Lecture 6 Authentication

- COMP 6712 Advanced Security and Privacy

Haiyang Xue

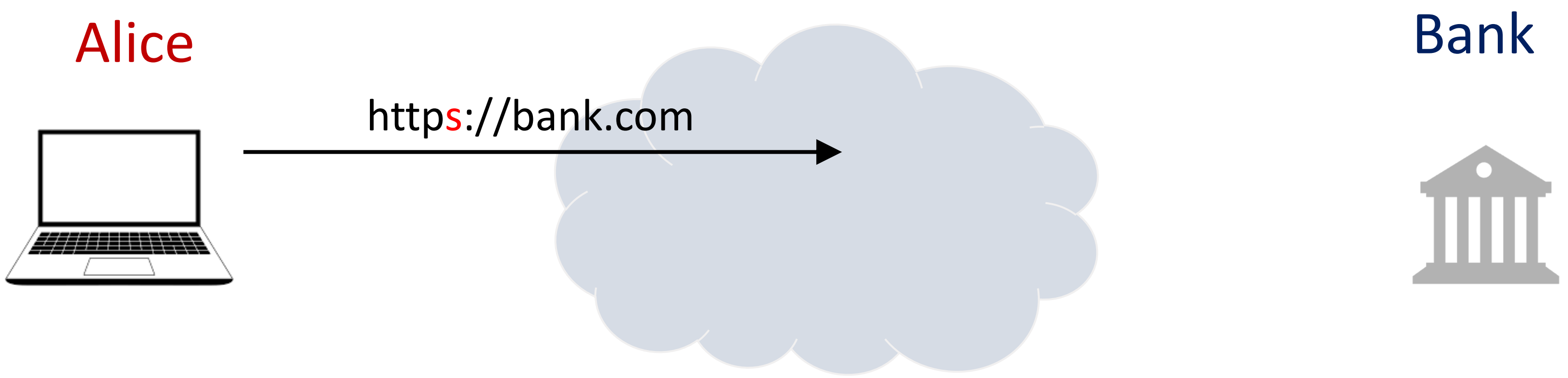
haiyang.xue@polyu.edu.hk

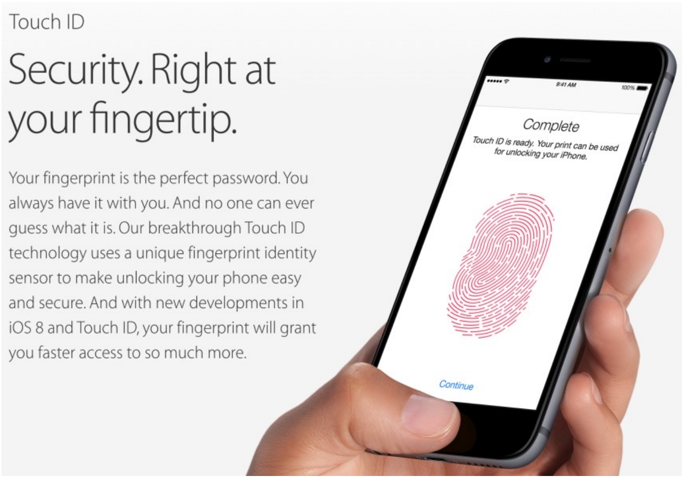
2024/2/26

Authentication

- What is authentication
- Password Authentication
  - Password requirements/strength
  - How is the password stored?
  - Attacks on password
- Biometric Authentication
- Public key Authentication
  - Web Authentication
  - SSH

In practice

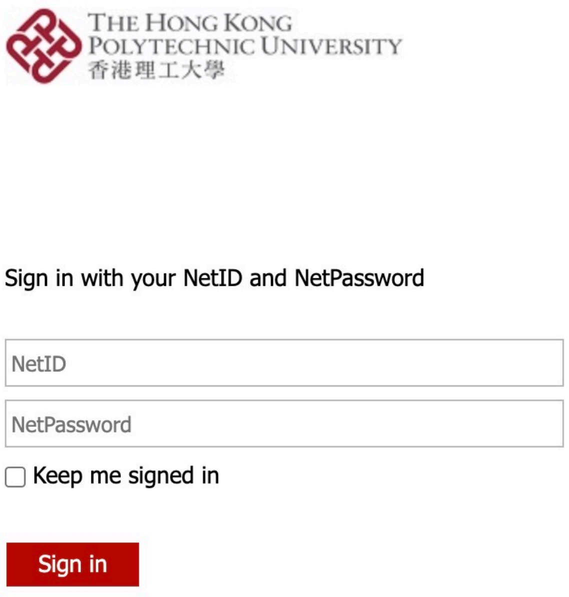



Touch ID

Security. Right at your fingertip.

Your fingerprint is the perfect password. You always have it with you. And no one can ever guess what it is. Our breakthrough Touch ID technology uses a unique fingerprint identity sensor to make unlocking your phone easy and secure. And with new developments in iOS 8 and Touch ID, your fingerprint will grant you faster access to so much more.

Complete
Touch ID is ready. Your print can be used for unlocking your iPhone.
Continue


THE HONG KONG POLYTECHNIC UNIVERSITY
香港理工大學

Sign in with your NetID and NetPassword

NetID

NetPassword

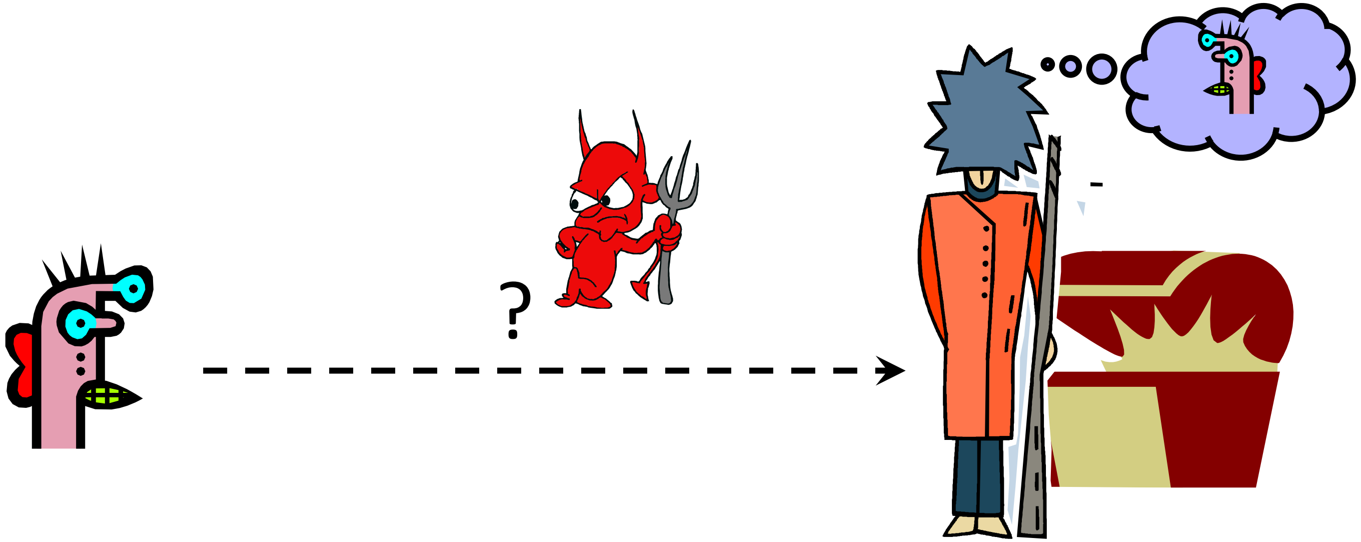
□ Keep me signed in

What is Authentication?

- is the act of proving an assertion, such as the identity of a computer system user

- the process of verifying someone or something’s identity

The Core Problem



How do you prove to someone that you are who you claim to be?

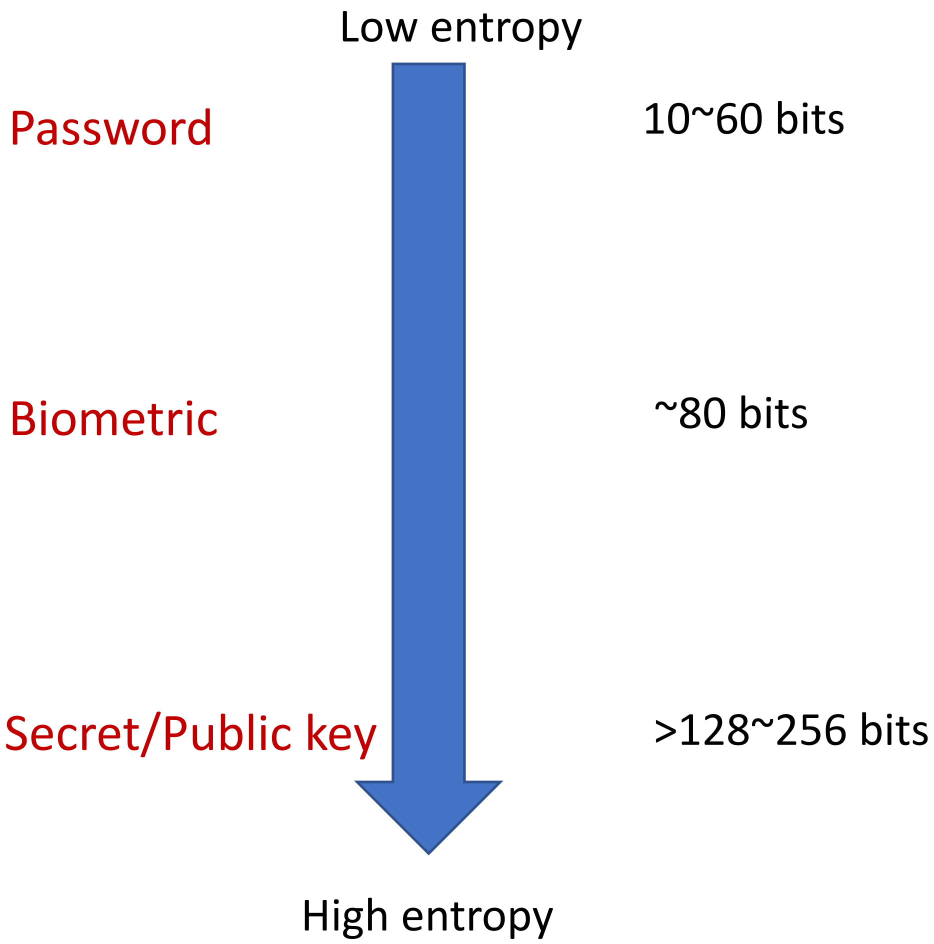
Any system with access control must solve this problem.

Factors

- **Idea:** Verify the user is who they say they are

- Authentication systems classically use three **factors**:
  - Something you know (e.g. a password)
  - Something you are (e.g. a fingerprint or other biometric data)
  - Something you have (e.g. a phone, SecurID or cryptographic secret key)

Factors



Low entropy

Password — 10~60 bits

Biometric — ~80 bits

Secret/Public key — >128~256 bits

High entropy

The Shannon entropy of a random variable

$$H(X) = -\sum p(x) \log p(x)$$

Ex:

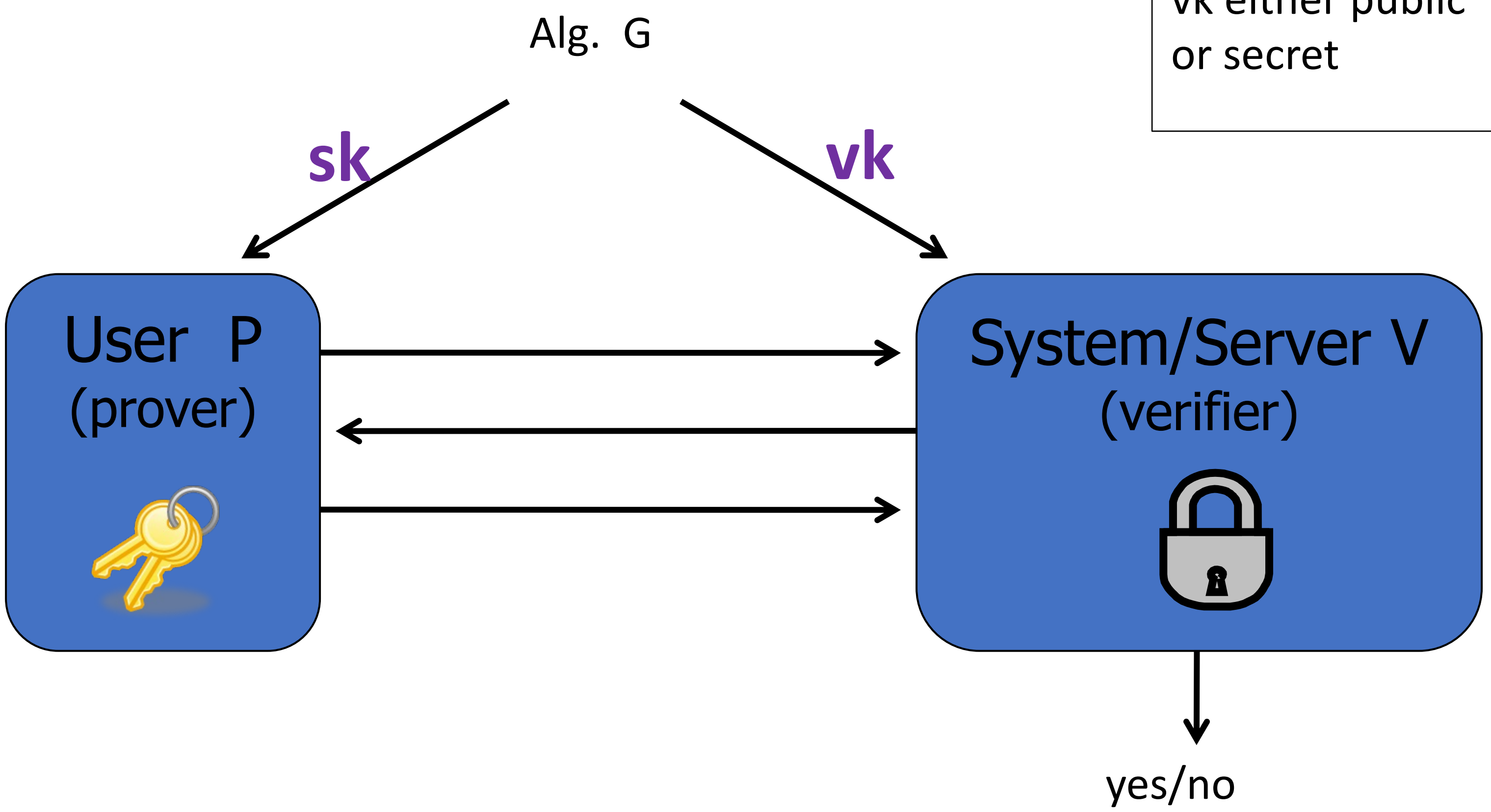
Authentication vs Authorization vs Access control

- **Authentication**: is the user (or program) who they claim they are?

