CS 88: Security and Privacy

14: Authentication

03-19-2023

slides courtesy Christo Wilson, Vitaly Shmatikov

SWARTHMORE COLLEGE

Reading Quiz


Microsoft Windows xp
To begin, click your user name
Administrator
Type your password
Turn off computer
After you log on, you can add or change accounts.
Just go to Control Panel and click User Accounts.

Authentication

- Authentication is the process of verifying an actor's identity
- Critical for security of systems
  - Permissions, capabilities, and access control are all contingent upon knowing the identity of the actor
- Typically parameterized as a username and a secret
  - The secret attempts to limit unauthorized access
- Desirable properties of secrets include being *unforgeable*, *unguessable*, and *revocable*

Types of Secrets

- Actors provide their secret to log-in to a system
- Three classes of secrets:
    1. Something you know
        - Example: a password
    2. Something you are
        - Examples: fingerprint, voice scan, iris scan
    3. Somewhere you are
        - IP address, geolocation
    4. Something you have
        - Examples: a smart card or smart phone, 2FA

Today’s Topics

- Password Storage
  - How should you securely store password to prevent cracking?
  - Are there ways to help detect password breaches?
- Password Cracking
  - Basic attacks: brute forcing and dictionary
  - Hash chains
  - Rainbow tables
- Local and Distributed Authentication Systems
  - Unix/Linux PAM system
  - NIS
  - Needham-Schroeder
  - Kerberos

Attacker Goals and Threat Model

- Assume we have a system storing usernames and passwords
- The attacker has access to the password database/file
- Our goal: even if the database is stolen, attacker should learn as little as possible about the passwords.

Database

User	Password
cbw	p4ssW0rd
sandi	puppies
amislove	3spr3ss0

Cracked Passwords

User	Password
cbw	p4ssW0rd
sandi	puppies
amislove	3spr3ss0

Password Storage Summary

1. Never store passwords in plain text
2. Always salt and hash passwords before storing them
3. Use hash functions with a high work factor

- These rules apply to any system that needs to authenticate users
    - Operating systems, websites, etc.

