SWE3009: Computer Security
Lecture 0x02: Authentication and Access Control

Hojoon Lee

Systems Security Lab

SUNGKYUNKWAN UNIVERSITY 1398 성균관대학교

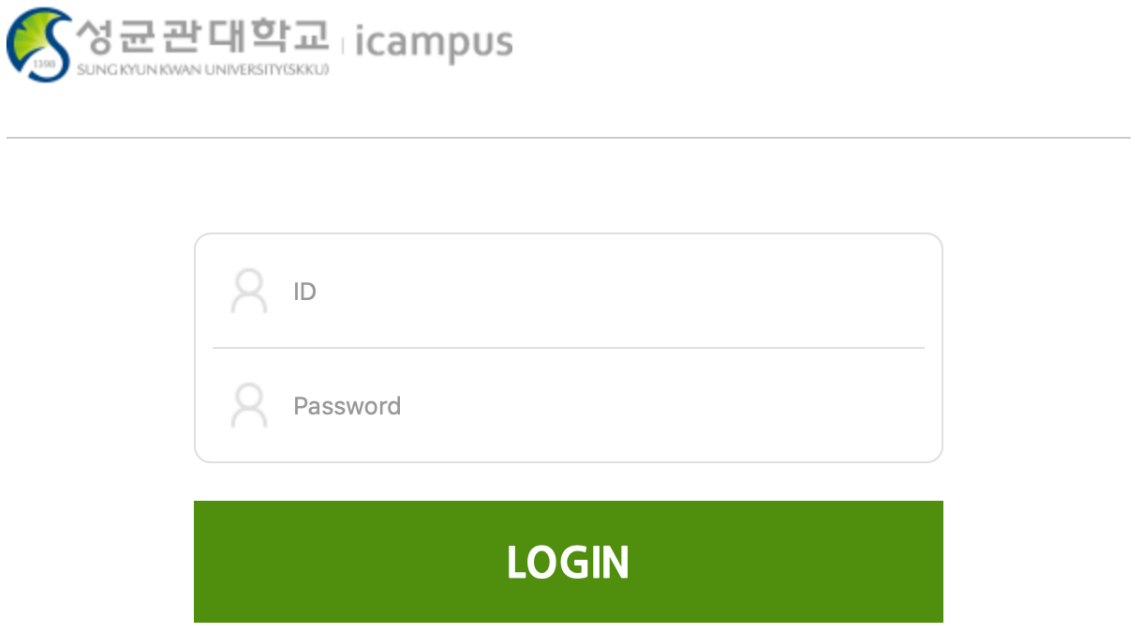
Authentication

What is Authentication

- What is Authentication?

What is Authentication

- What is Authentication?



Are You Who You Say You Are?

- Authenticate a human to a machine?
- Can be based on…
  - Something you know
    - For example, a password
  - Something you have
    - For example, a smartcard
  - Something you are
    - For example, your fingerprint

Something You Know

- Passwords
- Lots of things act as passwords!
  - PIN
  - Social security number
  - Mother's maiden name
  - Date of birth
  - Name of your pet, etc.

Trouble with Passwords

- “Passwords are one of the biggest practical problems facing security engineers today.”
- “Humans are incapable of securely storing high-quality cryptographic keys, and they have unacceptable speed and accuracy when performing cryptographic operations. (They are also large, expensive to maintain, difficult to manage, and they pollute the environment. It is astonishing that these devices continue to be manufactured and deployed.)”

Why Passwords?

- Why is “something you know” more popular than “something you have” and “something you are”?
- Cost: passwords are free
- Convenience: easier for sysadmin to reset pwd than to issue a new thumb

Keys vs Passwords: Security

- Crypto keys
- Private Key in RSA PKI
  - Common standard these days is RSA-2048
  - Government issued PKI in Korea a.k.a 공인인증서
  - Key length of 2048 bits
- Then 2^{2048} keys
- Choose key at random…
- …then attacker must try about 2^{2048} keys

- Passwords
- Made up of usually 6~? characters
- Possible values
  - 26*2 lower and uppercase alphabet
  - 10 numerical digits
  - 32 special characters
  - = 94 possible inputs
- If length is 8, then 94^8

Keys vs Passwords: Security

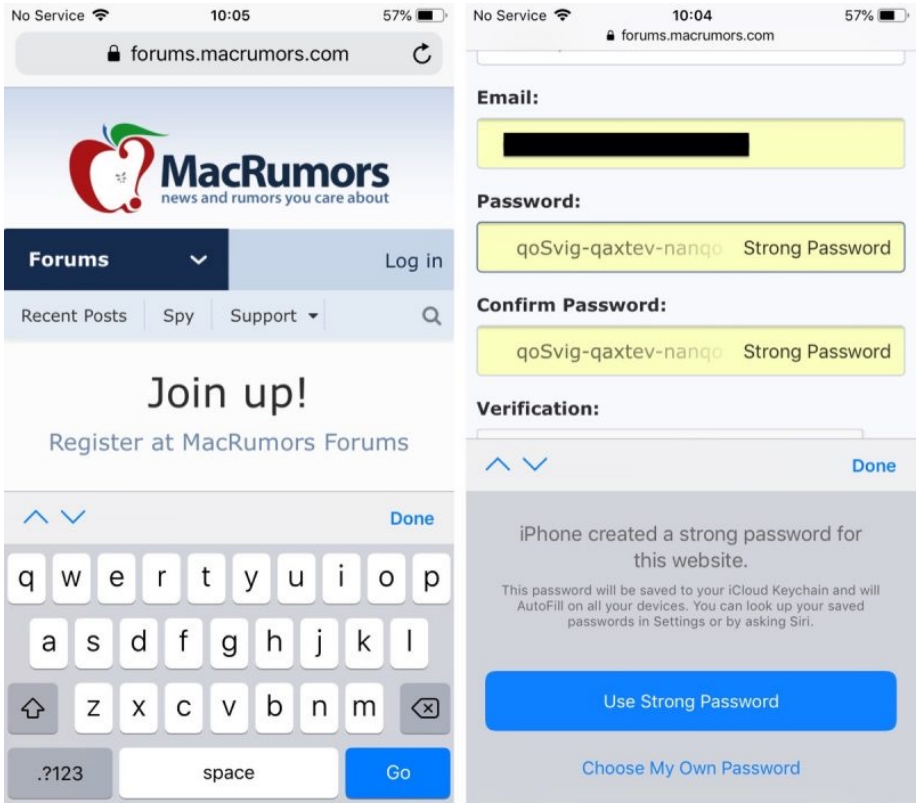
- Crypto keys
- Private Key in RSA PKI
  - Common standard these days is RSA-2048
  - Government issued PKI in Korea a.k.a 공인인증서
  - Key length of 2048 bits
- Then 2^{64} keys
- Choose key at random...
- ...then attacker must try about 2^{63} keys