- **Authorization**: should user (or program) have access to a given resource?
  - Authorization decisions rely on correct authentication

- **Access control**: policy and enforcement mechanism to allow authorized access

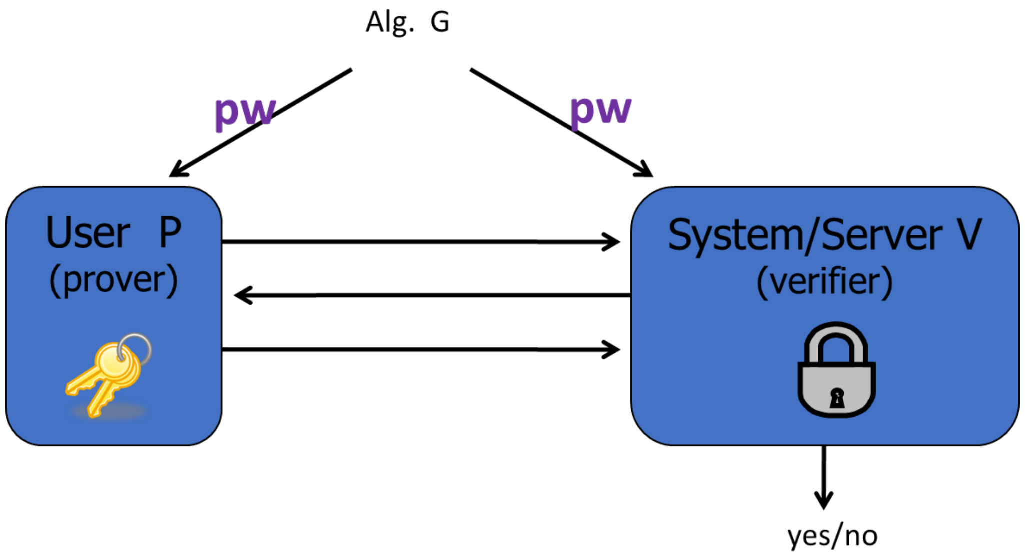
Authentication paradigm



vk either public
or secret

Password Authentication

- User has a secret password;
- System checks it to authenticate the user.



- Easy to deploy
- Easy to use (nothing to carry, etc.)
- No simple alternative

Chosen password requirements/password strength

How do people pick their passwords?

Often they don’t!

- Surveys show that half of users leave the default password in place for their routers at home.

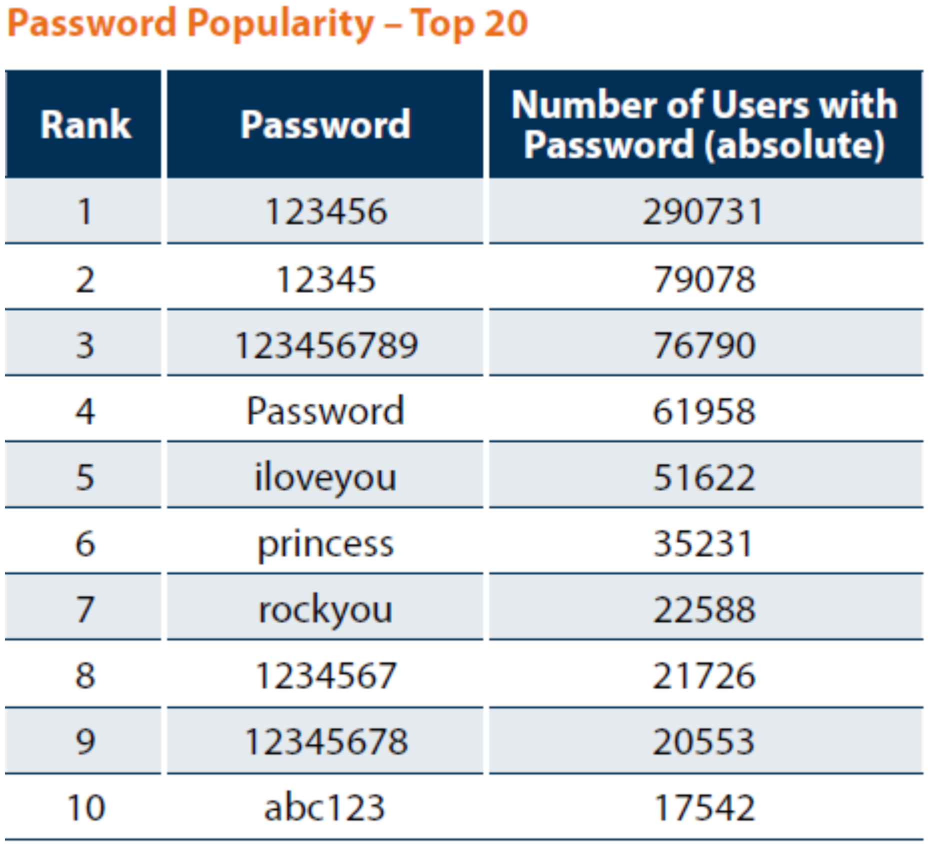
- Dixie bank: 99% of employees used password “password123”!

A. Tsow et al., “Warkitting: the Drive-by Subversion of Wireless Home Routers.” The Journal of Digital Forensic Practice, 2006!
B. Kevin Mitnick: Art of Intrusion

Another way

- RockYou was hacked in December 2009
- Disclosed 32 million user passwords; posted to internet
- Passwords were in clear (not hashed or encrypted)
- Main source today of research / knowledge about user password composition

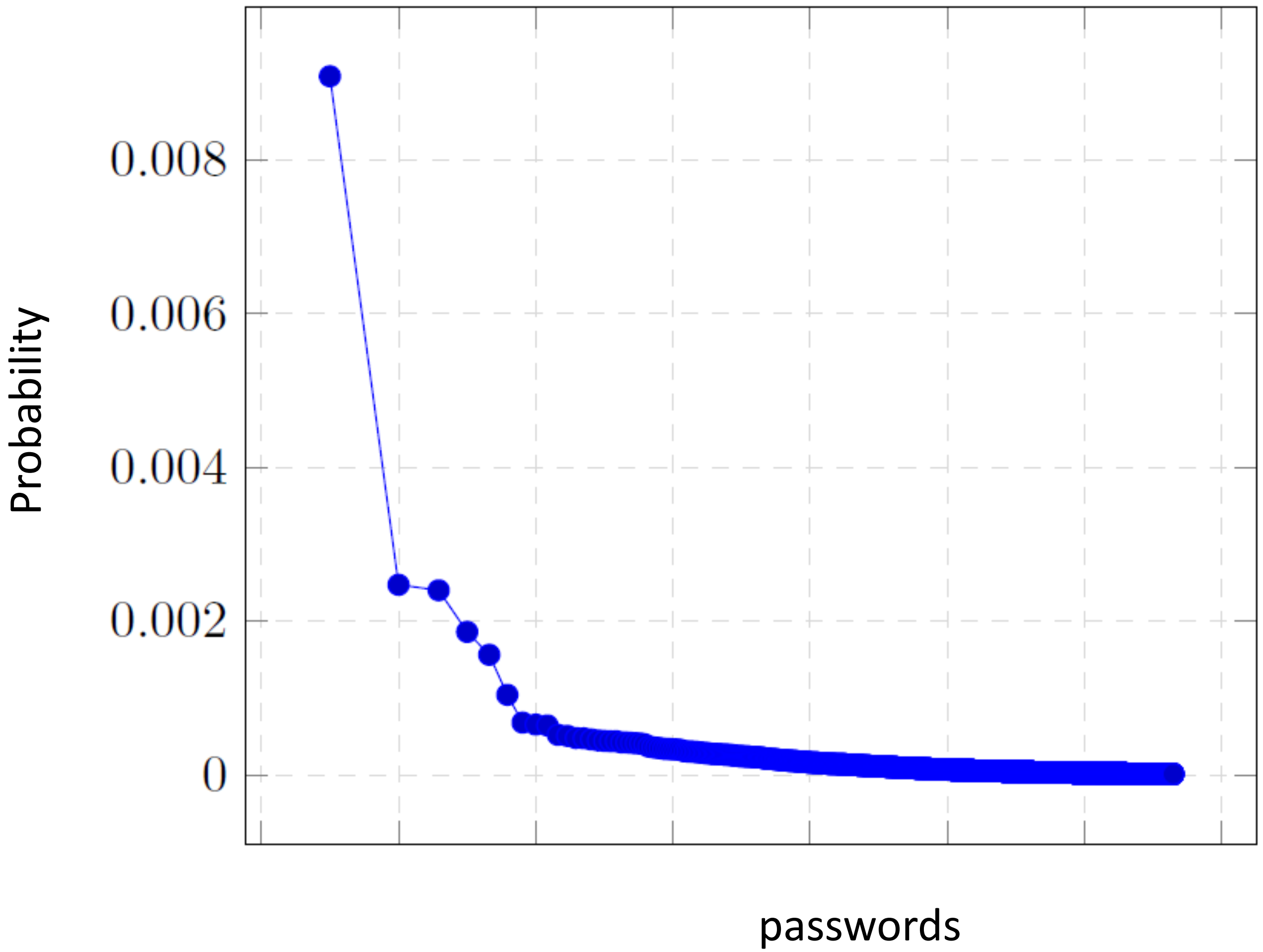
Learn from RockYou



Password Popularity – Top 20

Rank	Password	Number of Users with Password (absolute)
1	123456	290731
2	12345	79078
3	123456789	76790
4	Password	61958
5	iloveyou	51622
6	princess	35231
7	rockyou	22588
8	1234567	21726
9	12345678	20553
10	abc123	17542

Top 10 RockYou password



Measuring password strength: Entropy

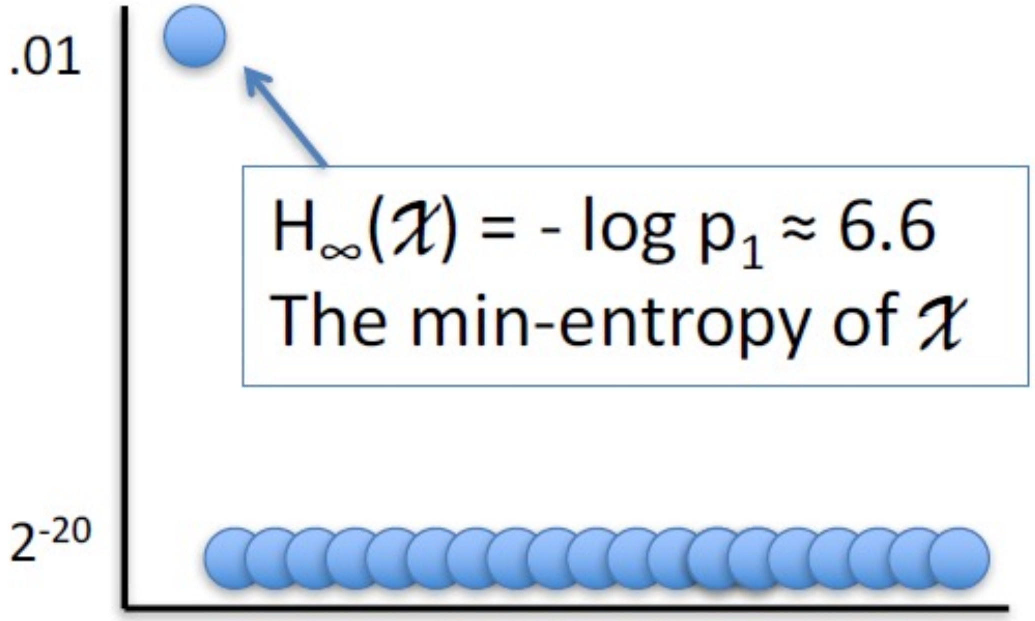
- Many ways to measure password strength
- Shannon Entropy:

- Let X be password distribution. Passwords are drawn from X
- n is size of support of X
- $p_1, p_2, \ldots, p_n$ are probabilities of passwords in decreasing order

$$H(X) = -\sum p_i \log p_i$$

Shannon entropy is a poor measure

- n = 1,000,000
- p_1= 1 / 100
- p_2 = (1 – 1/100)/999,999 ≈ 1 / 220
- …
- p_n = (1 – 1/100)/999,999 ≈ 1 / 220



$H_\infty(\mathcal{X})$ = - log p_1 ≈ 6.6
The min-entropy of $\mathcal{X}$

$H(X) \approx 19$

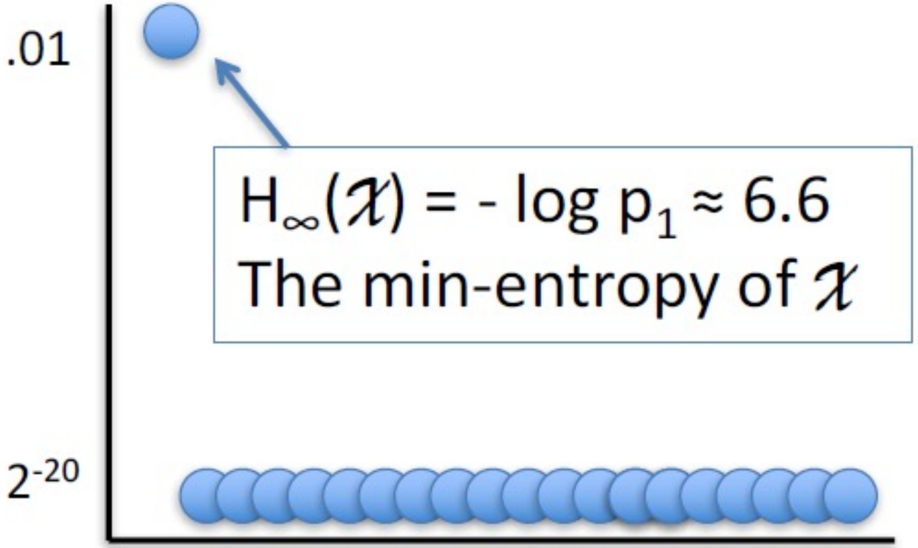
19 bits of “unpredictability”? i.e, $1/2^{19}$ It is not the truth.
Adversary will guess the “password1” with prob. 1/100

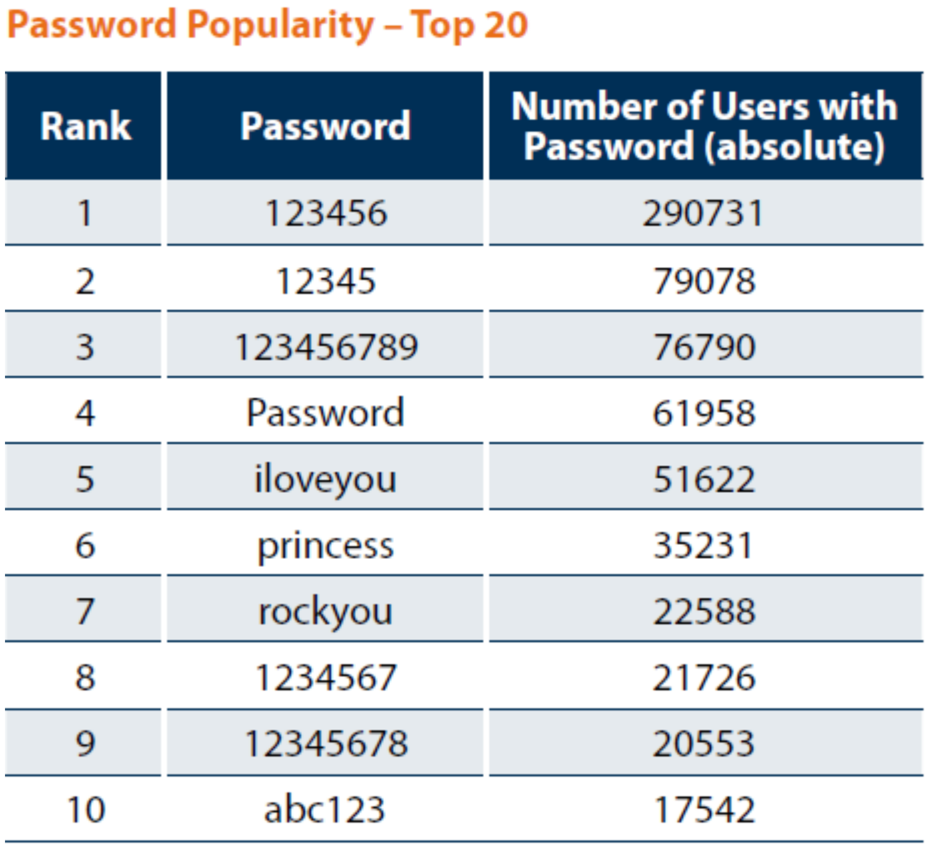
One important type

- *Min-entropy:* related to commonness of most popular password
- “guessing probability” or GP denote probability of most probable password over a population

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

- GP = Max probability is 2^{- $H_{min}\ (X)$}.





