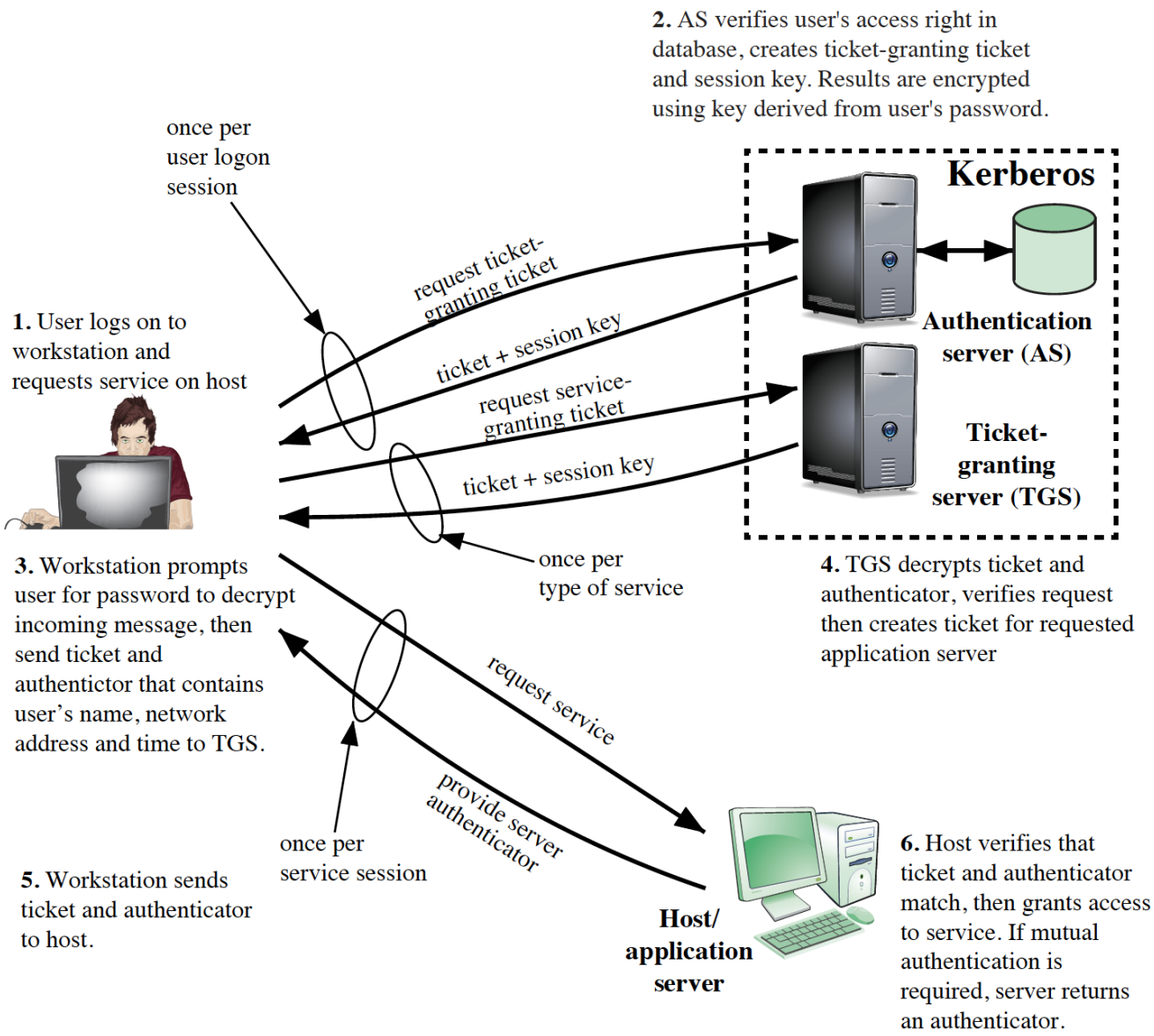
$Authenticator_c = E_{K_{c,v}}[ID_c || AD_c || TS_5]$

  - The server can decrypt the ticket, recover the session key, and decrypt the authenticator.
  - If mutual authentication is required, the server returns the value of the timestamp from the authenticator, incremented by 1, and encrypted in the session key.