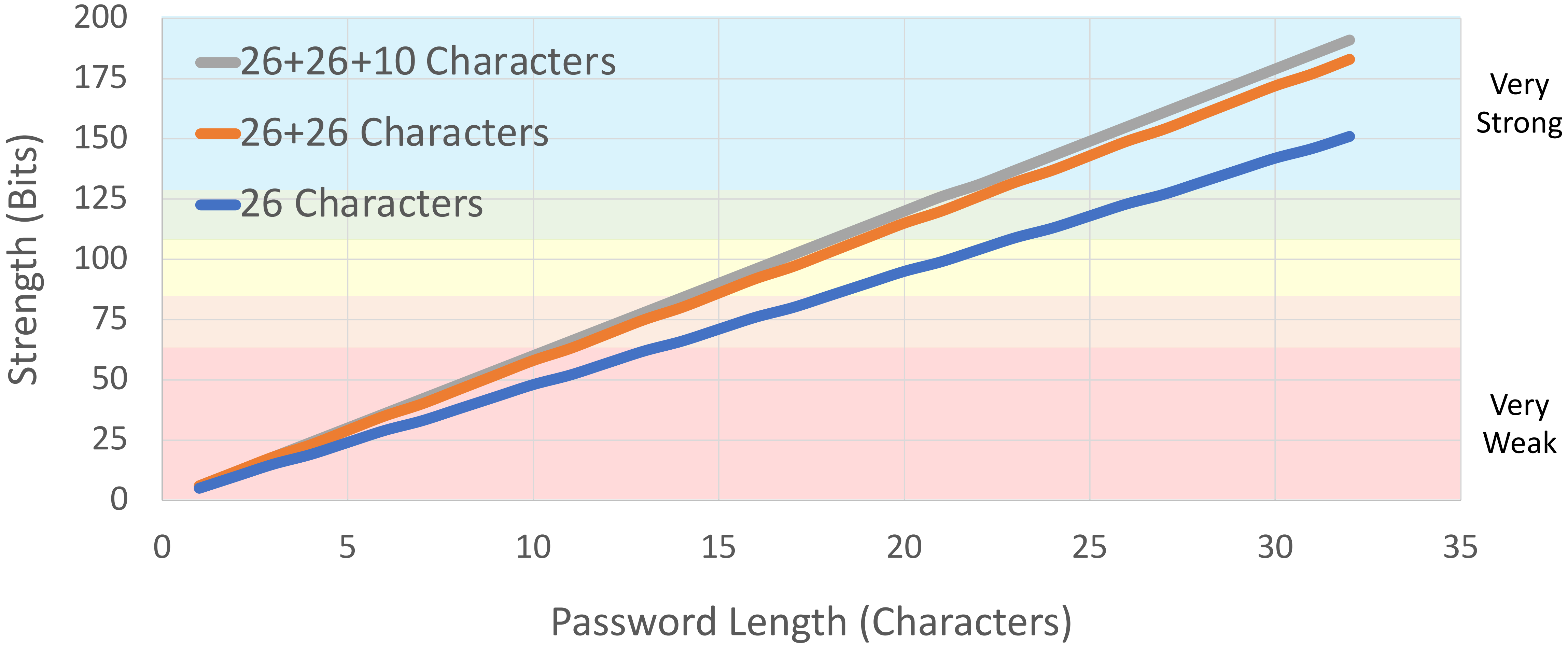
Password Quality

$$S = \log_2 N^L \rightarrow L = \frac{S}{\log_2 N}$$

- How do we measure password quality? Entropy
  - *N* – the number of possible symbols (e.g. lowercase, uppercase, numbers, etc.)
  - *L* – the length of the password
  - *S* – the strength of the password, in bits
- Formula tells you length *L* needed to achieve a desired strength S…
  - … for **randomly generated** passwords
- Is this a realistic measure in practice?

The Strength of Random Passwords

$$S = L * log_2 N$$



Password Cracking

Password Theory

Hash Chains

Rainbow Tables

Uncompressed Hash Chain Example

	Link 1		Link 2		Link 3	
p'	**H(p’) = h’**	**R(h’) = p”**	**H(p”) = h”**	**R(h”) = p’’’**	**H(p’’’) = h’’’**	**R(h’’’) = p***
abcde	\\WPNP_	vlsfqp	_QOZLR	eusrqv	CMRQ5X	cjldar
passw	VZDGEF	gfnxsk	ZLGEKV	yookol	EBOTHT	zvxscs
12345	SM-QK\9	sawtzg	RHKP_D	gvmdwm	BYE4LB	wjizbn
secrt	OKFTaY	btweoz	WA15HK	ttgovl	Q_4\6ZB	eivlqc

K = 3

Only these two columns get stored on disk

Rainbow Tables

Rainbow tables improve on hash chains by reducing the likelihood of collisions

Key idea: instead of using a single reduction R, use a family of reductions $\{R_1, R_2, \ldots, R_k\}$

- Usage of H is the same as for hash chains
- A collisions can only occur between two chains if it happens at the same position (e.g. R_i in both chains)

Final Thoughts on Rainbow Tables

Caveats

- Tables must be built for each hash function and character set
- Salting and key stretching defeat rainbow tables

Rainbow tables are effective in some cases, e.g. MD5 and NTLM

- Precomputed tables can be bought or downloaded for free

Password Management

Password Reuse

People have difficulty remembering >4 passwords

- Thus, people tend to reuse passwords across services
- What happens if any one of these services is compromised?