Password Popularity – Top 20

Rank	Password	Number of Users with Password (absolute)
1	123456	290731
2	12345	79078
3	123456789	76790
4	Password	61958
5	iloveyou	51622
6	princess	35231
7	rockyou	22588
8	1234567	21726
9	12345678	20553
10	abc123	17542

Top 10 RockYou password

GP = 0.9%; i.e., 0.9% of users, about 1 in 111, have this password!

GP measures vulnerability of the weakest accounts, which can be best for an attacker to target.

Practical Recommendations by system

- To help users create stronger passwords, system administrators often require passwords to exceed a certain length, contain at least a specific number of character classes, or not appear on a blocklist

- Recent paper suggests 1c12+NN10
- 1c12: 1 class with at least 12 characters
- NN10 required passwords to have password strength estimates no weaker than 10^10 guesses

Practical Recommendations for Stronger, More Usable Passwords Combining Minimum-strength, Minimum-length, and Blocklist Requirements. ACM CCS 2020

How is the password stored?

- **Important:** Never, never, never store passwords in plaintext

- **Otherwise**, the attacker will learn all users' passwords and be able to attack their accounts on other sites, assuming the user has re-used their password across sites (very likely)

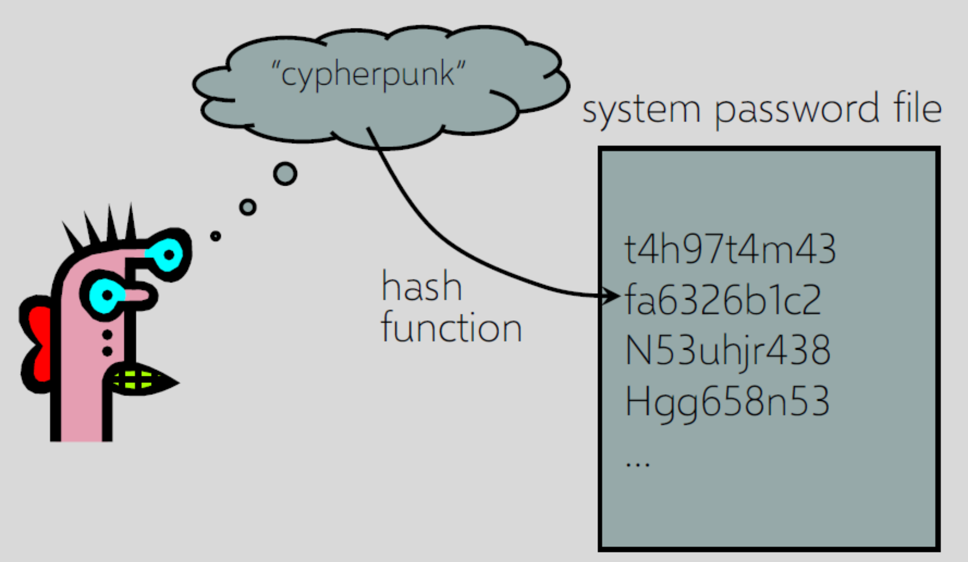
User table (plaintext)

Username	Password
alice	password
bob	hunter2
charlie	correct-battery-horse-staple
dakotah	hunter2

Hash the plaintext password

- **Important:** Hash the plaintext password, then store the hash in the database

- **Cryptographic hash function:**
  - One-way function:
    - Given y = H(M), hard to compute M
  - Deterministic:
    - H maps any message to a short digest (e.g., 256-bit string)
  - Collisions resistant:
    - Can’t find M, M’ s.t. H(M) = H(M’)



User table (Hashing)

Username	Password
alice	XohImNooBHFR0OVvjcYpJ3NgPQ1qq73WKhHvch0VQtg=
bob	9S+9MrKzuG/4jvbEkGKChfSCrxXdyylUH5S89Saj9sc=
charlie	0mk89QsPD4FIJQv8IcHnoSe6qjOzKvcNuTevydeUxWA=
dakotah	9S+9MrKzuG/4jvbEkGKChfSCrxXdyylUH5S89Saj9sc=

Problems with just hashing

- Users who have identical passwords are easy to spot

- Dictionary Attacks
  - SHA256 is quite fast to compute
  - Attacker can pre-compute H(word) for everyword in the dictionary – do this once offline, and build the Rainbow table.

Rainbow table: a precomputed table for reversing hash functions

Password salts

- **Goal:**
  - Prevent two users who use identical passwords from being revealed
  - Add entropy to weak passwords to make pre-computed lookup
  - attacks intractable

- **Solution:** A **salt** is fixed-length cryptographically-strong random value
  - No need to keep the salt secret; can be stored alongside the password
  - Concatenate the salt and the password before hashing it

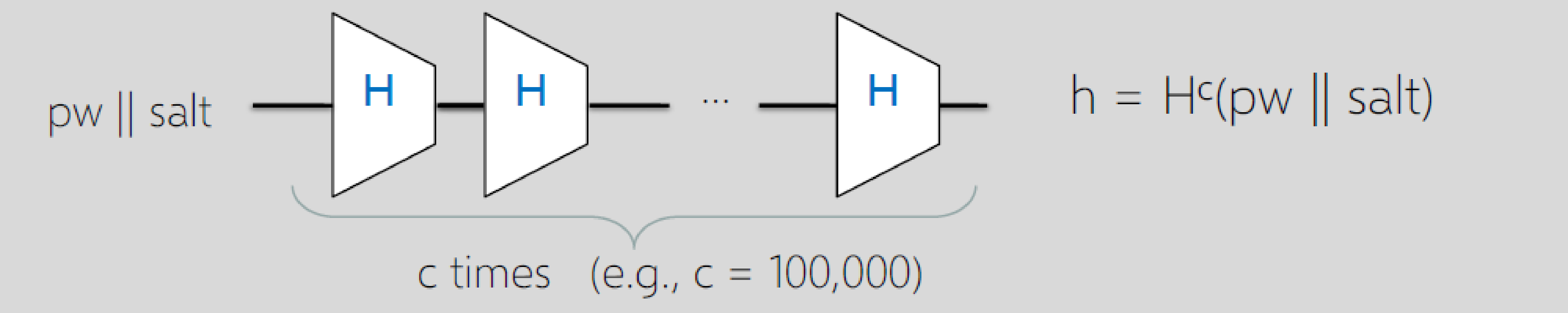
User table (Hashing with salt)

Username	Salt	Password
alice	ciMTj87Q5Ti/PDfSUM4jcAT6cFJWVwJFjEbMc2sqAn0=	AQAiFDIbEUk5Wdoe6tTL+bnCBOIsectOW2SfftG0je8=
bob	NB9zdy/OIVnGHkPK7fK01saCclpXrWV5rdtW8i5k/XY=	uxIXXvfrQ8/gTwrbTtgnsqsZCAw/y24O8nU3qlho5GE=
charlie	hetbWcTifseB9K3IQQPr6c/eMJyj3kVTqq/l+FqYf78=	FykuFcJV0AjBLyxMuQWrvuSTjRXyXStitVteWUJmPlM=
dakotah	IZu5hPamBS/QY4ILZzTcyVY8TK17Dt9hmXW7bC4XbCc=	ydVe+vA56bKbA0oXzRfYtkABUXaxgkF4ngB0xNJRvA4=

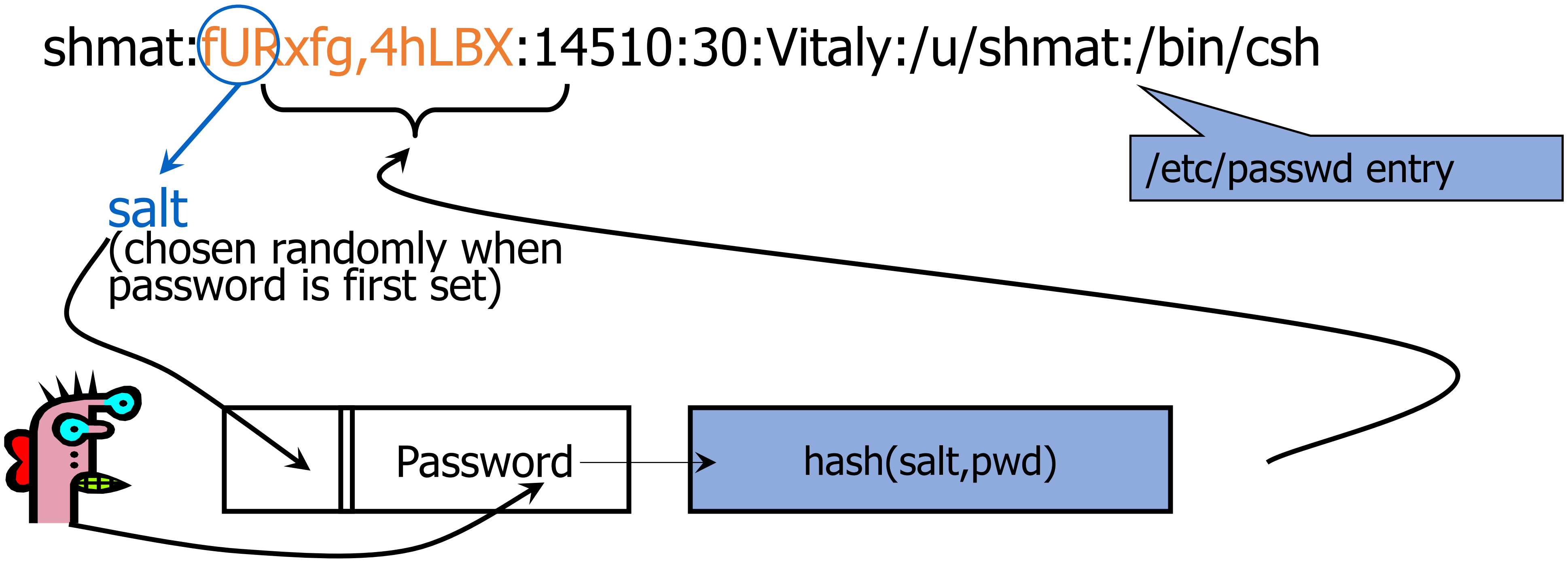
Making Attacking Harder

- Make hashing slower to slow down cracking attacks

- PKCS#5 approach:



- 1) iteration hashing
- 2) slower (Memory-hard) hash functions:: Scrypt and argon2



shmat:fURxfg,4hLBX:14510:30:Vitaly:/u/shmat:/bin/csh

/etc/passwd entry

salt
(chosen randomly when
password is first set)

Password

hash(salt,pwd)

- Users with the same password have <u>different</u> entries in the password file
- Offline dictionary attack becomes much harder

Attacks on Passwords

- **Online**
  - Try to guess passwords by logging to a live system
- **Offline**
  - Try to guess passwords in the (typically stolen) password database, or
  - Pre-computation can make offline attacks very fast

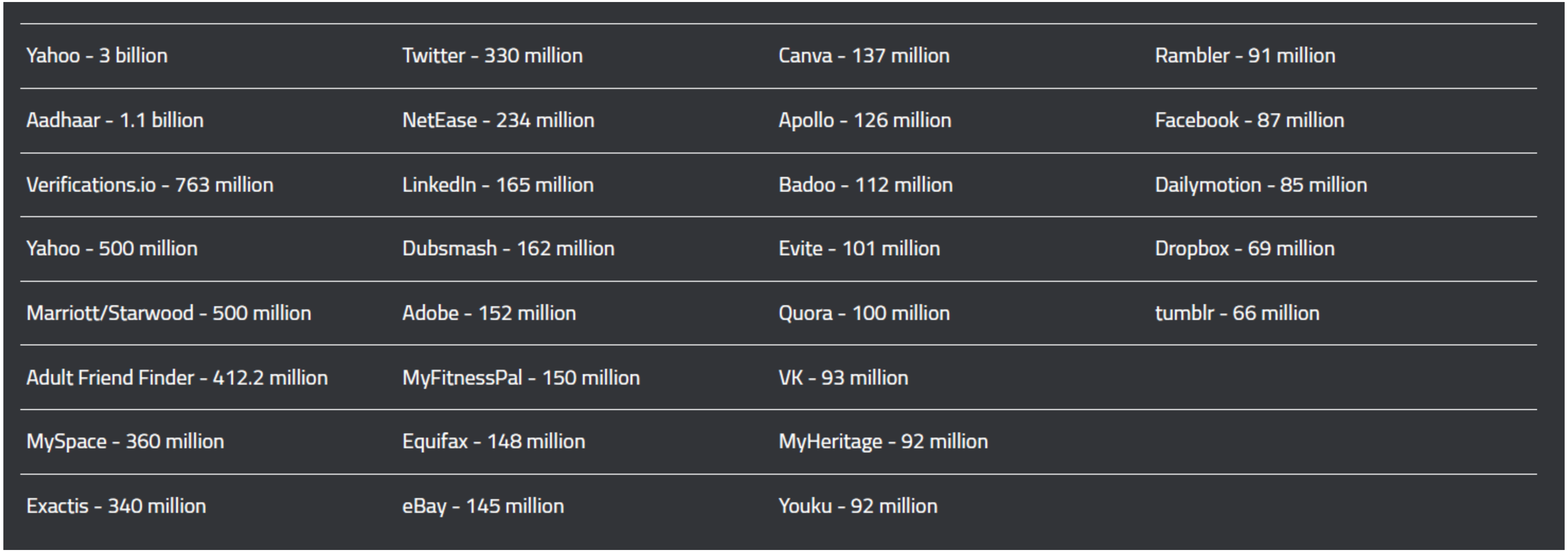
Online attack

- the number of guess attempts allowed is small

- But online attack is much more effective than what we thought since
  - people’s password choices vary much among each other.
  - Password is highly related to Personal information (birthday, information)
  - Etc.

Targeted Online Password Guessing: An Underestimated Threat. ACM CCS 16

Online attack: Biggest data breaches



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

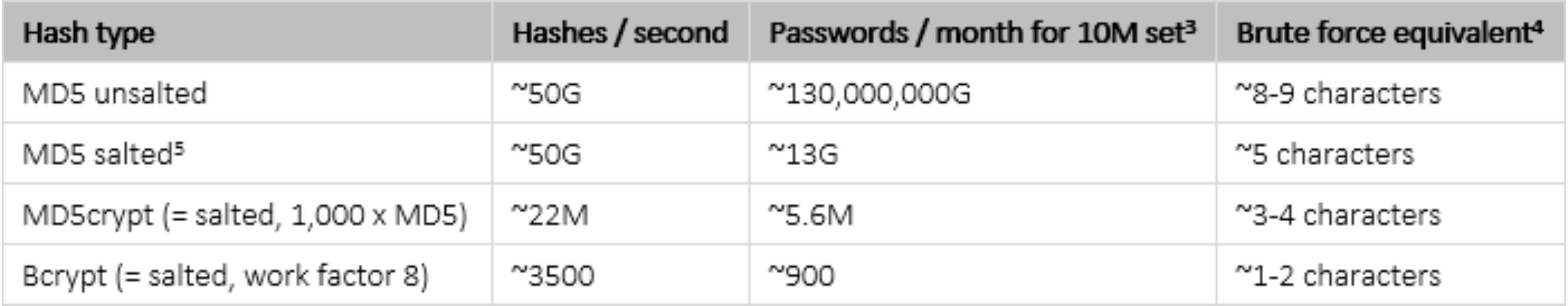
Targeted Online Password Guessing: An Underestimated Threat. ACM CCS 16

Were you in a breach?