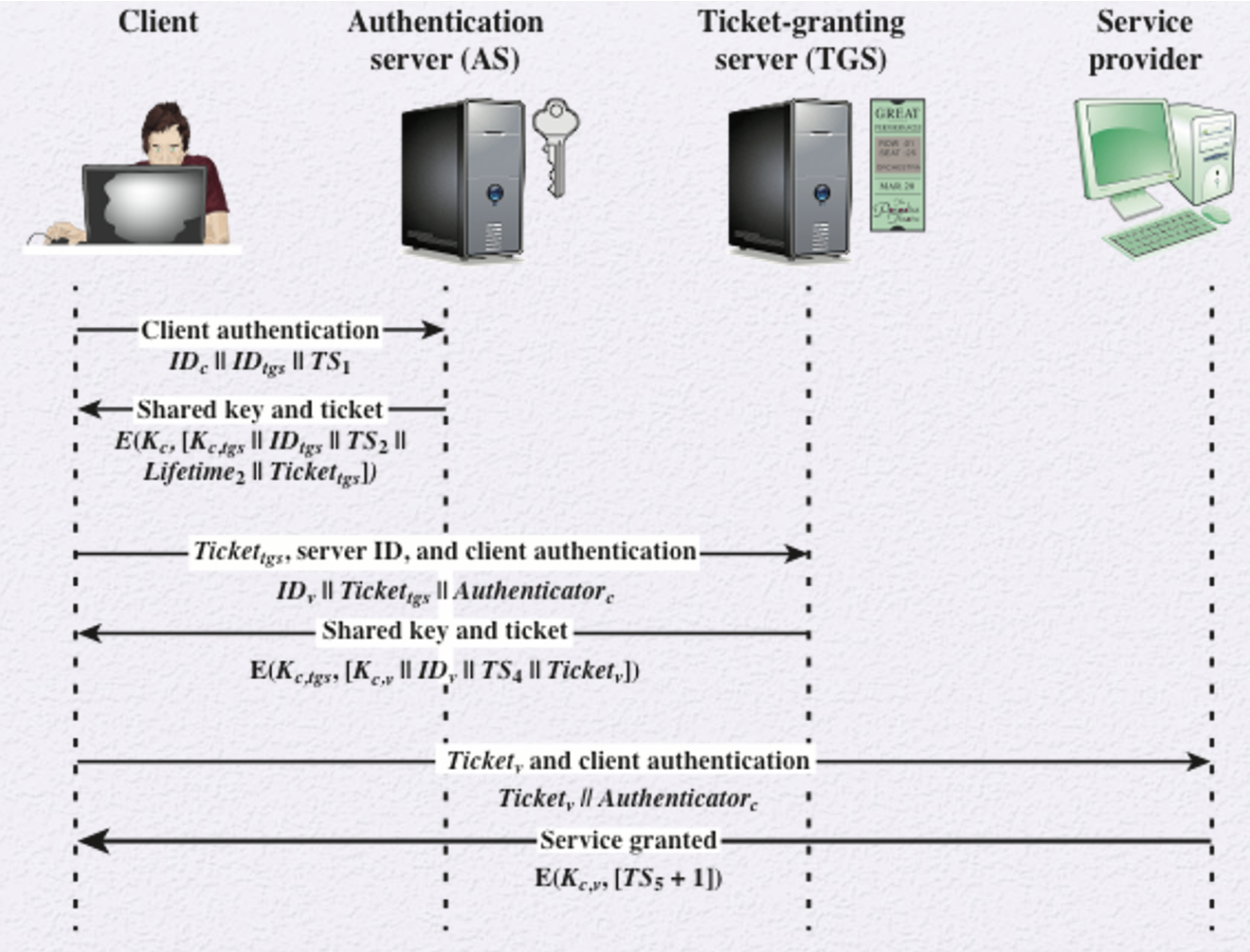
6) V → C: $E_{K_{c,v}}[TS_5+1]$

  - The client can decrypt this message to recover the incremented timestamp. Because the message was encrypted by the session key, C is assured that it could have been created only by V. The contents of the message assure C that this is not a replay of an old reply.

Overview of Kerberos



Kerberos Exchanges



Rationale for the elements of the Kerberos Ver. 4 Protocol

Message (1)	Client requests ticket-granting ticket.
ID_C	Tells AS identity of user from this client.
ID_{tgs}	Tells AS that user requests access to TGS.
TS_1	Allows AS to verify that client's clock is synchronized with that of AS.
Message (2)	AS returns ticket-granting ticket.
K_c	Encryption is based on user's password, enabling AS and client to verify password, and protecting contents of message (2).
$K_{c,tgs}$	Copy of session key accessible to client created by AS to permit secure exchange between client and TGS without requiring them to share a permanent key.
ID_{tgs}	Confirms that this ticket is for the TGS.
TS_2	Informs client of time this ticket was issued.
$Lifetime_2$	Informs client of the lifetime of this ticket.
$Ticket_{tgs}$	Ticket to be used by client to access TGS.