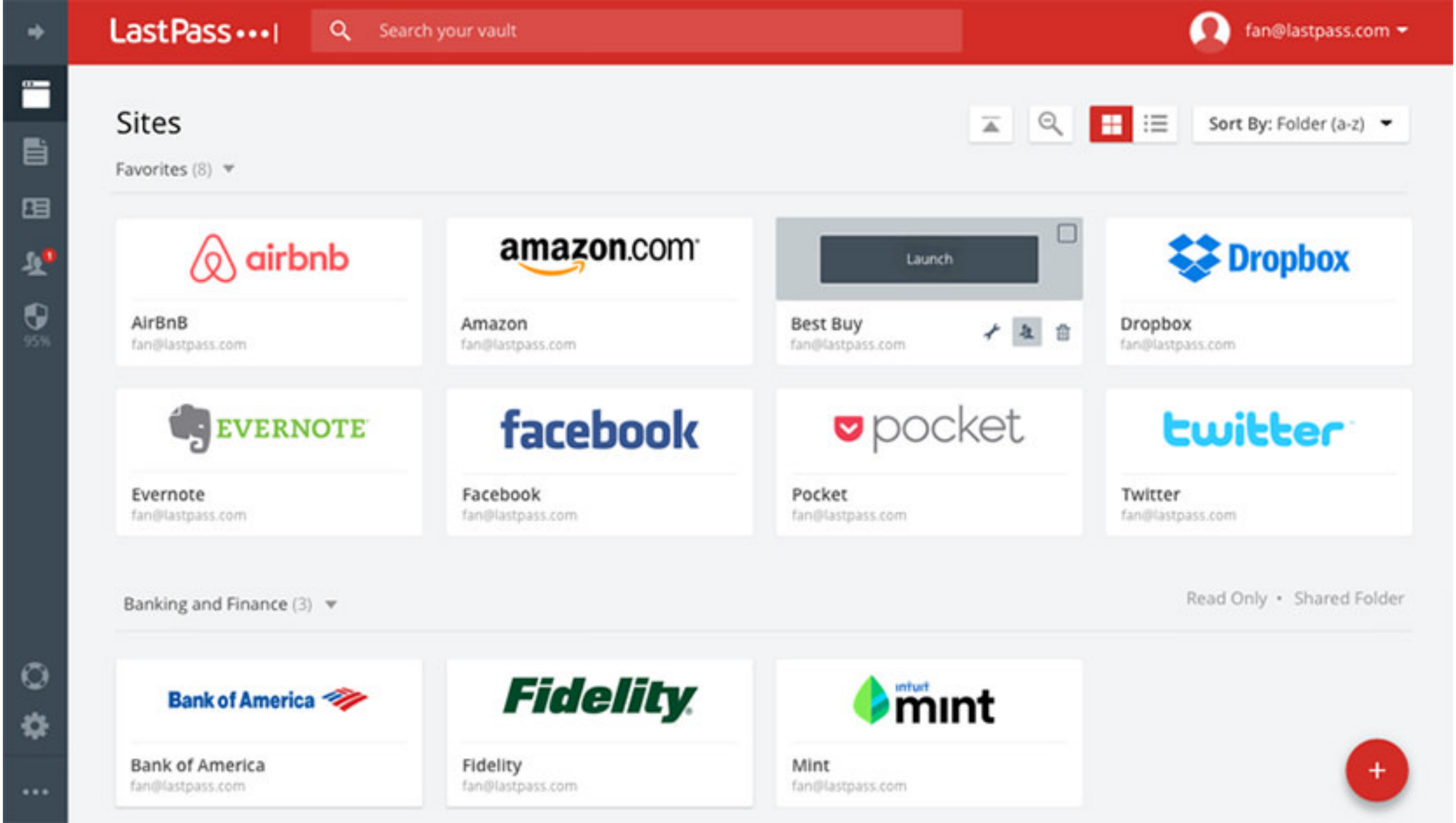
Service-specific passwords are a beneficial form of compartmentalization