- https://haveibeenpwned.com/

Offline attack

- Build Rainbow table



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

… with custom GPU and FPGA hardware

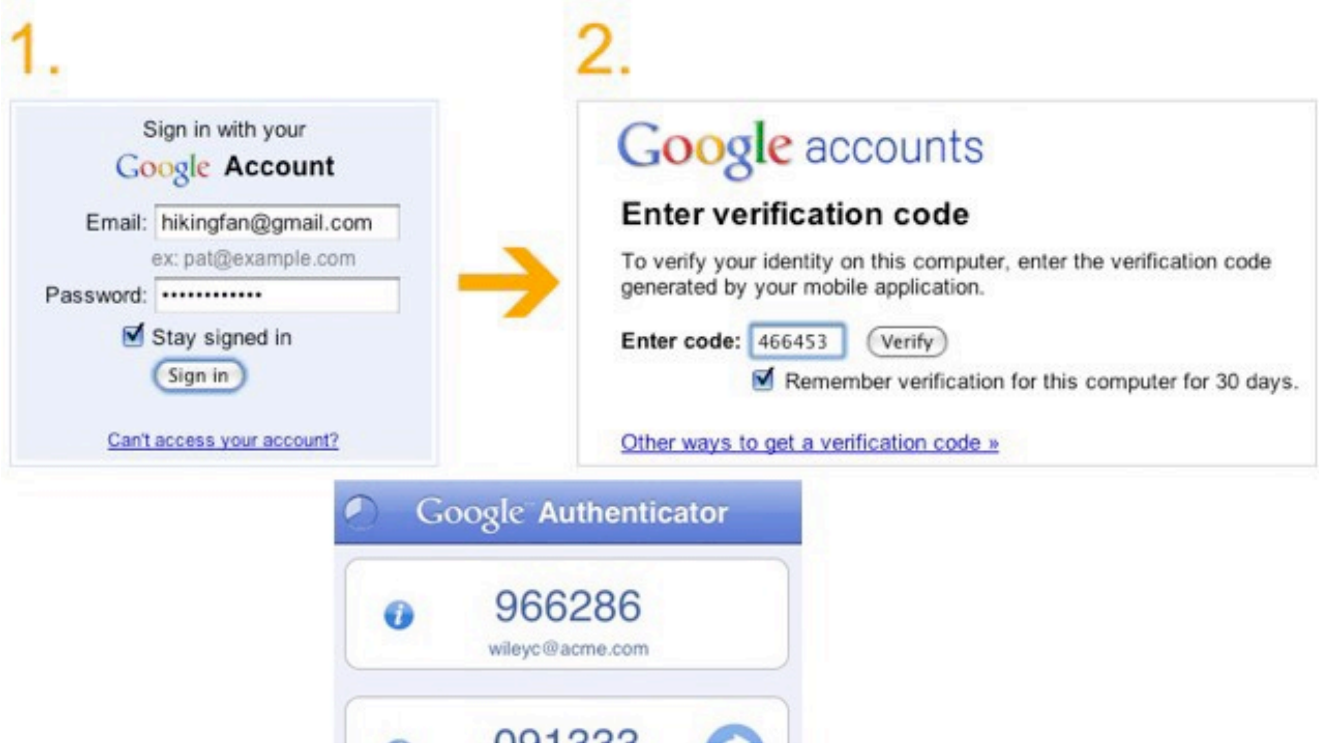
https://medium.com/@ScatteredSecrets/how-to-crack-billions-of-passwords-6773af298172

Multi forms of password authentication

- Single password authentication

- **Multi-Factor Authentication**

    - When you login google account
    - using an unusual equipment



1.

Sign in with your Google Account

Email: hikingfan@gmail.com

ex: pat@example.com

Password: ••••••••••••

☑ Stay signed in

2.

Google accounts

Enter verification code

To verify your identity on this computer, enter the verification code generated by your mobile application.

Enter code: 466453

☑ Remember verification for this computer for 30 days.

Other ways to get a verification code »

Google Authenticator

966286

wileyc@acme.com

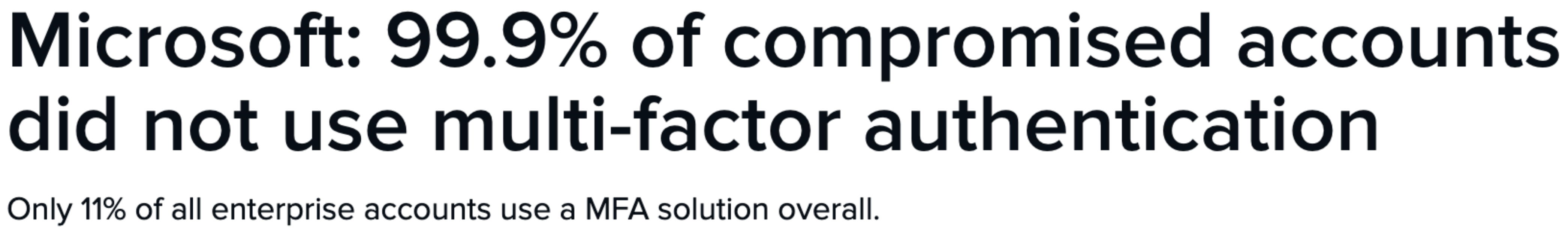
Factors for two factor authentication (2FA)

- Combine passwords with another way to authenticate user

- Second factor is usually proof of ownership of …

- Email address
  - Telephone number (via SMS)
  - Device (via authenticator app)
  - Hardware token (one-time-password token, universal second factor U2F token)

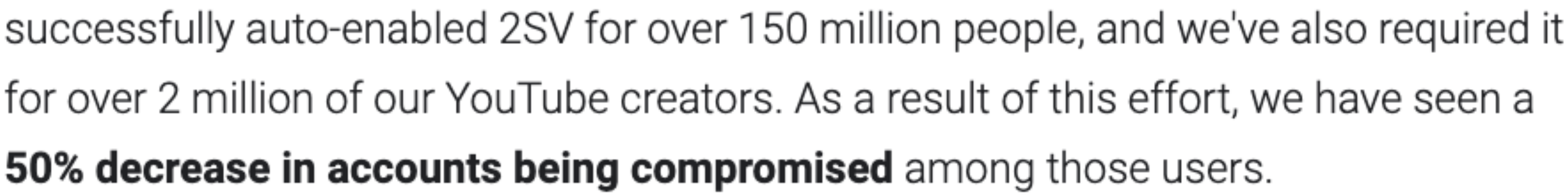
Effectiveness of 2FA



Microsoft: 99.9% of compromised accounts did not use multi-factor authentication

Only 11% of all enterprise accounts use a MFA solution overall.

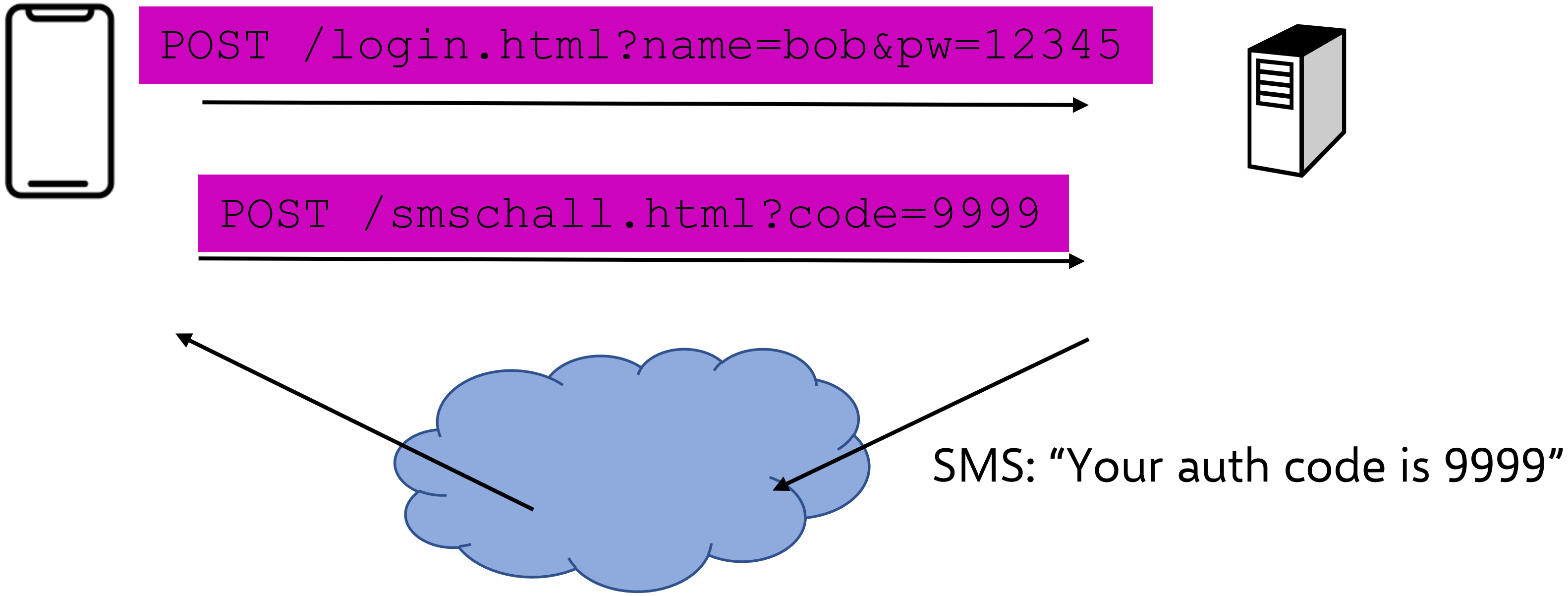
Microsoft report, Mar 2020



successfully auto-enabled 2SV for over 150 million people, and we've also required it for over 2 million of our YouTube creators. As a result of this effort, we have seen a **50% decrease in accounts being compromised** among those users.

Google report, Feb 2022

SMS (short message service) Authentication



```
POST /login.html?name=bob&pw=12345
```

```
POST /smschall.html?code=9999
```

SMS: “Your auth code is 9999”

Suppose you know someone’s password (e.g., due to breach) but their account is protected by SMS-based 2FA. What can you do as an attacker?

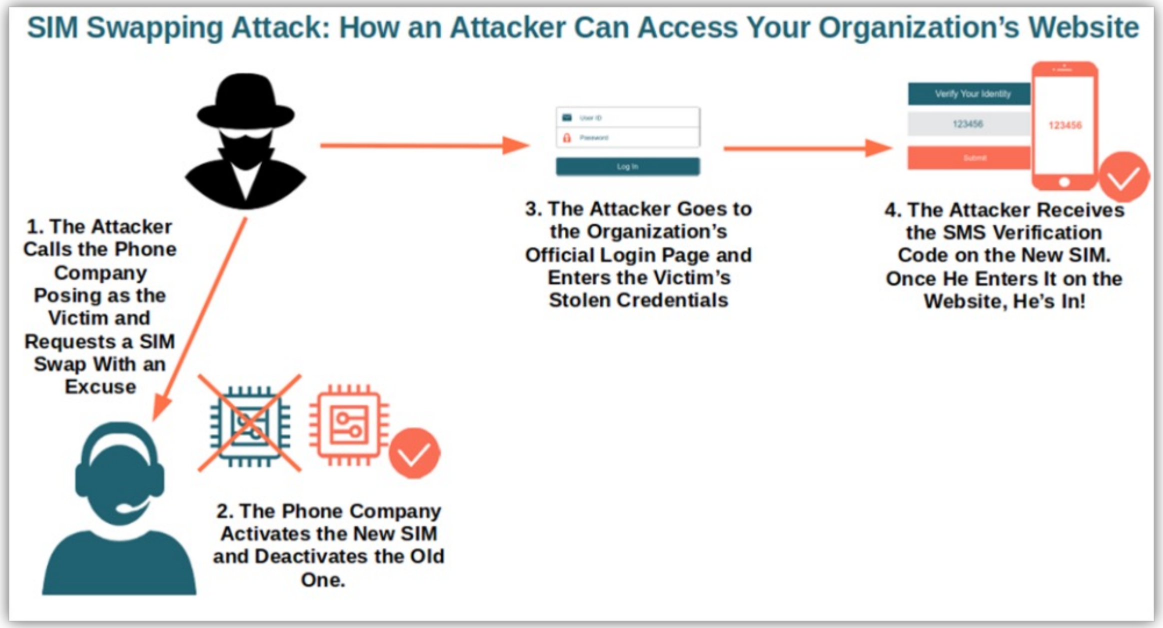
Circumventing SMS-Based 2FA

- Have physical access to device that receives SMS

- SIM swap: trick phone company into registering victim’s phone # to your device

- Phishing attacks: confuse or

  trick user into disclosing SMS to you



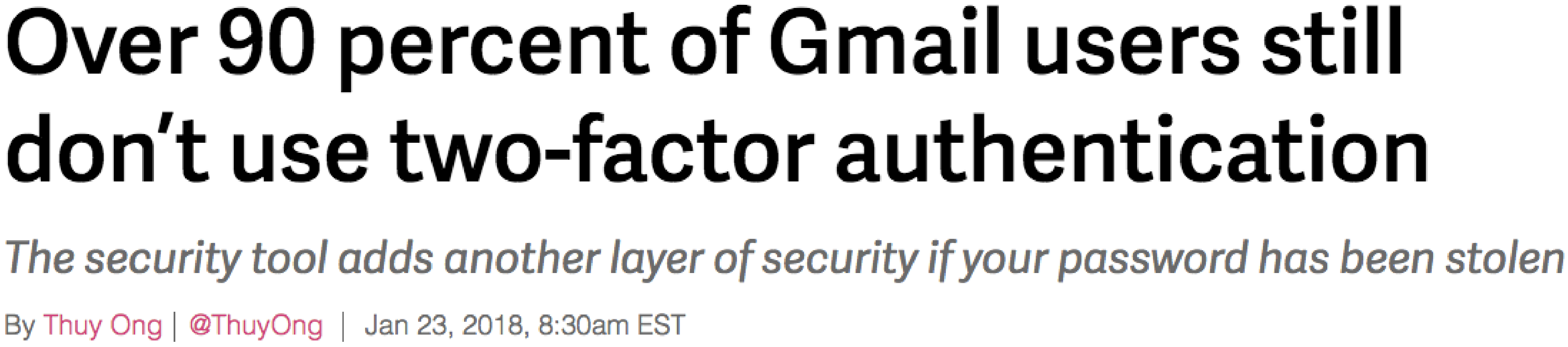
SIM Swapping Attack: How an Attacker Can Access Your Organization’s Website