Message (3)	Client requests service-granting ticket.
ID_V	Tells TGS that user requests access to server V.
$Ticket_{tgs}$	Assures TGS that this user has been authenticated by AS.
$Authenticator_c$	Generated by client to validate ticket .
Message (4)	TGS returns service-granting ticket.
$K_{c,tgs}$	Key shared only by C and TGS protects contents of message (4).
$K_{c,v}$	Copy of session key accessible to client created by TGS to permit secure exchange between client and server without requiring them to share a permanent key.
ID_V	Confirms that this ticket is for server V.
TS_4	Informs client of time this ticket was issued.
$Ticket_V$	Ticket to be used by client to access server V.
$Ticket_{tgs}$	Reusable so that user does not have to reenter password.
K_{tgs}	Ticket is encrypted with key known only to AS and TGS, to prevent Tampering.
$K_{c,tgs}$	Copy of session key accessible to TGS used to decrypt authenticator, thereby authenticating ticket.
ID_C	Indicates the rightful owner of this ticket.
AD_C	Prevents use of ticket from workstation other than one that initially requested the ticket.
ID_{tgs}	Assures server that it has decrypted ticket properly.
TS_2	Informs TGS of time this ticket was issued.
$Lifetime_2$	Prevents replay after ticket has expired.
$Authenticator_c$	Assures TGS that the ticket presenter is the same as the client for whom the ticket was issued has very short lifetime to prevent replay.
$K_{c,tgs}$	Authenticator is encrypted with key known only to client and TGS, to prevent tampering.
ID_C	Must match ID in ticket to authenticate ticket.
AD_C	Must match address in ticket to authenticate ticket.
TS_3	Informs TGS of time this authenticator was generated.

Message (5)	Client requests service.
$Ticket_V$	Assures server that this user has been authenticated by AS.
$Authenticator_c$	Generated by client to validate ticket.
Message (6)	Optional authentication of server to client.
$K_{c,v}$	Assures C that this message is from V.
$TS_5 + 1$	Assures C that this is not a replay of an old reply.
$Ticket_v$	Reusable so that client does not need to request a new ticket from TGS for each access to the same server.
K_v	Ticket is encrypted with key known only to TGS and server, to prevent Tampering.
$K_{c,v}$	Copy of session key accessible to client; used to decrypt authenticator, thereby authenticating ticket.
ID_C	Indicates the rightful owner of this ticket.
AD_C	Prevents use of ticket from workstation other than one that initially requested the ticket.
ID_V	Assures server that it has decrypted ticket properly.
TS_4	Informs server of time this ticket was issued.
$Lifetime_4$	Prevents replay after ticket has expired.
$Authenticator_c$	Assures server that the ticket presenter is the same as the client for whom the ticket was issued; has very short lifetime to prevent replay.
$K_{c,v}$	Authenticator is encrypted with key known only to client and server, to prevent tampering.
ID_C	Must match ID in ticket to authenticate ticket.
AD_c	Must match address in ticket to authenticate ticket.
TS_5	Informs server of time this authenticator was generated.

The Importance of Separate Authentication and Ticket Granting Servers

- *why the Authentication Server (AS) and the Ticket Granting Server (TGS) in Kerberos are two separate servers?*
  - The separation of the AS and TGS roles is a security measure that helps to prevent unauthorized access to network resources and services.
  - AS is responsible for authenticating users and issuing ticket-granting tickets (TGTs) that contain a session key, which can be used to authenticate the user to the TGS.
  - TGS is responsible for granting service tickets to users that contain a secret key used to encrypt communication between the user and the network service.
  - The separation ensures that the TGS only issues service tickets to users who have successfully authenticated with the AS and have a valid TGT, preventing unauthorized access to network resources and services.
  - Combining the AS and TGS roles into a single server would make it more difficult to control access to service tickets and increase the risk of unauthorized access to network resources and services.
  - Separating also ensures that authentication and ticket-granting functions are kept separate, reducing the risk of compromise by attackers.

Kerberos Realms and Multiple Kerberi

- A full-service Kerberos environment consisting of a Kerberos server, a number of clients, and a number of application servers requires that:
  - the Kerberos server to have the user IDs and hashed passwords of all participating users in its database, and to share a secret key with each application server.
    - *all users are registered with the Kerberos server*
    - *all servers are registered with the Kerberos server*