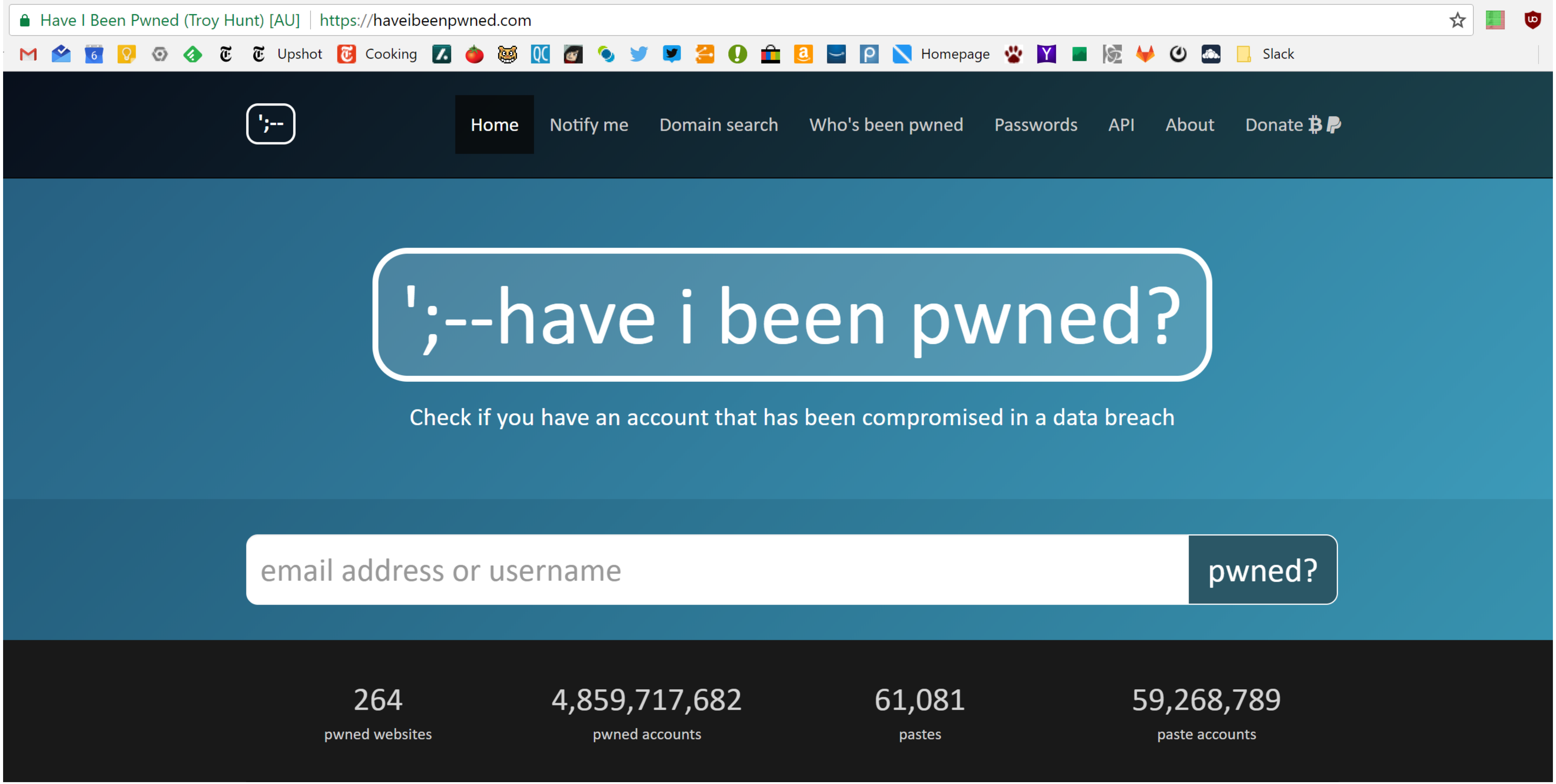
- Limits the damage when one service is inevitably breaches

Use a password manager

Some service providers now check for password reuse

- Forbid users from selecting passwords that have appeared in leaks


LastPass
Search your vault
fan@lastpass.com
Sites
Sort By: Folder (a-z)
Favorites (8)
AirBnB
fan@lastpass.com
Amazon
fan@lastpass.com
Launch
Best Buy
fan@lastpass.com
Dropbox
fan@lastpass.com
Evernote
fan@lastpass.com
Facebook
fan@lastpass.com
Pocket
fan@lastpass.com
Twitter
fan@lastpass.com
Banking and Finance (3)
Read Only • Shared Folder
Bank of America
fan@lastpass.com
Fidelity
fan@lastpass.com
Mint
fan@lastpass.com



Have I Been Pwned (Troy Hunt) [AU] | https://haveibeenpwned.com
Upshot
Cooking
Homepage
Slack
';--
Home
Notify me
Domain search
Who's been pwned
Passwords
API
About
Donate
';--have i been pwned?
Check if you have an account that has been compromised in a data breach
email address or username
pwned?
264
pwned websites
4,859,717,682
pwned accounts
61,081
pastes
59,268,789
paste accounts