1. The Attacker Calls the Phone Company Posing as the Victim and Requests a SIM Swap With an Excuse

2. The Phone Company Activates the New SIM and Deactivates the Old One.

3. The Attacker Goes to the Organization’s Official Login Page and Enters the Victim’s Stolen Credentials

4. The Attacker Receives the SMS Verification Code on the New SIM. Once He Enters It on the Website, He’s In!



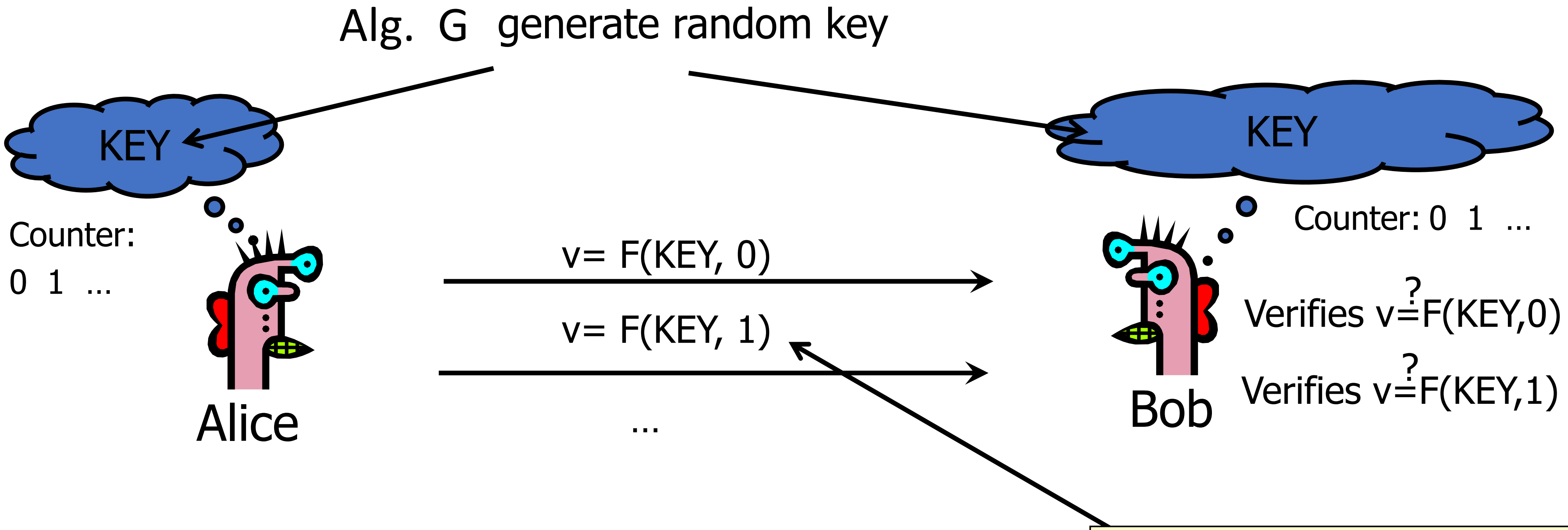
Over 90 percent of Gmail users still don’t use two-factor authentication

The security tool adds another layer of security if your password has been stolen

By Thuy Ong | @ThuyOng | Jan 23, 2018, 8:30am EST

Usability remains a key issue preventing adoption

Time-based One-Time Passwords



Alg. G generate random key

KEY

KEY

Counter:
0 1 …

Alice

$v= F(KEY, 0)$

$v= F(KEY, 1)$

…

Counter: 0 1 …

Bob

Verifies $v\overset{?}{=}F(KEY,0)$

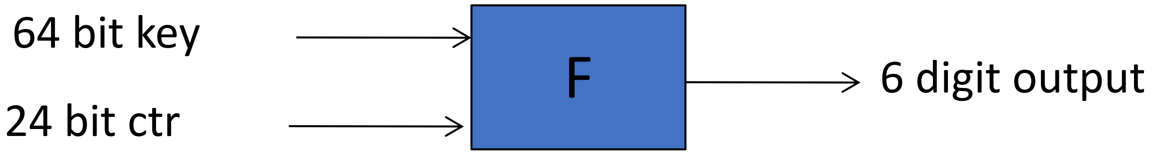
Verifies $v\overset{?}{=}F(KEY,1)$

RSA uses a custom function
Input: 64-bit key, 24-bit ctr
Output: 6-digit value

- Advancing the counter
  - Time-based (60 seconds) or every button press

- “Theorem”: if F is a secure PRF
  then protocol is secure against eavesdropping

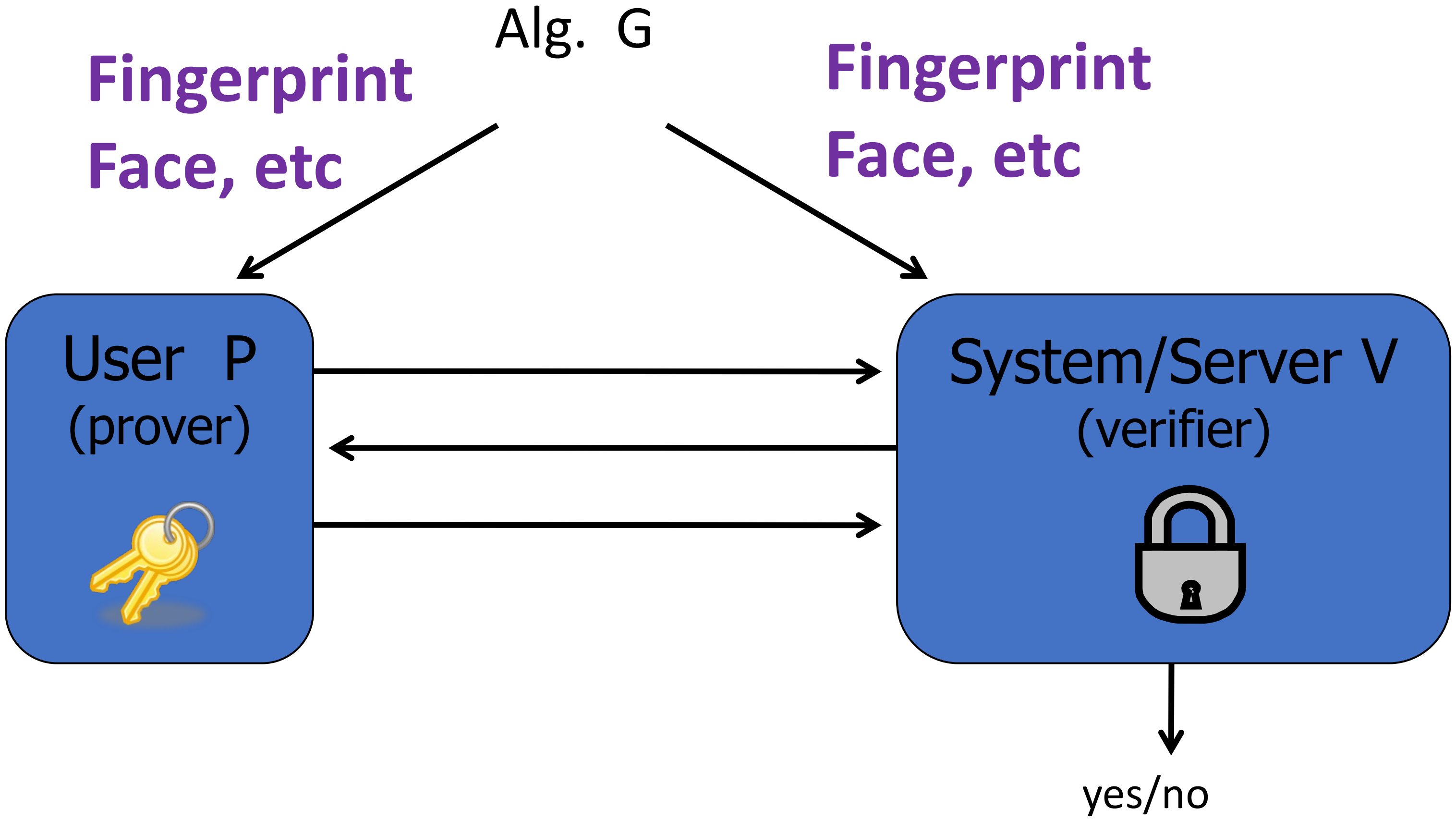
- RSA SecurID uses a custom PRF:



- Advancing state: sk ← (k, i+1)
  - Time based: every 60 seconds
  - User action: every button press
- Both systems allow for skew in the counter value

Biometric Authentication

What you are


Alg. G
Fingerprint
Face, etc
Fingerprint
Face, etc
User P
(prover)
System/Server V
(verifier)
yes/no

Biometric Error Rates

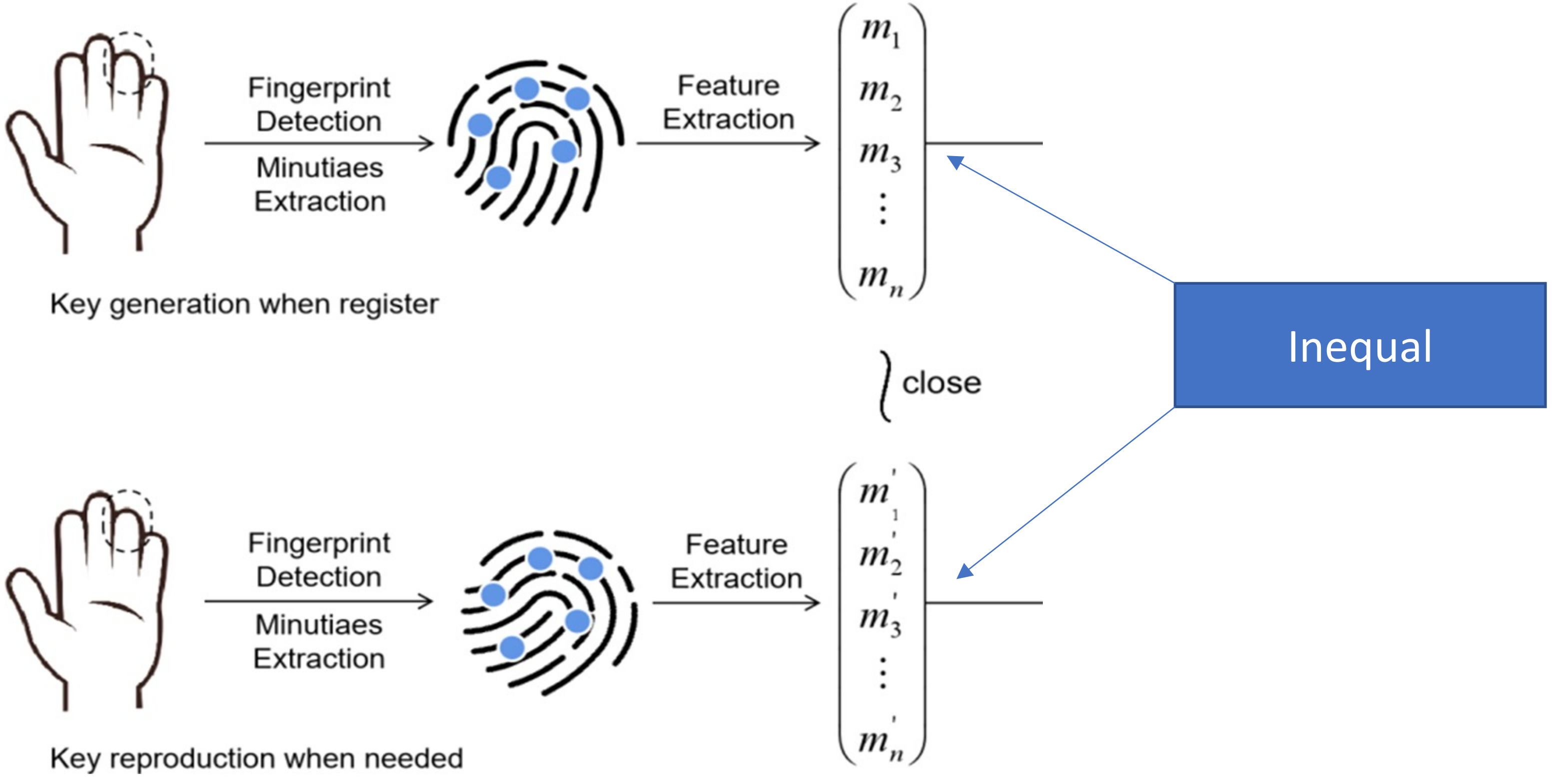
- “Fraud rate” vs. “insult rate”
  - Fraud = system accepts a forgery (false accept)
  - Insult = system rejects valid user (false reject)

- Increasing acceptance threshold increases fraud rate, decreases insult rate

- How to optimize both fraud rate and insult rate?

Biometric Error Rates

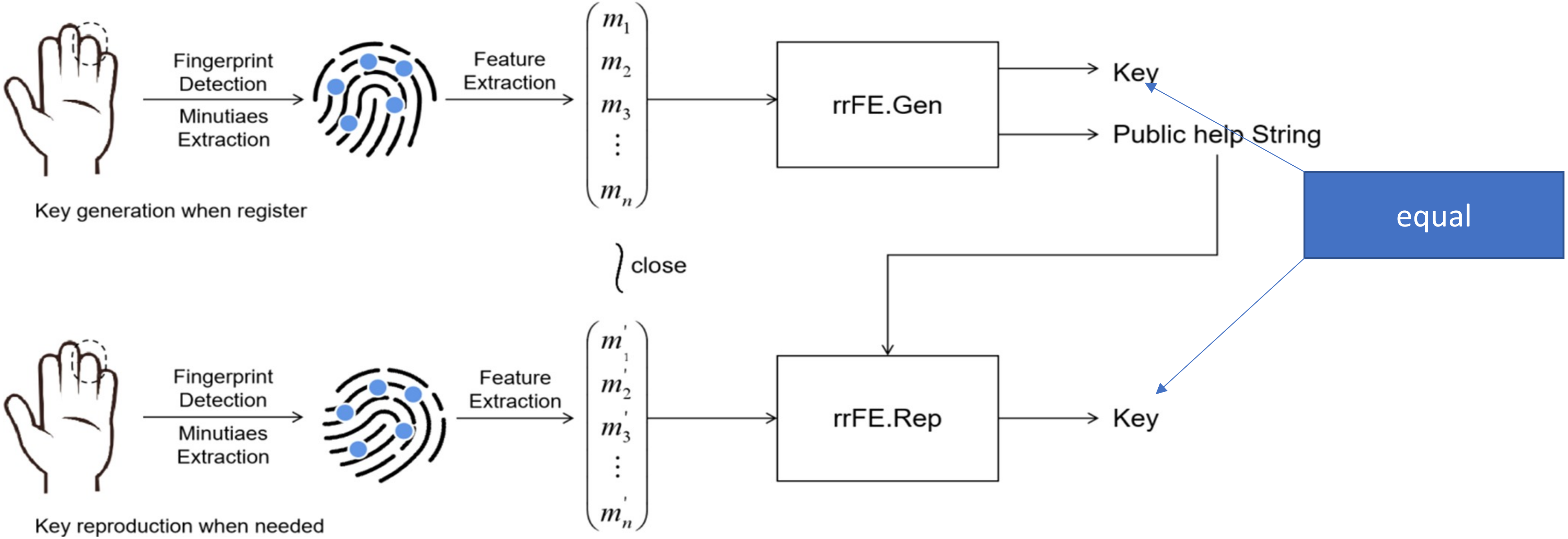
- Error Rate is mainly due to the instability of Bio-feature



Biometric Error Rates

- Design better Fuzzy extractor such that

$$FE(m) = FE(m') \qquad \text{even } m \neq m' \text{ but close to } m'$$



Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data, EUROCRYPT 2004

Pros and Cons

- Advantages:
  - Nothing to remember
  - Passive
  - Can’t share (generally)
- Problems
  - Private, but not secret: Sharing between multiple systems?
  - Revocation is difficult (impossible?): Please change a new password. Face??
  - Birthday paradox: With false accept rate of 1 in a million, probability of false match is above 50% with only 1609 samples