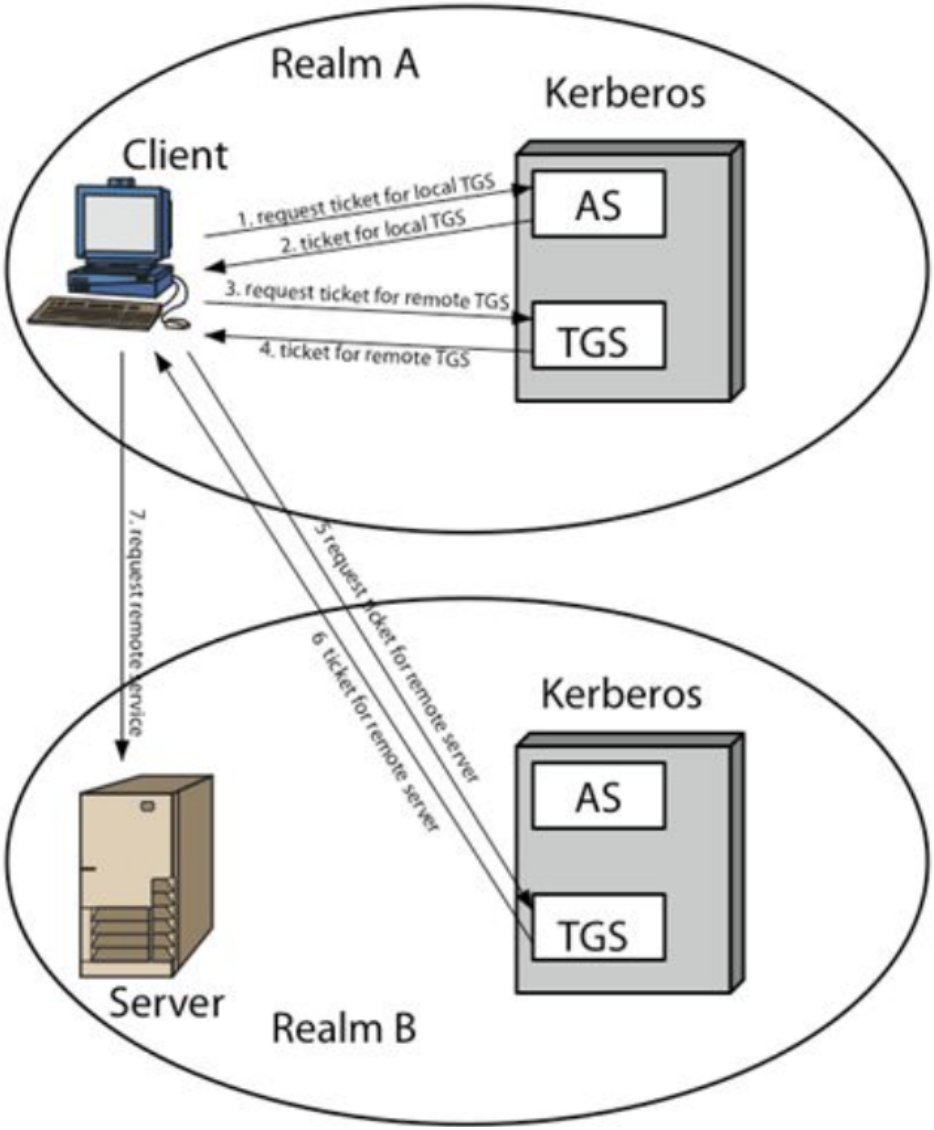
Kerberos Realms and Multiple Kerberi

- In addition, Kerberos supports multiple realms, which are separate administrative domains that can each have their own Kerberos server, clients, and application servers.
- Kerberos realms can interoperate with each other, but this requires that the Kerberos server in each realm share a secret key with the server in the other realm, and that the servers are registered with each other.
  - The scheme requires that the Kerberos server in one realm trust the Kerberos server in the other realm to authenticate its users.
  - In addition, the participating servers in the second realm must also be willing to trust the Kerberos server in the first realm.
- *Kerberos realms and multiple Kerberos servers allow for a more distributed and scalable authentication system, but require careful management of trust relationships and secret key sharing to ensure security.*

Kerberos Realms

- A Kerberos realm is a set of managed nodes that share the same Kerberos database
- The database resides on the Kerberos master computer system, which should be kept in a physically secure room
  - A read-only copy of the Kerberos database might also reside on other Kerberos computer systems
  - All changes to the database must be made on the master computer system
  - Changing or accessing the contents of a Kerberos database requires the Kerberos master password
- Users in one realm may need access to servers in other realms, and some servers may be willing to provide service to users from other realms, provided that those users are authenticated

Kerberos Realms



Key Points

- User presents ID and authentication information to system; system verifies that they are authorized to access
- Authentication information:
  - What you know: passwords
  - What you possess: tokens
  - What you are or do: biometrics
- Always store a hash of a salted password
- Educate users and employ proactive password checking strategies
- Tokens and biometrics can increase security, but at extra cost and inconvenience
- Remote user authentication

Security Issues

- Password selection and usage practices are poor for many systems
- Many vulnerabilities for user authentication techniques; multi-factor authentication adds security
- Check if you have an account that has been compromised in a data breach https://haveibeenpwned.com/
- Secure Password Check. Never enter your real password: https://password.kaspersky.com/

Reading

- Computer Security: Principles and Practice
  - *Chapter 3: User Authentication*