- Passwords
- Made up of usually 6~? characters
- Possible values
  - 26*2 lower and uppercase alphabet
  - 10 numerical digits
  - 32 special characters
  - = 94 possible inputs
- If length is 8, then 94^8???

Keys vs Passwords: Security

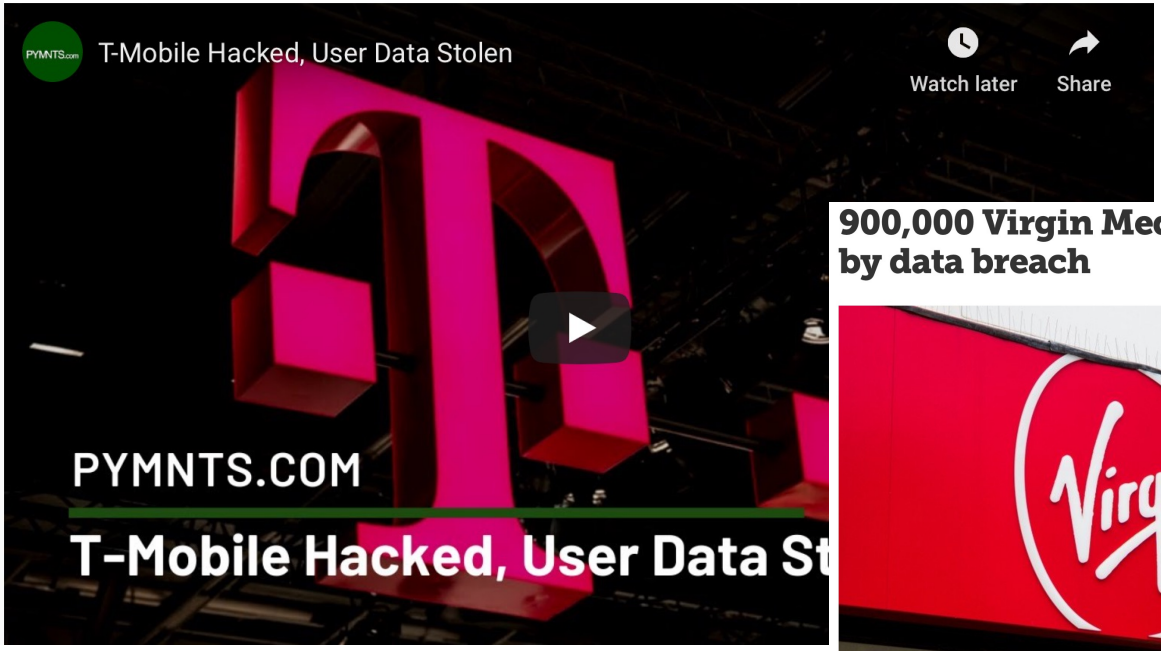
- Users do not select passwords at random
  - password
  - password123
  - qwerty
  - 1234567
  - johndoe
  - ilovebts
  - dropswe3025
- Yes, ^ these are bad passwords but do you make perfectly random passwords?
  - ss9d#k%dkDkS2
  - dke1idU3&12(
  - %234*3kDl2ldis
- Attacker has far less than 94^8 to try

Keys vs Passwords: Security



What if....

- Wait... But isn't it game over if the server itself is breached?