Biometric Two Factor Authentication

Biometrics

SMS

Authentication Codes

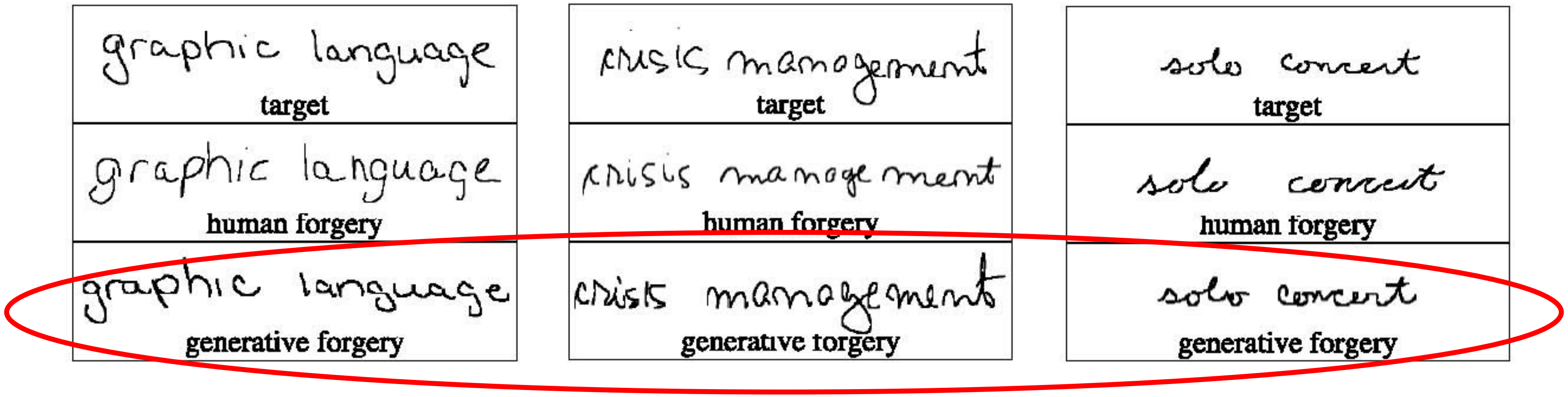
Smartcards & Hardware Tokens

Identification vs. Authentication

- Goal: associate an identity with an event
  - Example: a fingerprint at a crime scene
  - Key question: given a particular biometric reading, does there exist another person who has the same value of this biometric?
- Goal: verify a claimed identity
  - Example: fingerprint scanner to enter a building
  - Key question: do there exist any two persons who have the same value of this biometric?
    - Birthday paradox!

Forging Handwriting

[Ballard, Monrose, Lopresti]



Generated by computer algorithm trained on handwriting samples

Biometrics


I AM MORDAC, THE PREVENTER OF INFOR-MATION SERVICES, AND I BRING YOU MY NEWEST BIOMETRIC SCANNER.
scottadams@aol.com
www.dilbert.com
INSTEAD OF A PASS-WORD, I PUT THIS ON YOUR HEAD AND SQUEEZE UNTIL YOU SCREAM IN A WAY THAT ONLY YOU CAN SCREAM.
© 2007 Scott Adams, Inc./Dist. by UFS, Inc.
NO, THAT'S NOT YOU.
GAAA! GAAA! GAAA!

Fundamental Issue With Biometrics

Biometrics are immutable

- You are the password, and you can't change
- Unless you plan on undergoing plastic surgery?

Once compromised, there is no reset

- Passwords and tokens can be changed

Example: the Office of Personnel Management (OPM) breach

- US gov agency responsible for background checks
- Had fingerprint records of all people with security clearance
- Breached by China in 2015, all records stolen :(

Play-Doh Fingers

[Schuckers]

- Alternative to gelatin
- Play-Doh fingers fool 90% of fingerprint scanners
  - Clarkson University study
- Suggested perspiration measurement to test “liveness” of the finger

Token-based Two Factor Authentication

Types of Secrets

Actors provide their secret to log-in to a system

Three classes of secrets:

1. Something you know
   - Example: a password
2. Something you are
   - Examples: fingerprint, voice scan, iris scan
3. Something you have
   - Examples: a smart card or smart phone

Something You Have

Two-factor authentication has become more commonplace

Possible second factors:

- SMS passcodes
- Time-based one time passwords
- Hardware tokens

SMS Two Factor

Relies on your phone number as the second factor

- Key assumption: only your phone should receive SMS sent to your number

SMS two factor is deprecated. Why?