Biometric Birthday paradox

- With 23 people we have 253 pairs:

$$\frac{23 \cdot 22}{2} = 253$$

- The chance of 2 people having different birthdays is:

$$1 - \frac{1}{365} = \frac{364}{365} = .997260$$

- But making **253 comparisons** and having them *all* be different

$$\left(\frac{364}{365}\right)^{253} = .4995$$

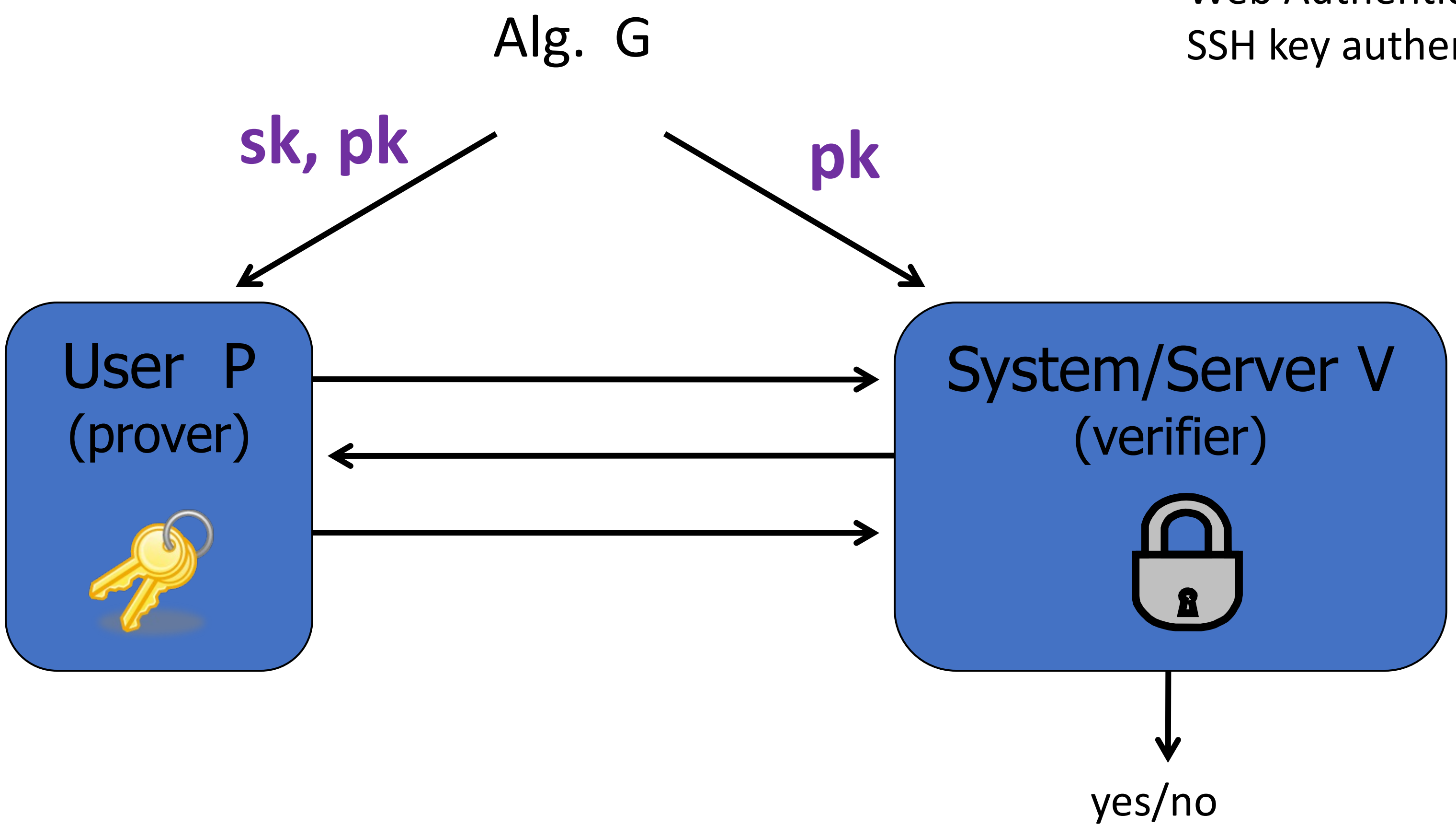
Biometric Authentication

- Primarily should be used as a second factor authentication

- Rather than a primary authentication factor

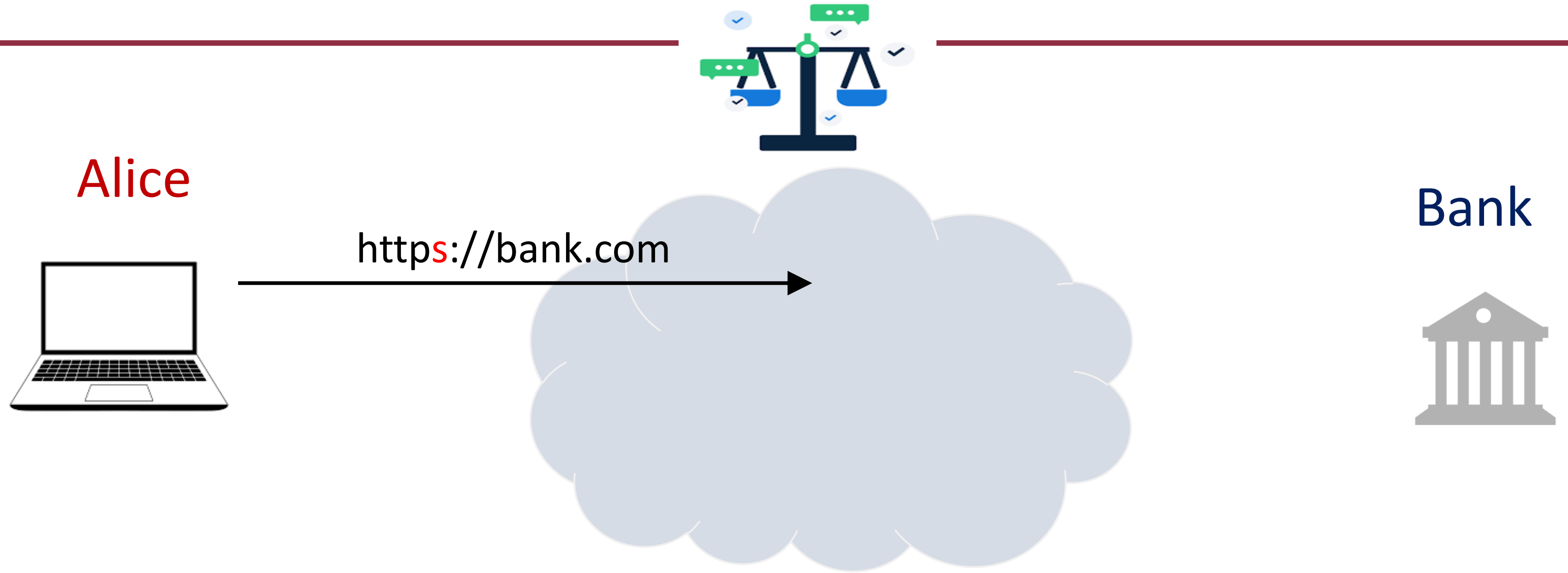
Public key Authentication

What you have



Example:
Web Authentication
SSH key authentication

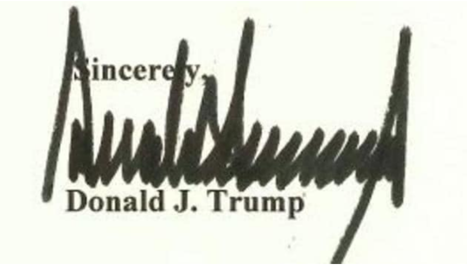
Web Authentication



- HTTPS (HTTP over SSL) refers to the combination of HTTP and SSL to implement secure communication

- The principal difference seen by a user is that URL addresses begin with https:// rather than http://.
  - A normal HTTP connection uses port 80.
  - If HTTPS is specified, port 443 is used, which invokes TLS/SSL.

Prepare: digital signature


Sincerely,
Donald J. Trump

unforgeable

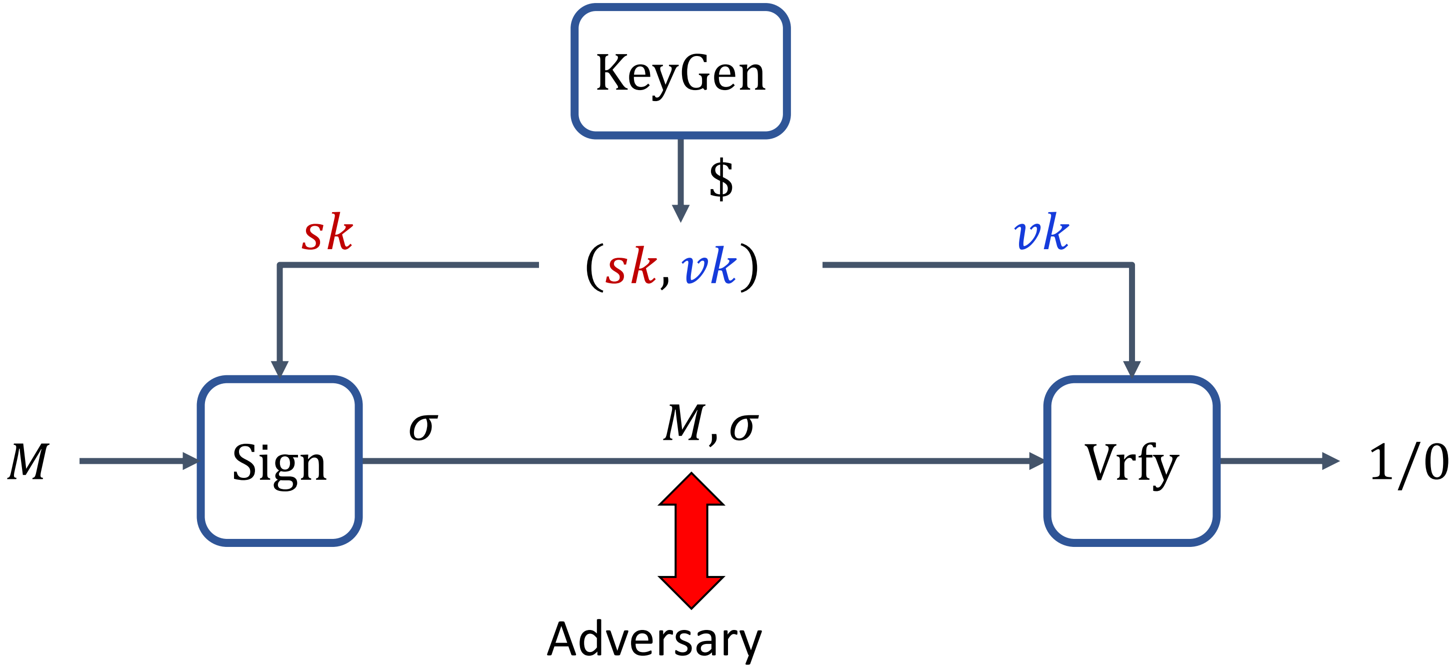
$\sigma_1, \sigma_2, \ldots$

unforgeable

σ^*

Digital Signature

- A **digital signature** is a mathematical scheme for verifying the authenticity of digital messages or documents. A valid digital signature on a message gives a recipient confidence that the message came from a sender known to the recipient
- It roughly consists of **secret key** sk and **verification public key** vk



Digital signature using in practice (we will see them later)

- RSA signature
    - RSAwithSHA-256,382,512 (PKCS #1 V2.1, RFC 6594)
- ECDSA signature
    - ECDSA256,384,512 (NIST FIPS 186-4)
    - EdDSA (RFC 6979)
- Schnorr signature

Trusted Third Party (TTP) Certification Authorities

- Digital Certification

vk_{ca}

TTP

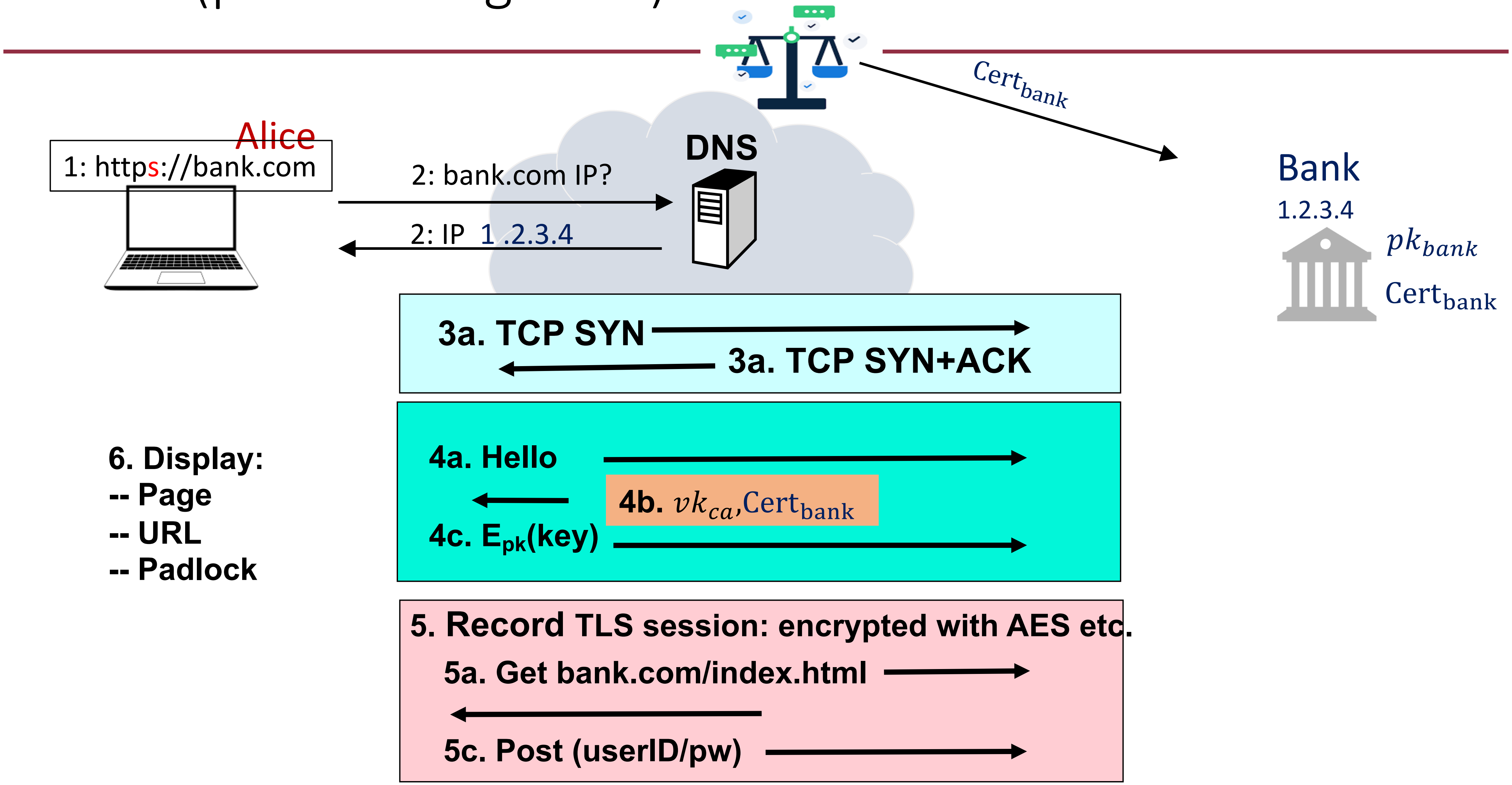
sk_{ca}

$$\mathrm{Cert}_{bank} = \mathrm{Sign}(\mathrm{sk}_{ca}, \mathrm{Bank's\ public\ (sign)\ key\ is\ } vk_{bank};\ \ \mathrm{URL\ ishttps://www.hangseng.com/}\ )$$

Any one with vk_{ca} can verify the Cert_{bank}

- Ex: Digicert, Apple, Google, Amazon etc.