'해킹' 페이스북서 개인정보 털린 한국인 계정 3만5천개

송고시간 | 2018-10-14 17:44

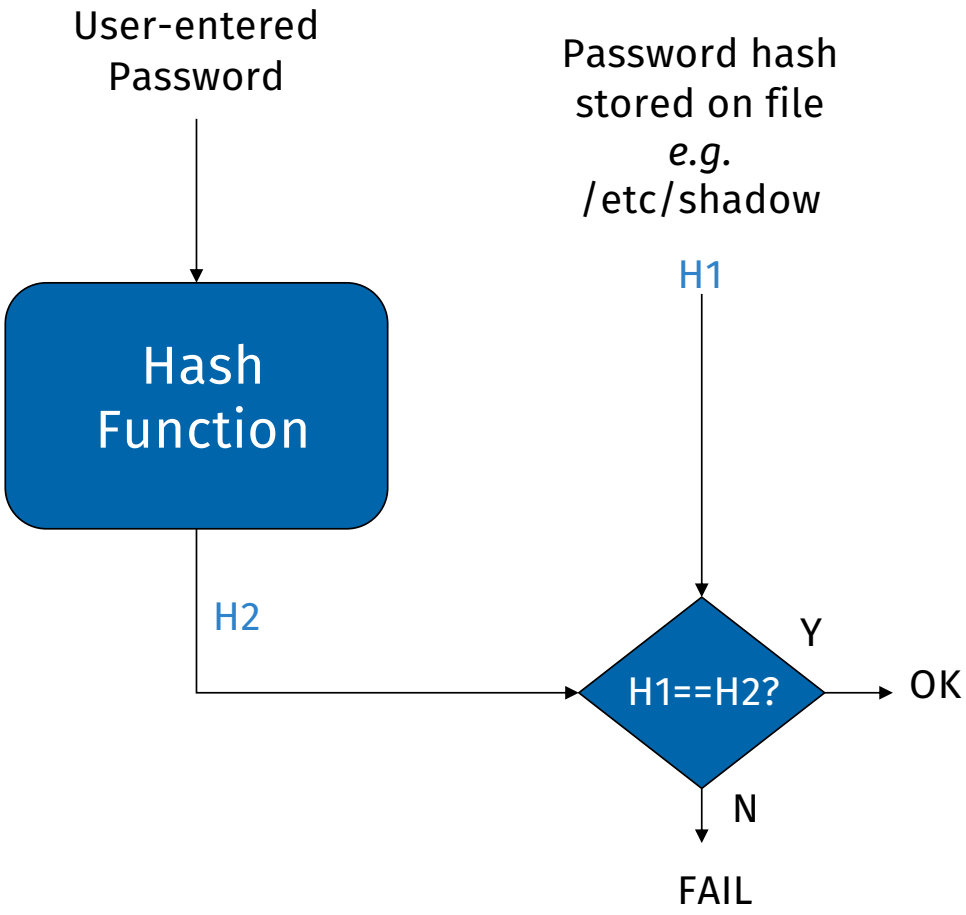
신재우 기자

| 방통위 "유출 규모·경위, 보호조치 준수 여부 조사"

Passwords and Hashes

- Services store hashes of user passwords
- Passwords are turned into hashes using one-way encryption algorithms
  - $y = h(password)$
  - Mathematically proven one way hash functions ensure that a hash of a password can be calculated but not the other way around
    - MD5, SHA1, SHA256 ...
  - e.g. md5("Hello") = "1nDCf/fZO$WxvR2Up9X3dyOGezrH8Lh0"
- But Attacker can try a *forward search*
  - Guess x and check whether $y = h(x)$

Password Verification



Cryptographic Hash Functions (CHF)

- Purpose: produce a fixed-size "fingerprint" or digest of arbitrarily long input data
- Hash passwords such that password plaintext need not be saved on the service or server
- To guarantee integrity

Cryptographic Hash Functions (CHF)

Properties of a good cryptographic HASH function H():

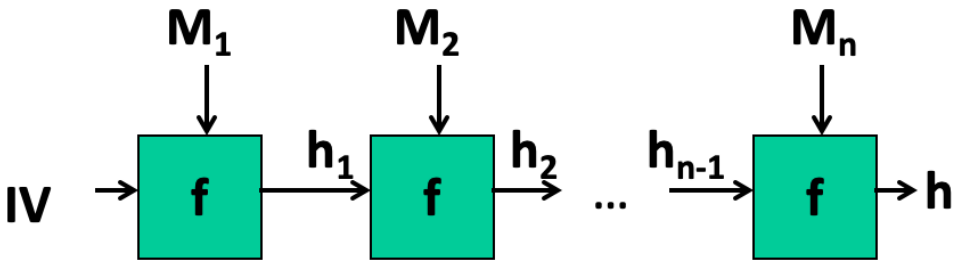
1. Takes input of any size
2. Produces fixed-length output
3. Fast computation
4. Given h, computationally infeasible to find any x such that H(x) = h
5. For a given x, computationally infeasbile to find y such that H(y) = H(x) and y != x
6. Computationally infeasible to find any (x, y) such that H(x) = H(y) and x != y

Cryptographic Hash Functions (CHF)

1. Given h, computationally infeasible to find any x such that H(x) = h
2. For a given x, computationally infeasbile to find y such that H(y) = H(x) and y != x
3. Computationally infeasible to find any (x, y) such that H(x) = H(y) and x != y

- Onewayness guarantee
- Weak Collision Resistance
- Strong Collision Resistance

Cryptographic Hash Functions (CHF)



- A hash function is typically based on internal compression function f() that gets fixed-size input blocks M_i
- Chained Block Cipher
  - Produces hash value for each block based on M_i and hash value from previous block
  - 1-bit difference creates "avalanche effect" makes final output complete different