Social engineering the phone company

1. Call and pretend to be the victim
2. Say “I got a new SIM, please activate it”
3. If successful, phone calls and SMS are now sent to your SIM in your phone, instead of the victim

Not hypothetical: successfully used against many victims

One Time Passwords

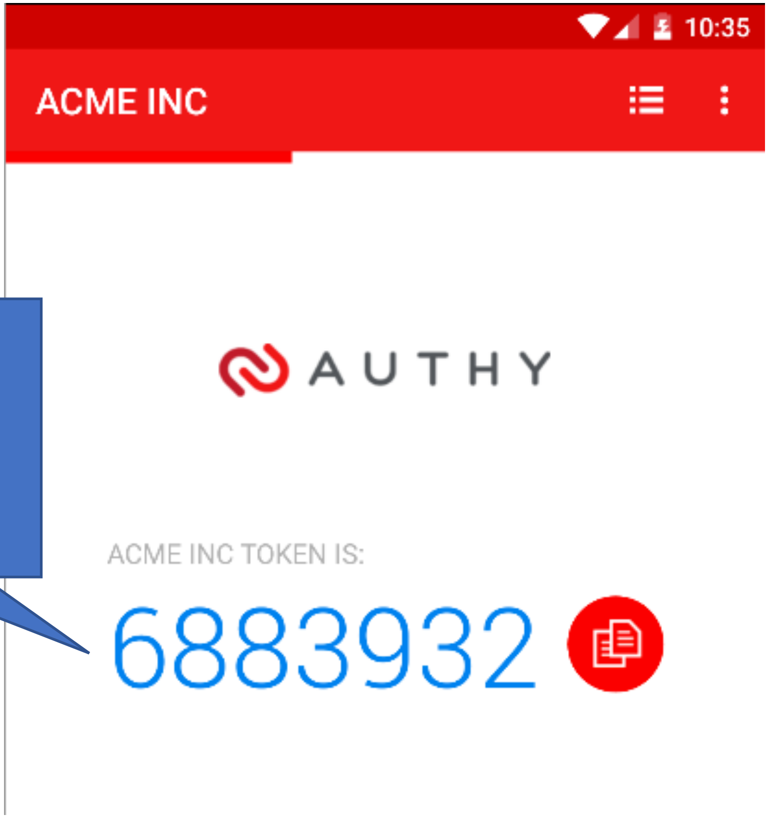
Generate ephemeral passcodes that change over time