HTTPS (put it all together)



Alice

1: https://bank.com

2: bank.com IP?

2: IP 1 .2.3.4

DNS

Cert_{bank}

Bank

1.2.3.4

pk_{bank}

Cert_{bank}

3a. TCP SYN

3a. TCP SYN+ACK

4a. Hello

4b. vk_{ca},Cert_{bank}

4c. E_{pk}(key)

5. Record TLS session: encrypted with AES etc.

5a. Get bank.com/index.html

5c. Post (userID/pw)

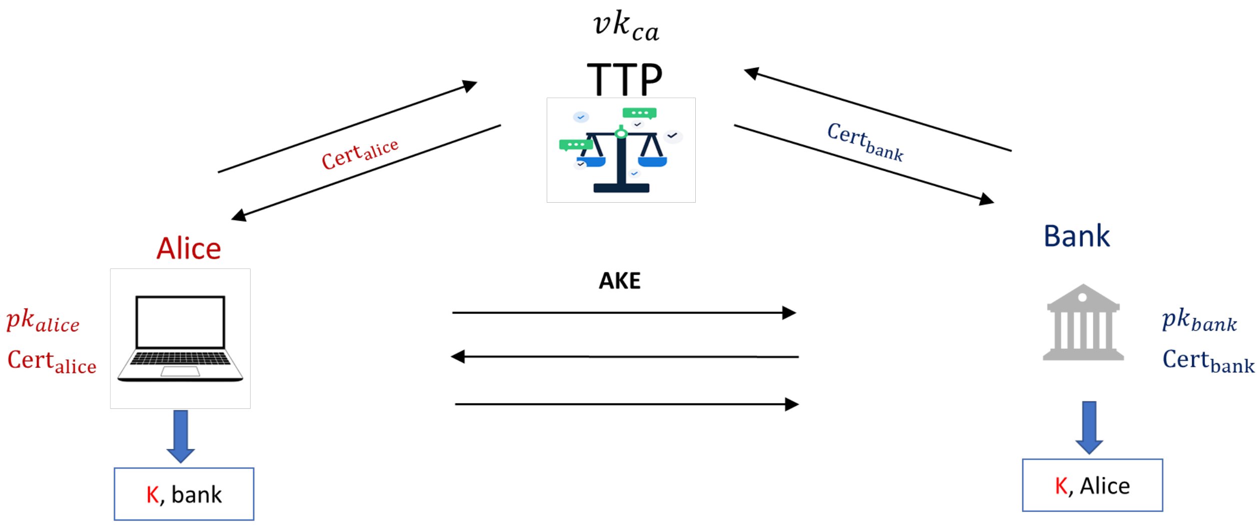
6. Display:
-- Page
-- URL
-- Padlock

Problem: public key infrastructure (PKI)

- A single TTP

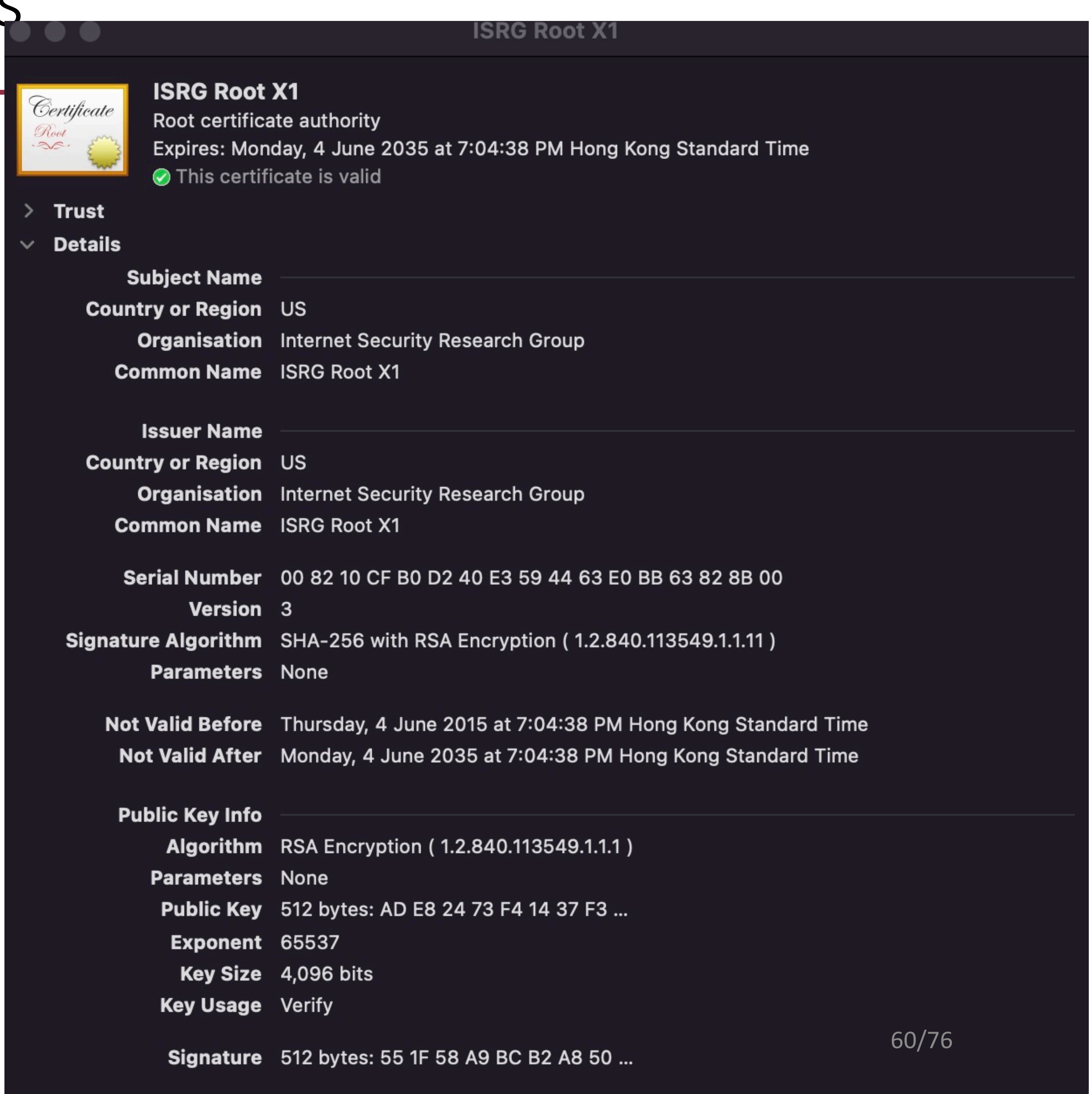
- Single point of failure
  - What if TTP is corrupted?

- How should we deploy the trust of certification?
  - How does Bank communicate with TTP to get $Cert_{bank}$?



TTP: Certification Authorities

- Subject Name
  - Who’s CA
- Issuer Name
  - Who gives this CA
  - Sign name
  - Valid
- PK information
  - pk
  - What is the pk is used
  - Key size



ISRG Root X1

ISRG Root X1
Root certificate authority
Expires: Monday, 4 June 2035 at 7:04:38 PM Hong Kong Standard Time
✓ This certificate is valid

> Trust

⌄ Details

Subject Name	
Country or Region	US
Organisation	Internet Security Research Group
Common Name	ISRG Root X1
Issuer Name	
Country or Region	US
Organisation	Internet Security Research Group
Common Name	ISRG Root X1
Serial Number	00 82 10 CF B0 D2 40 E3 59 44 63 E0 BB 63 82 8B 00
Version	3
Signature Algorithm	SHA-256 with RSA Encryption (1.2.840.113549.1.1.11)
Parameters	None
Not Valid Before	Thursday, 4 June 2015 at 7:04:38 PM Hong Kong Standard Time
Not Valid After	Monday, 4 June 2035 at 7:04:38 PM Hong Kong Standard Time
Public Key Info	
Algorithm	RSA Encryption (1.2.840.113549.1.1.1)
Parameters	None
Public Key	512 bytes: AD E8 24 73 F4 14 37 F3 ...
Exponent	65537
Key Size	4,096 bits
Key Usage	Verify
Signature	512 bytes: 55 1F 58 A9 BC B2 A8 50 ...

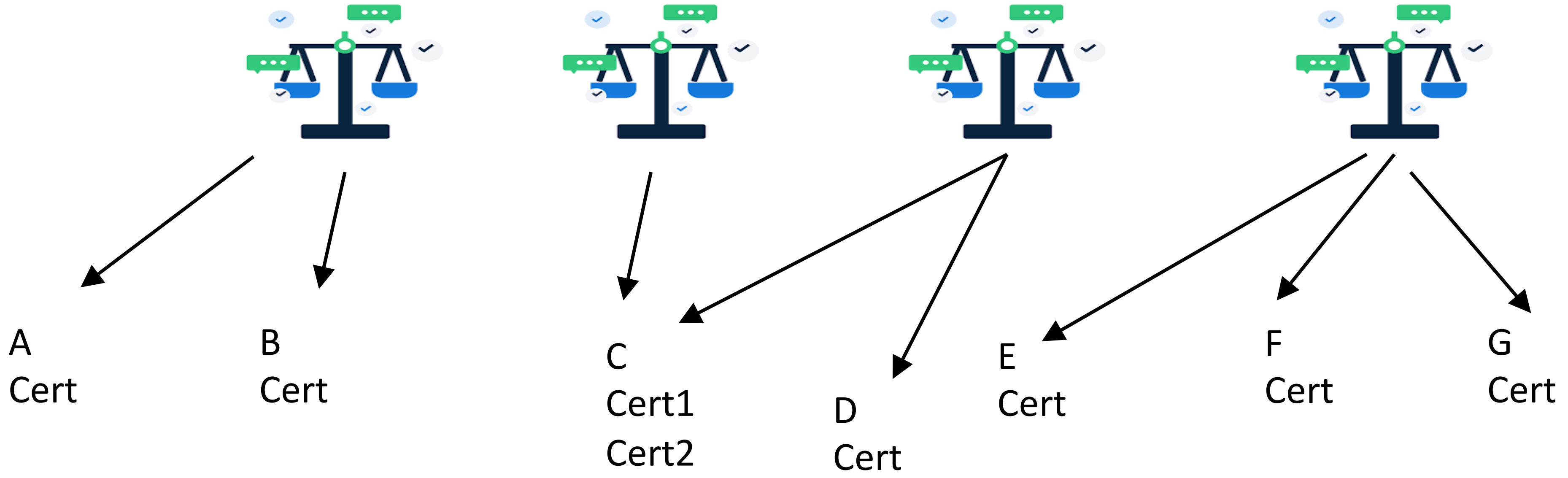
Certification Authorities(CA)

vk_{ca}

TTP

- How should I get the vk_{ca} of TTP?
- a root CA’s public key is provided together with the browser/System

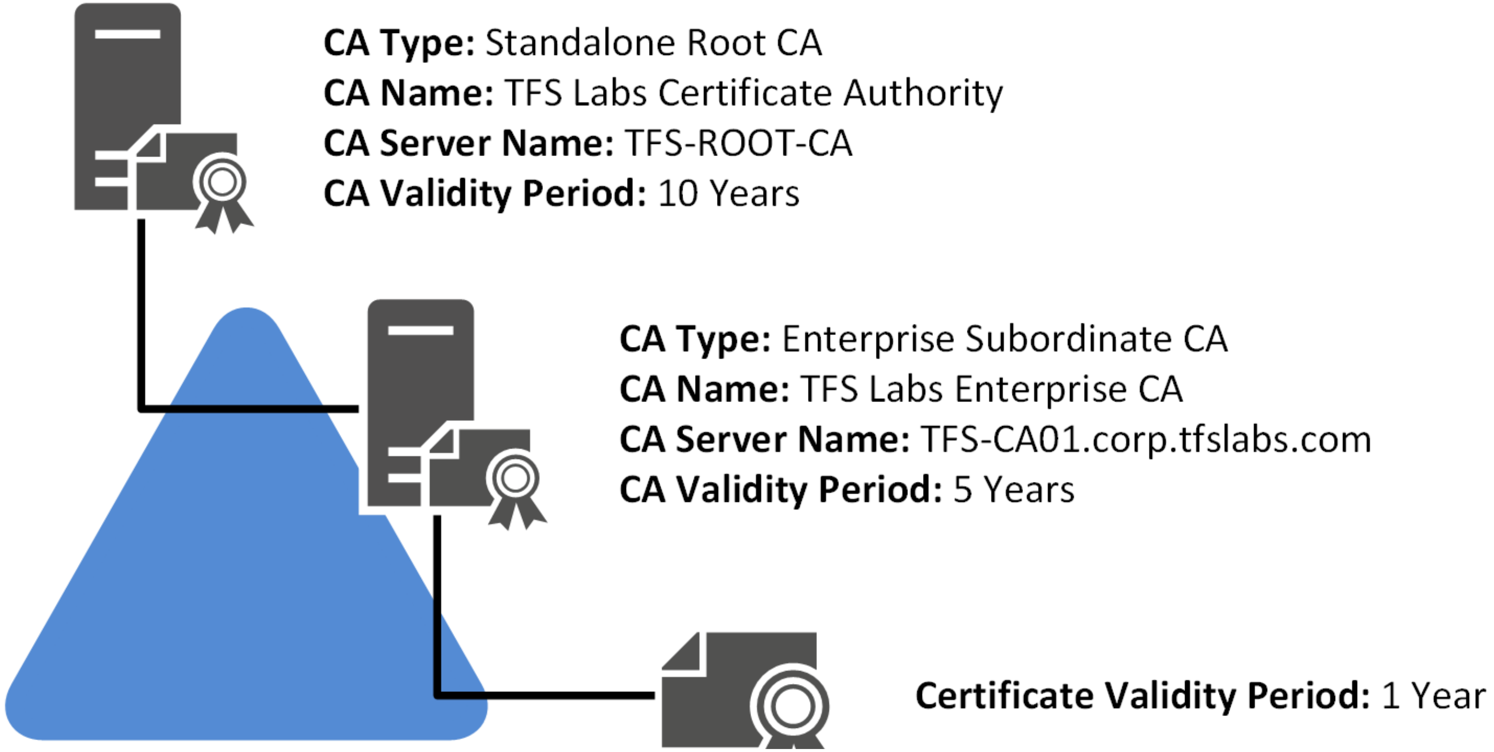
Multiple CAs



- Reduce the risk of single point of failure

Authentication Chain

We could build the trust of certificate chains from a single Root CA

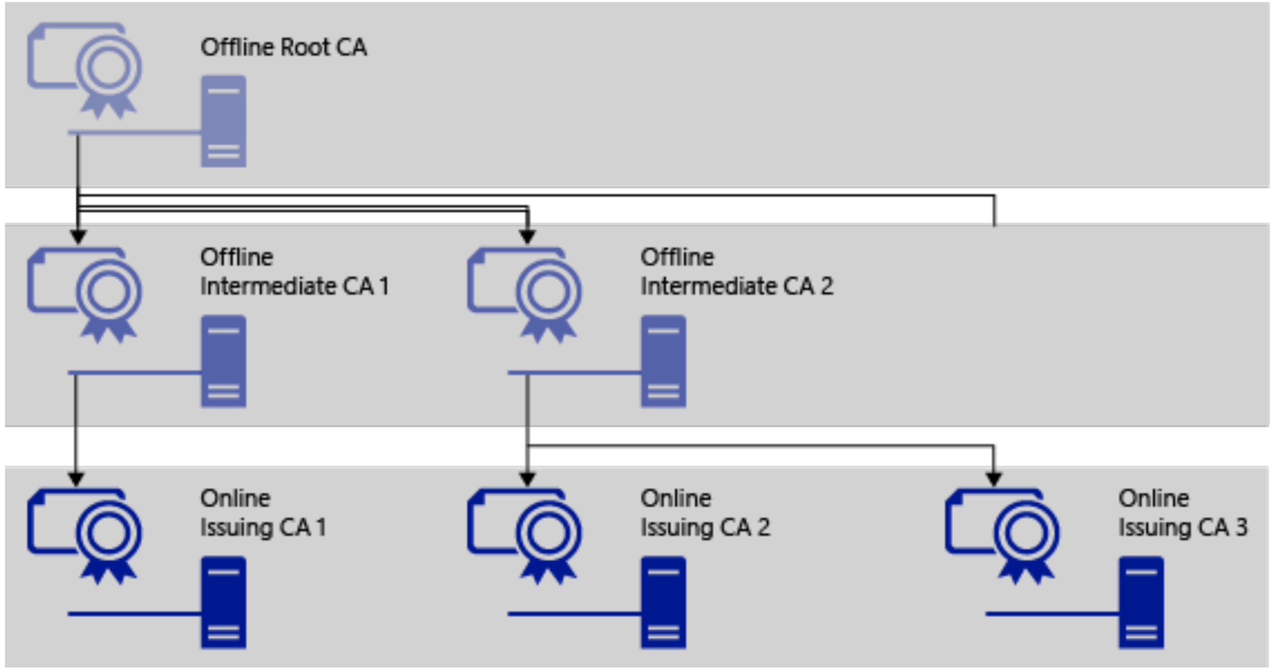


CA Type: Standalone Root CA
CA Name: TFS Labs Certificate Authority
CA Server Name: TFS-ROOT-CA
CA Validity Period: 10 Years