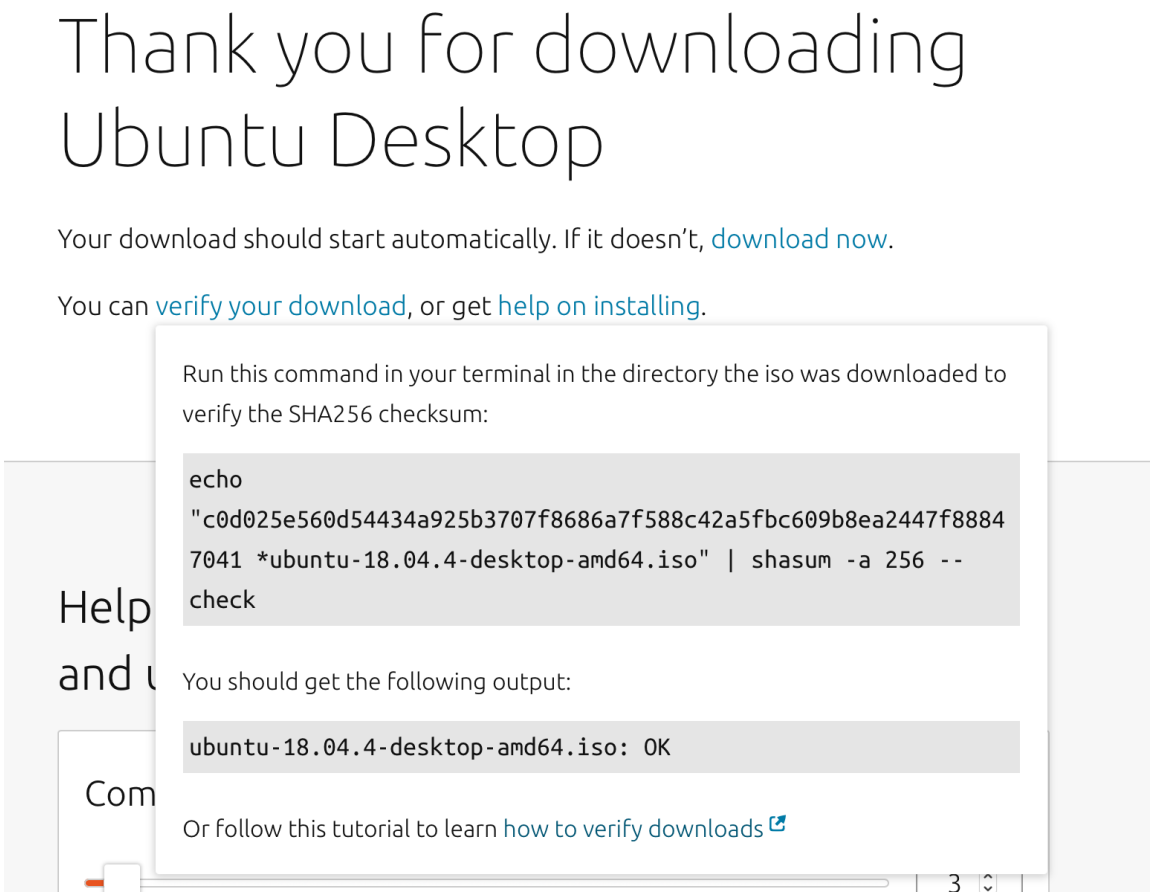
Cryptographic Hash Functions (CHF)

- Weak Resistance
  - Given input **x** to a hash function **H()**
  - It must be infeasible for attacker to find another input **x'** that satisifies **H(x)=H(x')**
- Scenario: attacker has 1) H() 2) H(x) and trying to find **H(x')**

Cryptographic Hash Functions (CHF)

- Strong Resistance
  - Given input hash function **H()**
  - It must be infeasible for attacker to find a pair of input (**x, x'**) that result in **H(x)=H(x')**

Cryptographic Hash Functions (CHF)

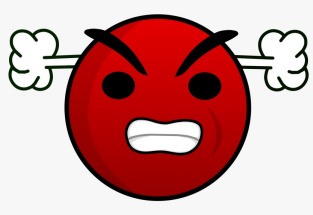


Bruteforcing

- "Trying until getting it right"
- Login screen bruteforcing vs hash bruteforcing
- Theoretically you can recover password by trying all possible combinations
  - Not so easy with today's computation power
  - E.g. $94^8 = 6095689385410816$
- Dictionary Attack
  - Take advantage of fact that people use easy-to-remembers words
  - List of commonly used words (apple, car, ...)
  - Generate possible passwords (apple123, Car ...)
  - Try until getting it right
  - Far more effective than you might think

Bruteforcing Mitigation Measures: Adding Delays

- Adding wait times for retry
  - "Please enter again in 5 seconds..."
- More extreme measures....
  - "Your Account has been locked, please contact .."