To login, supply normal password and the current one time password

Relies on a shared secret between your mobile device and the service provider

- Shared secret allows both parties to know the current one time password

Changes every few minutes



Duo Mobile

Lastpass Authenticator

Google Authenticator

Time-based One-time Password Algorithm

T0 = <the beginning of time, typically Thursday, 1 January 1970 UTC>

TI = <length of time the password should be valid>

K = <shared secret key>

d = <the desired number of digits in the password>

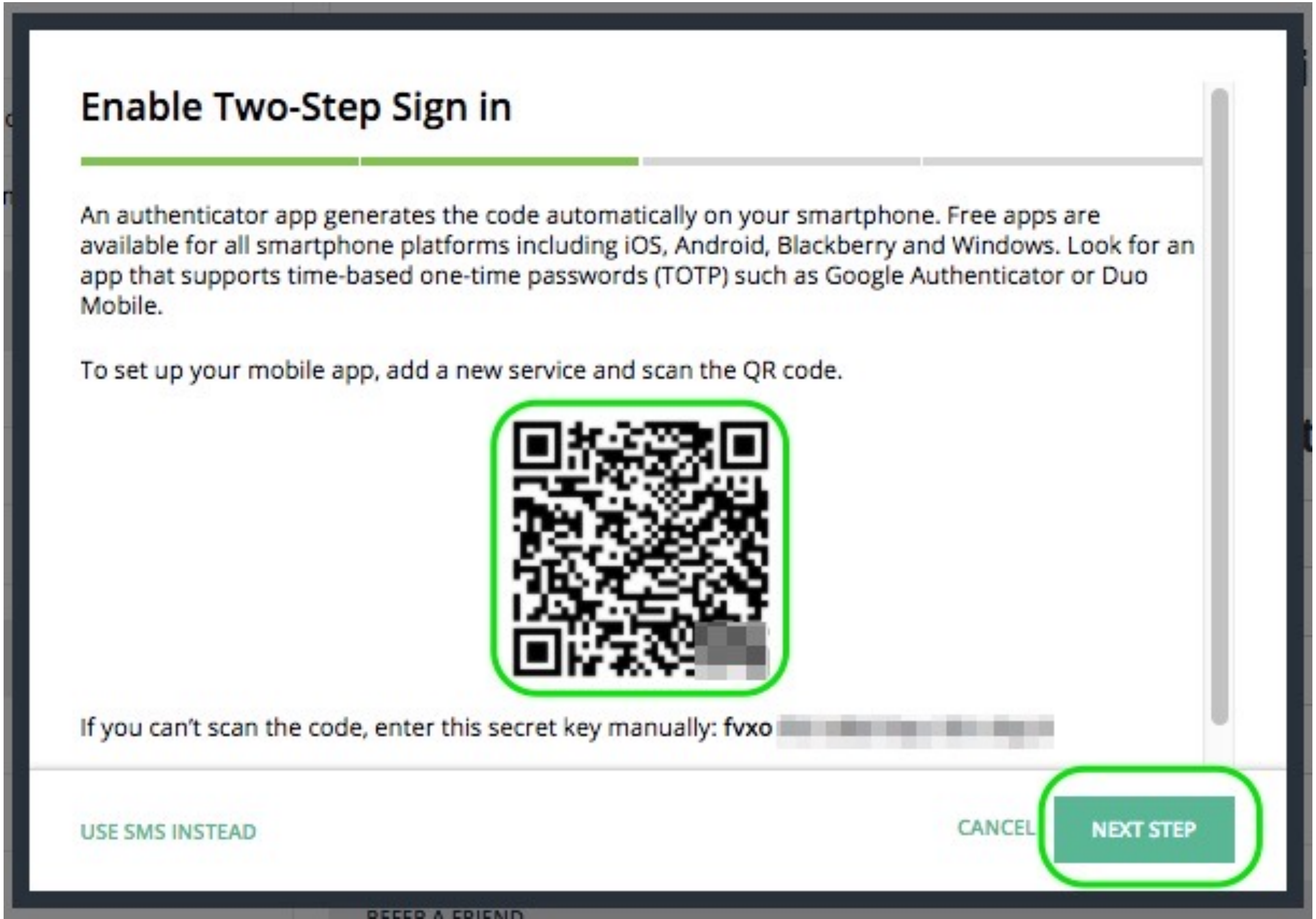
TC = floor((unixtime(now) − unixtime(*T0*)) / *TI*),

TOTP = HMAC(*K*, *TC*) % 10^d

Specially formatted SHA1-based signature

Given *K*, this algorithm can be run on your phone and by the service provider

Secret Sharing for TOTP



Hardware Two Factor

Special hardware designed to hold cryptographic keys

Physically resistant to key extraction attacks

- E.g. scanning tunneling electron microscopes

Uses:

- 2nd factor for OS log-on
- 2nd factor for some online services
- 2nd factor for password manager
- Storage of PGP and SSH keys

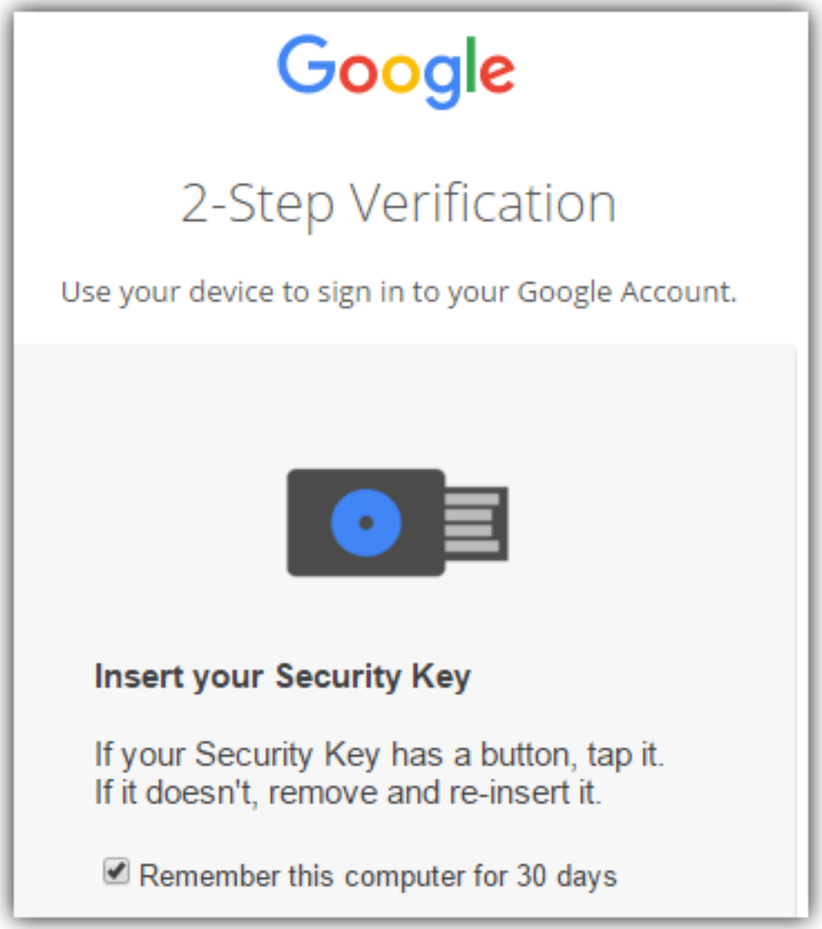
Universal 2nd Factor (U2F)