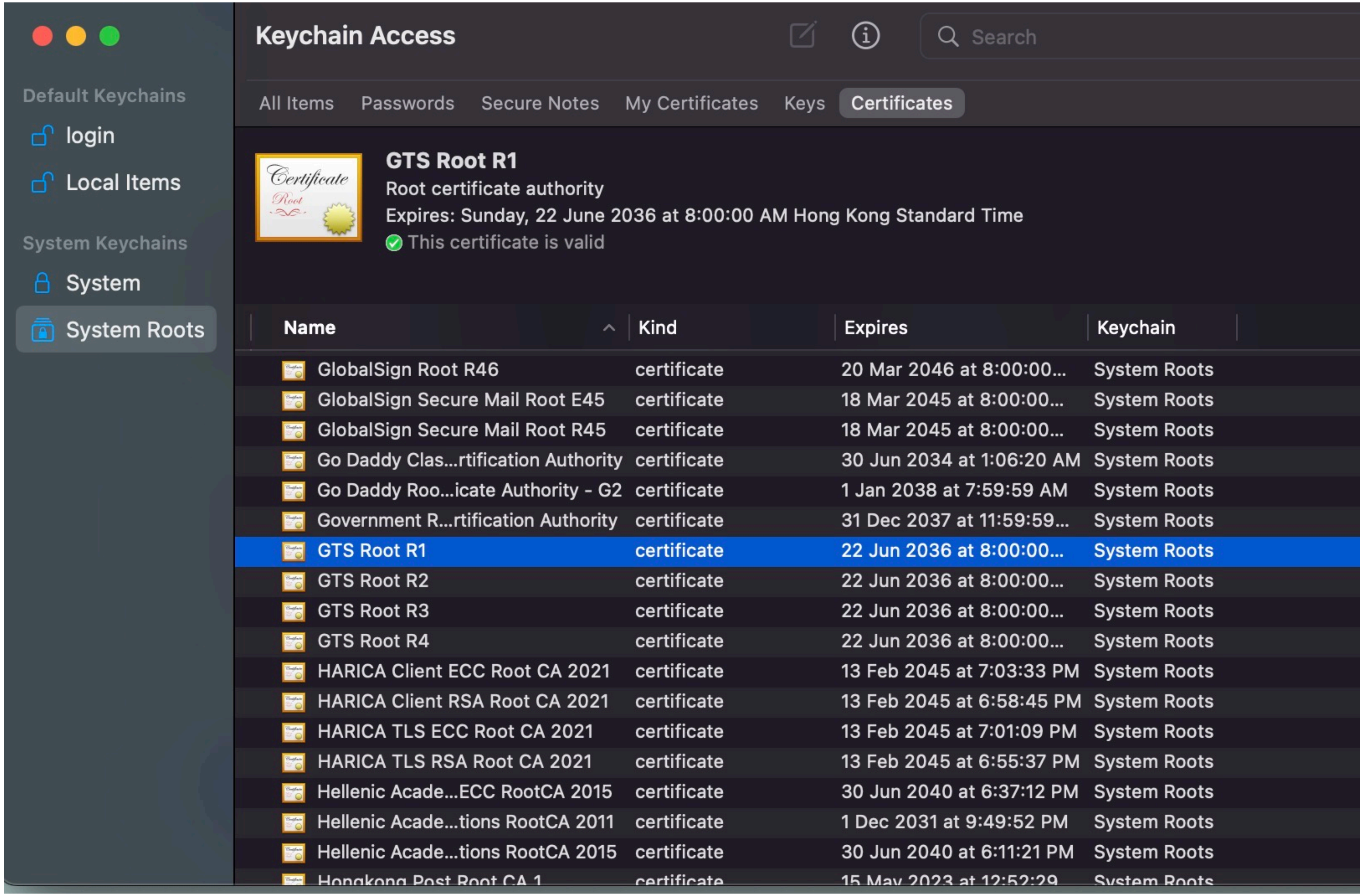
CA Type: Enterprise Subordinate CA
CA Name: TFS Labs Enterprise CA
CA Server Name: TFS-CA01.corp.tfslabs.com
CA Validity Period: 5 Years

Certificate Validity Period: 1 Year

Authentication Chain



Root CA in Mac OS



GTS Root R1

Root certificate authority

Expires: Sunday, 22 June 2036 at 8:00:00 AM Hong Kong Standard Time

✓ This certificate is valid

Name	Kind	Expires	Keychain
GlobalSign Root R46	certificate	20 Mar 2046 at 8:00:00...	System Roots
GlobalSign Secure Mail Root E45	certificate	18 Mar 2045 at 8:00:00...	System Roots
GlobalSign Secure Mail Root R45	certificate	18 Mar 2045 at 8:00:00...	System Roots
Go Daddy Clas...rtification Authority	certificate	30 Jun 2034 at 1:06:20 AM	System Roots
Go Daddy Roo...icate Authority - G2	certificate	1 Jan 2038 at 7:59:59 AM	System Roots
Government R...rtification Authority	certificate	31 Dec 2037 at 11:59:59...	System Roots
GTS Root R1	certificate	22 Jun 2036 at 8:00:00...	System Roots
GTS Root R2	certificate	22 Jun 2036 at 8:00:00...	System Roots
GTS Root R3	certificate	22 Jun 2036 at 8:00:00...	System Roots
GTS Root R4	certificate	22 Jun 2036 at 8:00:00...	System Roots
HARICA Client ECC Root CA 2021	certificate	13 Feb 2045 at 7:03:33 PM	System Roots
HARICA Client RSA Root CA 2021	certificate	13 Feb 2045 at 6:58:45 PM	System Roots
HARICA TLS ECC Root CA 2021	certificate	13 Feb 2045 at 7:01:09 PM	System Roots
HARICA TLS RSA Root CA 2021	certificate	13 Feb 2045 at 6:55:37 PM	System Roots
Hellenic Acade...ECC RootCA 2015	certificate	30 Jun 2040 at 6:37:12 PM	System Roots
Hellenic Acade...tions RootCA 2011	certificate	1 Dec 2031 at 9:49:52 PM	System Roots
Hellenic Acade...tions RootCA 2015	certificate	30 Jun 2040 at 6:11:21 PM	System Roots
Hongkong Post Root CA 1	certificate	15 May 2023 at 12:52:29	System Roots

Root CA in Windows

- Root CA in windows
  - Select Run from the Start menu, and then enter certlm.msc. The Certificate Manager tool for the local device appears.

Root CA in web browser

- chrome://settings/security

- Firefox

SSH Authentication

- Web authentication is a kind of public key authentication

- SSH is another one

    - SSL was originally designed to protect HTTP traffic carried between web browsers and web servers

    - SSH (Secure Shall) was originally designed to protect remote login sessions

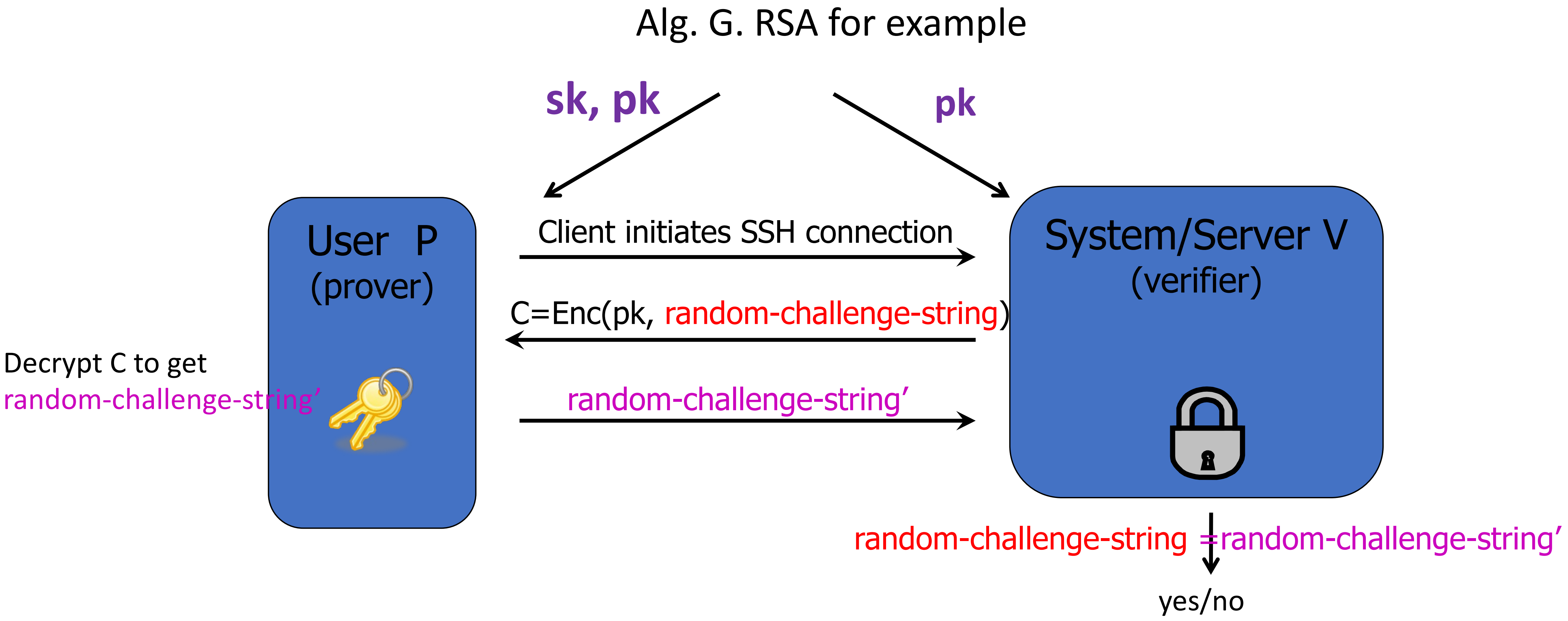
SSH Authentication

- SSH Authentication does not aim to establish a shared secret key (as key exchange does)

- It was designed to protect remote login sessions
- No Public key infrastructure is required

- Client generates the public/secret key locally
- Upload public key to server and store secret key on the device

SSH Public Key Authentication simplified



Alg. G. RSA for example

sk, pk

pk

User P (prover)

System/Server V (verifier)

Client initiates SSH connection

C=Enc(pk, random-challenge-string)

Decrypt C to get random-challenge-string’

random-challenge-string’

random-challenge-string =random-challenge-string’

yes/no

Pros of SSH key authentication

- SSH keys are more difficult to hack than passwords and thus are more secure.

- SSH keys aren’t human generated, so you’ll avoid having easy-to-guess keys like “123456” or “password”.

- Unlike passwords, your private SSH key isn’t sent to the server.

Disadvantages of SSH key authentication

- the private key needs to be stored on the device

- distribution of public keys and education of staff on how to use SSH keys can be more cumbersome.

SSH

- https://www.ssh.com/academy/ssh
- RFC 4251
- RFC 4252

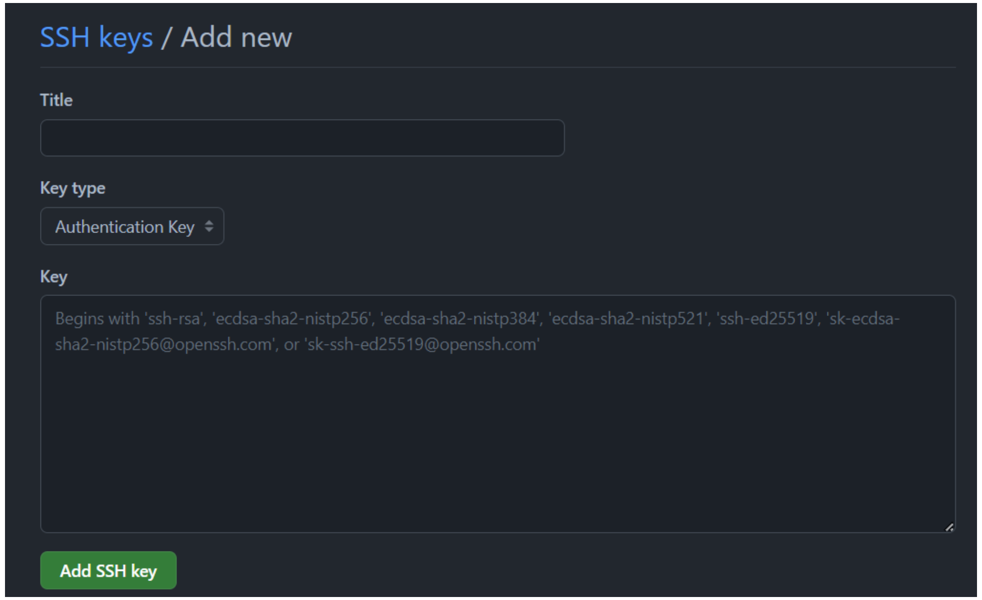
- Demo when remote login Github
- https://docs.github.com/authentication

Example

- Use SSH key to login Github
- https://docs.github.com/authentication

Demo

- My private repository Problems-in-FoC
    - git clone git@github.com:haiyangxc/Problems-in-FoC.git
- Create new RSA public secret keys
    - ssh-keygen -t rsa -b 4096
- Add public key in Github
    - To github and client
- Testing your SSH connection
    - ssh –T git@github.com
    - git clone git@github.com:haiyangxc/Problems-in-FoC.git



SSH keys / Add new

Title

Key type

Authentication Key

Key

Begins with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-ecdsa-sha2-nistp256@openssh.com', or 'sk-ssh-ed25519@openssh.com'

Thank you