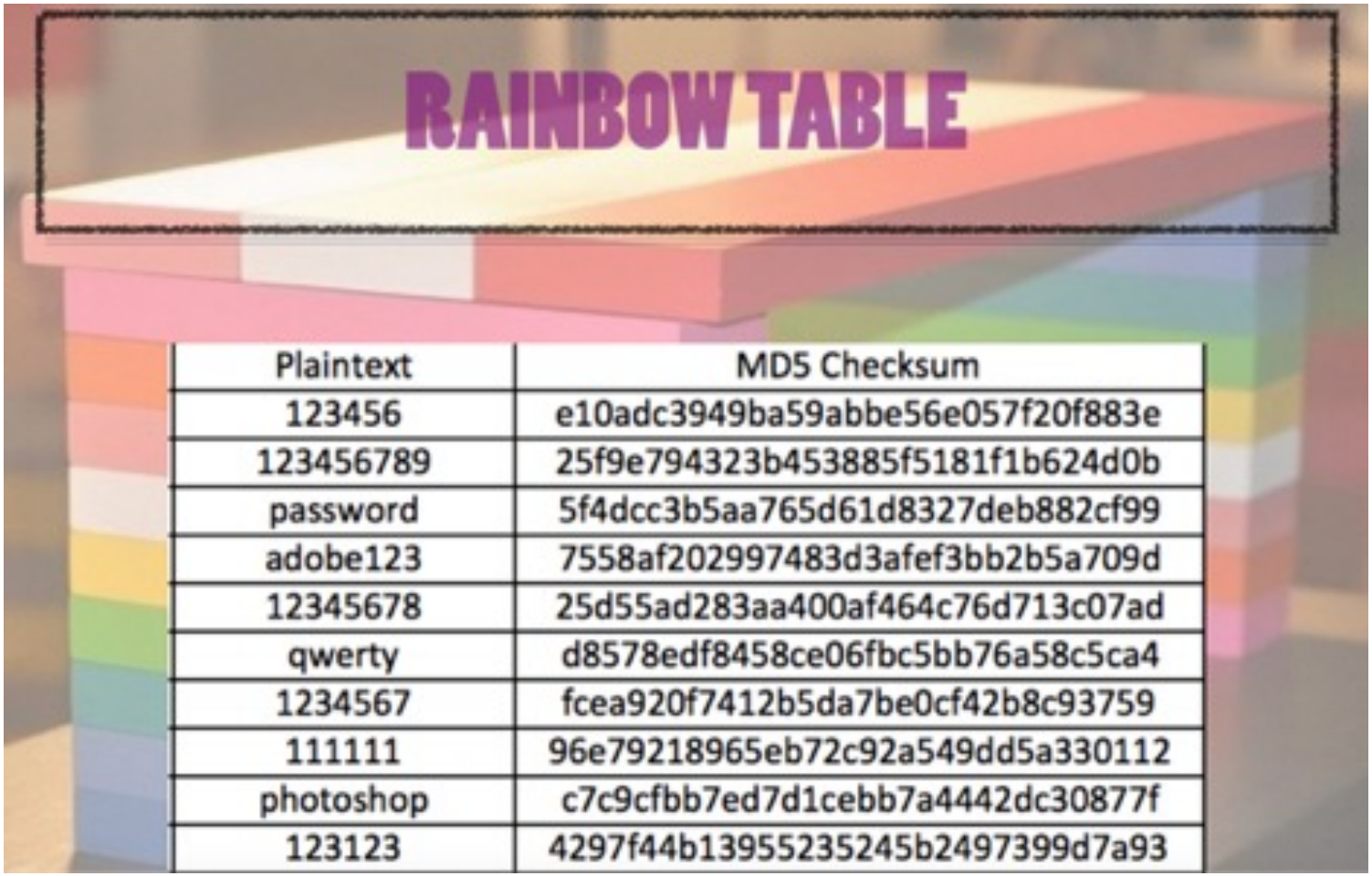


- Not effective if attacker has the hash value of your password
  - Security of your password now lies on its strength now

Rainbow Table Attack

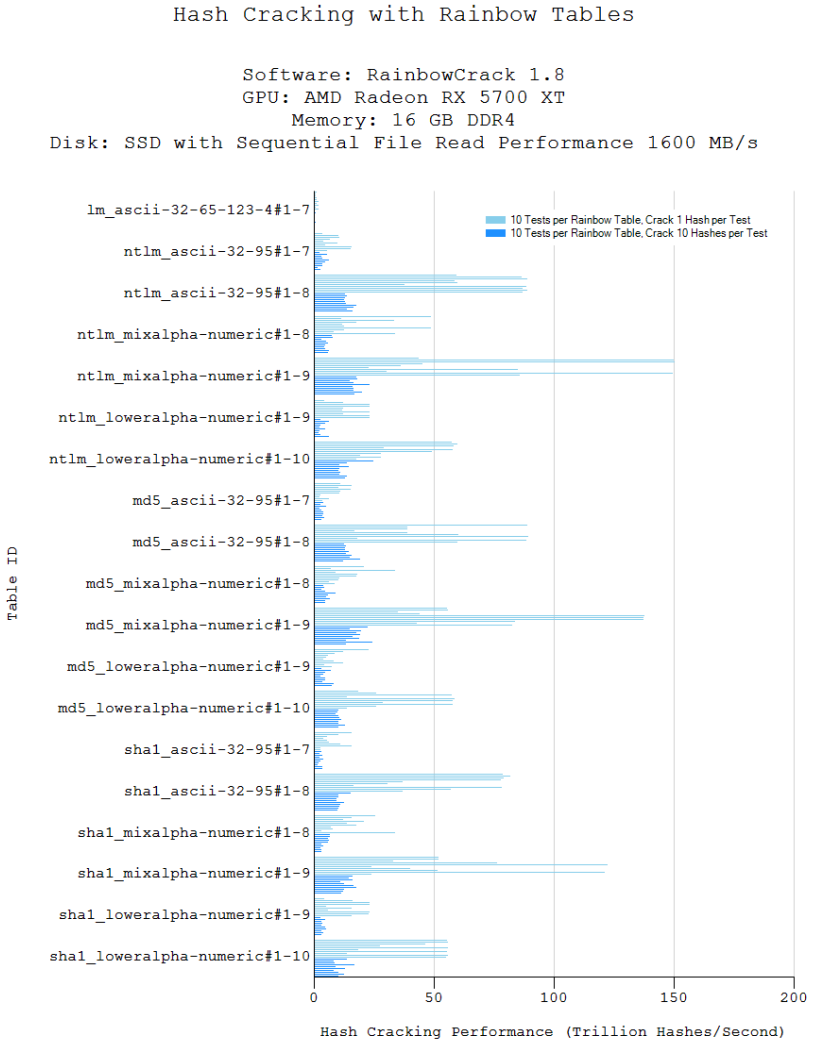
- Attacker pre-generates hashes of commonly used passwords or easy passwords
- all dictionary words (apple,car..) or combinations of words
  - "applejuice", "goodsecurity"
- passwords known to be used by many non-techy people
  - "admin", "password123", "mywifi"
- What if hackers around world decide to pre-compute zillions and zillions of hashes and share among themselves?
- e.g. Imagine all the computation power used for Bitcoin mining is used for pre-computing hashes?

Rainbow Table Attack



Plaintext	MD5 Checksum
123456	e10adc3949ba59abbe56e057f20f883e
123456789	25f9e794323b453885f5181f1b624d0b
password	5f4dcc3b5aa765d61d8327deb882cf99
adobe123	7558af202997483d3afef3bb2b5a709d
12345678	25d55ad283aa400af464c76d713c07ad
qwerty	d8578edf8458ce06fbc5bb76a58c5ca4
1234567	fcea920f7412b5da7be0cf42b8c93759
111111	96e79218965eb72c92a549dd5a330112
photoshop	c7c9cfbb7ed7d1cebb7a4442dc30877f
123123	4297f44b13955235245b2497399d7a93

Rainbow T


Hash Cracking with Rainbow Tables
Software: RainbowCrack 1.8
GPU: AMD Radeon RX 5700 XT
Memory: 16 GB DDR4
Disk: SSD with Sequential File Read Performance 1600 MB/s
10 Tests per Rainbow Table, Crack 1 Hash per Test
10 Tests per Rainbow Table, Crack 10 Hashes per Test
Table ID
lm_ascii-32-65-123-4#1-7
ntlm_ascii-32-95#1-7
ntlm_ascii-32-95#1-8
ntlm_mixalpha-numeric#1-8
ntlm_mixalpha-numeric#1-9
ntlm_loweralpha-numeric#1-9
ntlm_loweralpha-numeric#1-10
md5_ascii-32-95#1-7
md5_ascii-32-95#1-8
md5_mixalpha-numeric#1-8
md5_mixalpha-numeric#1-9
md5_loweralpha-numeric#1-9
md5_loweralpha-numeric#1-10
sha1_ascii-32-95#1-7
sha1_ascii-32-95#1-8
sha1_mixalpha-numeric#1-8
sha1_mixalpha-numeric#1-9
sha1_loweralpha-numeric#1-9
sha1_loweralpha-numeric#1-10
0
50
100
150
200
Hash Cracking Performance (Trillion Hashes/Second)

Bruteforcing Mitigation Measures: Salt

- Hash password with salt
- Choose random salt **s** and compute

  $$y = h(password, s)$$

  and store **(s,y)** in the password file
- Note that the salt **s** is not secret

Bruteforcing Mitigation Measures: Salt

- Again a salt is not a secret
- Salt is also leaked along with password hashes when server's database is breached
- Then why use a salt?

Bruteforcing Mitigation Measures: Salt

- 1. Prevents identical passwords to be recognized
  - SWE3025 Student1 : MDPJ80rH$mcHzuxrOMf3l3GGf6DzE/0.
  - SWE3025 Student2 : MDPJ80rH$mcHzuxrOMf3l3GGf6DzE/0.
  - Attacker: "Must be something about the course!!"
  - MDPJ80rH$mcHzuxrOMf3l3GGf6DzE/0 = "ihateprofessorlee"
  - Password Cracked...

Bruteforcing Mitigation Measures: Salt

2. Prevention of pre-computed hash attacks

- Rainbow table attack becomes significantly more difficult if a random salt is added to the password during hashing
- H("easy_password") is completely different from H("easy_password+dk3skd@")
- Imagine that attacker knows the salt value ("dk3skd@")
- and has the hash value of "easy_password" in RBT, attacker still has to do a lot of computation to find out that the password

Tips on Making Passwords

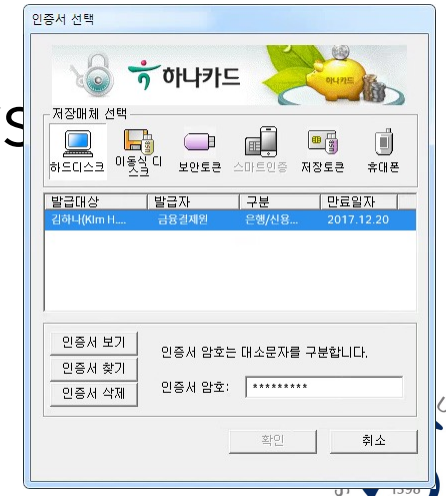
- Longer passwords are better but rememberability is also important
- Include lower-case,upper-case,number, and special characters
  - e.g. lower case alphabets only: 26^n
  - e.g. upper AND lower alphabets: 52^n
  - e.g. combination of all 93^n
- Don't use same password everywhere

How I Make Passwords

- Classified top-secret only shared with SWE3025 students
- Create a password at least 6-10 letters with
  - at least 1 capital letter
  - at least 1 special character
  - at least 1 number
- Add my own per-site salt at the end
  - e.g. password for skku
    - 스꾸 -> tmRN
    - add tmRn at the end
  - Now you know my password for skku is *********tmRN ☺

Cryptographic Authentication

- Introduction to Cryptography is for another time
- TLS used for HTTPS uses Public Key Infrastructure (PKI)
- You use a private key with a password to authenticate yourself to Korean banking sys (**공인인증서**)

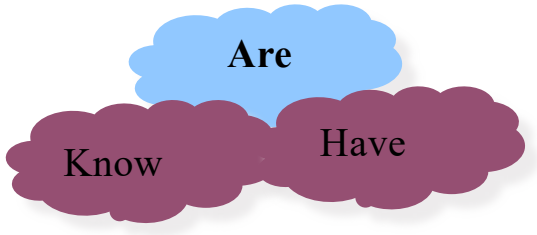


Something You Are

- Biometric
  - "You are your key" — Schneier

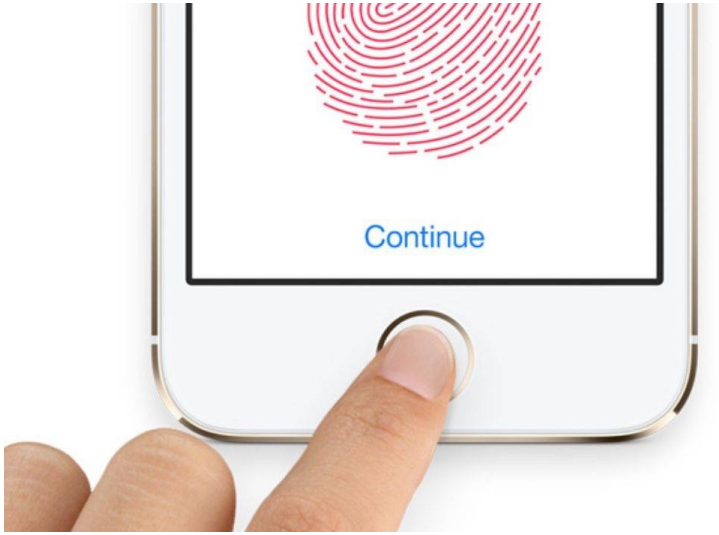
Examples

- Fingerprint
- Handwritten signature
- Facial recognition
- Speech recognition
- Gait (walking) recognition
- "Digital doggie" (odor recognition)
- Many more!



Biometric Authentication

- Biometric authentication is becoming more and more common



Ideal Biometric

- Universal — applies to (almost) everyone
  - In reality, no biometric applies to everyone
- Distinguishing — distinguish with certainty
  - In reality, cannot hope for 100% certainty
- Permanent — physical characteristic being measured never changes
  - In reality, OK if it to remains valid for long time
- Collectable — easy to collect required data
  - Depends on whether subjects are cooperative
- Also, safe, user-friendly, and ???

Fingerprint Comparison

- Examples of loops, whorls, and arches
- Minutia extracted from these features

Loop (double)

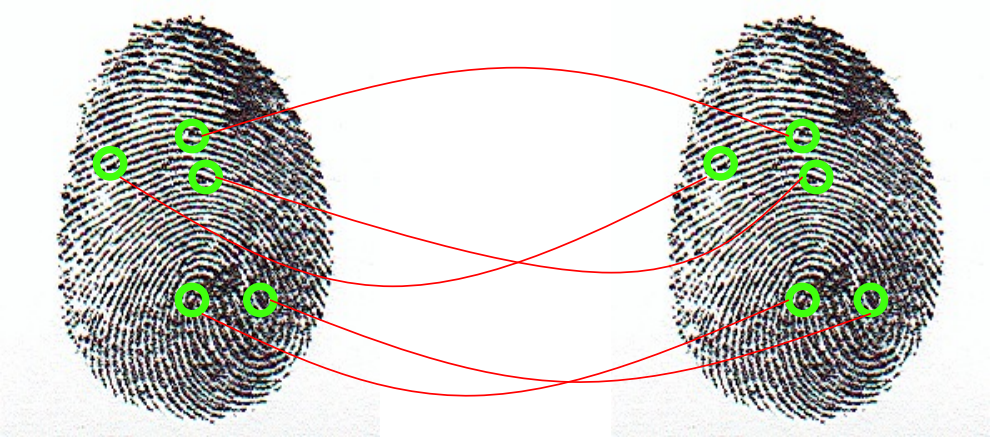
Whorl

Arch

Fingerprint: Enrollment

- Capture image of fingerprint
- Enhance image
- Identify “points”

Fingerprint: Recognition



- Extracted points are compared with information stored in a database
- Is it a statistical match?
- Aside: Do identical twins' fingerprints differ?

Vein Authentication

- A popular biometric
- Measures pattern of veins in your wrist
- Fast and accurate?

Vein Authentication: FAIL

Hackers use a fake wax hand to fool veir authentication security

It was done using modified consumer tech

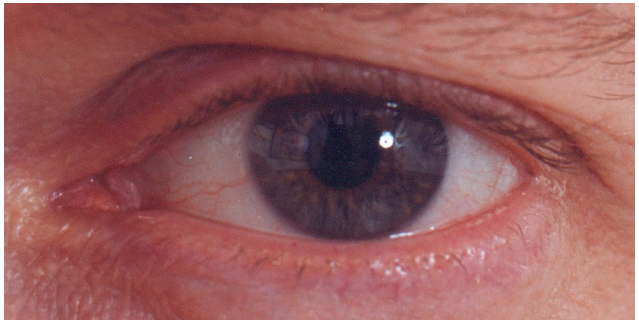
By Jon Porter | @JonPorty | Dec 31, 2018, 5:20am EST

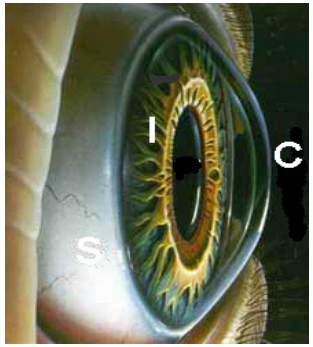
SHARE

Image: Krissler, Albrecht

Iris Patterns

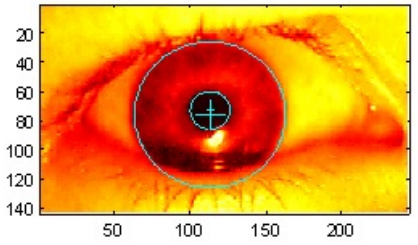
- Iris pattern development is “chaotic”
- Little or no genetic influence
- Even for identical twins, uncorrelated
- Pattern is stable through lifetime

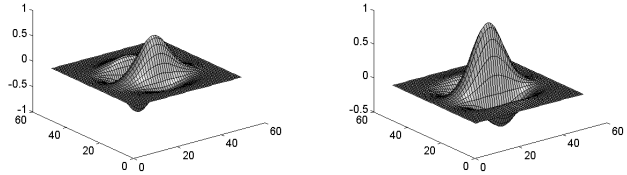


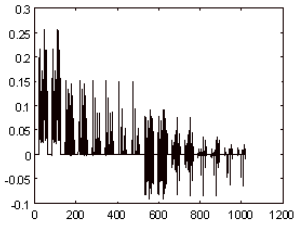


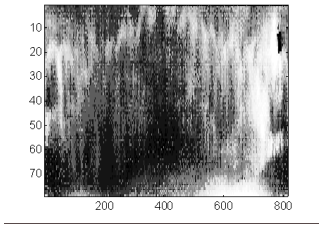
Iris Scan

- Scanner locates iris
- Take b/w photo
- Use polar coordinates...
- 2-D wavelet transform
- Get 256 byte iris code









Attack on Iris Scan

- Good photo of eye can be scanned
  - Attacker could use photo of eye

- Afghan woman was authenticated by iris scan of old photo
- To prevent attack, scanner could use light to be sure it is a “live” iris

Biometrics: The Bottom Line

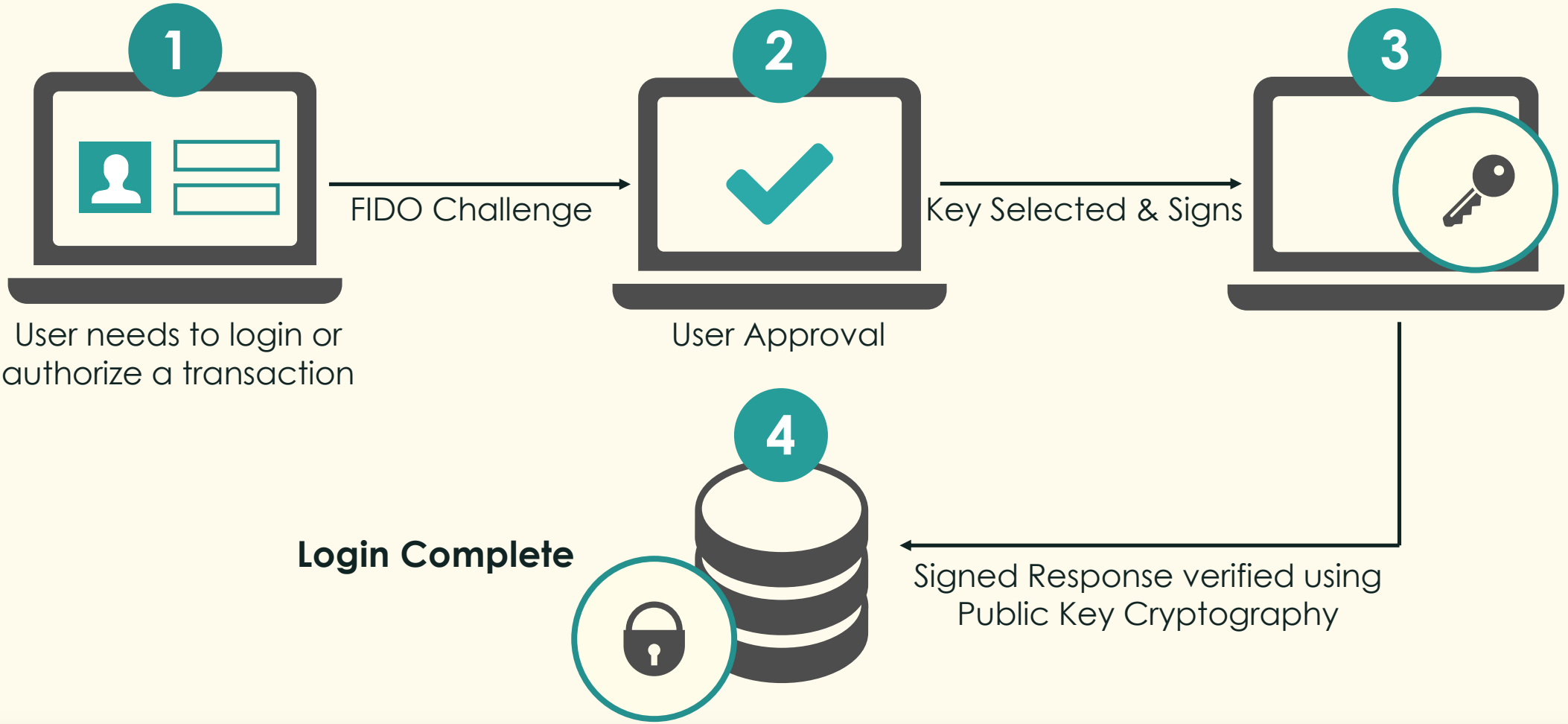
- Biometrics are hard to forge
- But attacker could
  - Steal Alice's thumb
  - Photocopy Bob's fingerprint, eye, etc.
  - Subvert software, database, "trusted path" ...
- And how do we revoke a leaked biometric?

Something You Have

- Something in your possession

YubiKey

FIDO Authentication



Password card and OTP

2-factor Authentication

- Requires any **2** out of **3** of
  - Something you **know**
  - Something you **have**
  - Something you **are**
- Examples
  - ATM: Card and PIN
  - Credit card: Card and signature
  - Password generator: Device and PIN
  - Smartcard with password/PIN

Single Sign-on

- A hassle to enter password(s) repeatedly
  - Alice would like to authenticate only once
  - “Credentials” stay with Alice wherever she goes
  - Subsequent authentications transparent to Alice
- Kerberos — a single sign-on protocol

Web Cookies

- Cookie is provided by a Website and stored on user's machine
- Cookie indexes a database at Website
- Cookies maintain state across sessions
  - Web uses a stateless protocol: HTTP
  - Cookies also maintain state within a session
- Sorta like a single sign-on for a website
  - But, very, very weak form of authentication
- Cookies also create privacy concerns

Coming up in Next Lecture

- Mandatory Access Control(MAC)
- Role-based Access Control (RBAC)
- (Traditional) Access Control in Unix/Linux
- Modern Access Control in Unix/Linux
- Modern Access Control Examples
  - Containers (e.g., Docker)
  - Virtualization
  - ETC ...