Supported by Chrome, Opera, and Firefox

Works with Google, Dropbox, Facebook, Github, Gitlab, etc.

Pro tip: always buy 2 security keys

- Associate both with your accounts
- Keep one locked in a safe, in case you lose your primary key ;)



Authentication in Linux

Unix, PAM, and crypt

Network Information Service (NIS, aka Yellow Pages)

Needham-Schroeder and Kerberos

Status Check

- At this point, we have discussed:
  - How to securely store passwords
  - Techniques used by attackers to crack passwords
  - Biometrics and 2nd factors
- Next topic: building authentication systems
  - Given a user and password, how does the system authenticate the user?
  - How can we perform efficient, secure authentication in a distributed system?

Authentication in Unix/Linux

- Users authenticate with the system by interacting with *login*
  - Prompts for username and password
  - Credentials checked against locally stored credentials
- By default, password policies specified in a centralized, modular way
  - On Linux, using Pluggable Authentication Modules (PAM)
  - Authorizes users, as well as environment, shell, prints MOTD, etc.

Example PAM Configuration

```
# cat /etc/pam.d/system-auth
#%PAM-1.0

auth required pam_unix.so try_first_pa
auth optional pam_permit.so
auth required pam_env.so

account required pam_unix.so
account optional pam_permit.so
account required pam_time.so

password required pam_unix.so try_first_pass nullok sha512 shadow
password optional pam_permit.so

session required pam_limits.so
session required pam_unix.so
session optional pam_permit.so
```

- Use SHA512 as the hash function
- Use /etc/shadow for storage

Unix Passwords

- Traditional method: *crypt*
  - First eight bytes of password used as key (additional bytes are ignored)
  - 12-bit salt
  - 25 iterations of DES on a given passwords
- Modern version of *crypt* are more extensible
  - Full password used
  - Up to 16 bytes of salt
  - Support for additional hash functions like MD5, SHA256, and SHA512
  - Key lengthening: defaults to 5000 iterations, up to $10^8 - 1$

Password Files

- Password hashes used to be in */etc/passwd*
  - World readable, contained usernames, password hashes, config information
  - Many programs read config info from the file…
  - But very few (only one?) need the password hashes
- Are world-readable hashes a good idea?

Password Storage on Linux

/etc/passwd

username:x:UID:GID:full_name:home_directory:shell

cbw:x:1001:1000:Christo Wilson:/home/cbw/:/bin/bash
n Mislove:/home/amislove/:/bin/sh

$<algo>$<salt>$<hash>
Algo: 1 = MD5, 5 = SHA256, 6 = SHA512

/etc/shadow

ername:password:last:may:must:warn:expire:disable:reserved

cbw:$1$0nSd5ewF$0df/3G7iSV49nsbAa/5gSg:9479:0:10000::::
amislove:1l3RxU5F1$:8172:0:10000::::

Distributed Authentication

Distributed Authentication

- Design a system that would authenticate you to the lab machines
  - should we have a /etc/shadow per machine that manages logins?
  - how about access to printers and files on other machines where a user may not have an account?

The Yellow Pages

- Network Information Service (NIS), a.k.a. the Yellow Pages
  - Developed by Sun to distribute network configurations
  - Central directory for users, hostnames, email aliases, etc.
  - Exposed through *yp** family of command line tools
- For instance, depending on */etc/nsswitch.conf*, hostname lookups can be resolved by using
  - */etc/hosts*
  - DNS
  - NIS
- Superseded by NIS+, LDAP (Lightweight Directory Access Protocol)

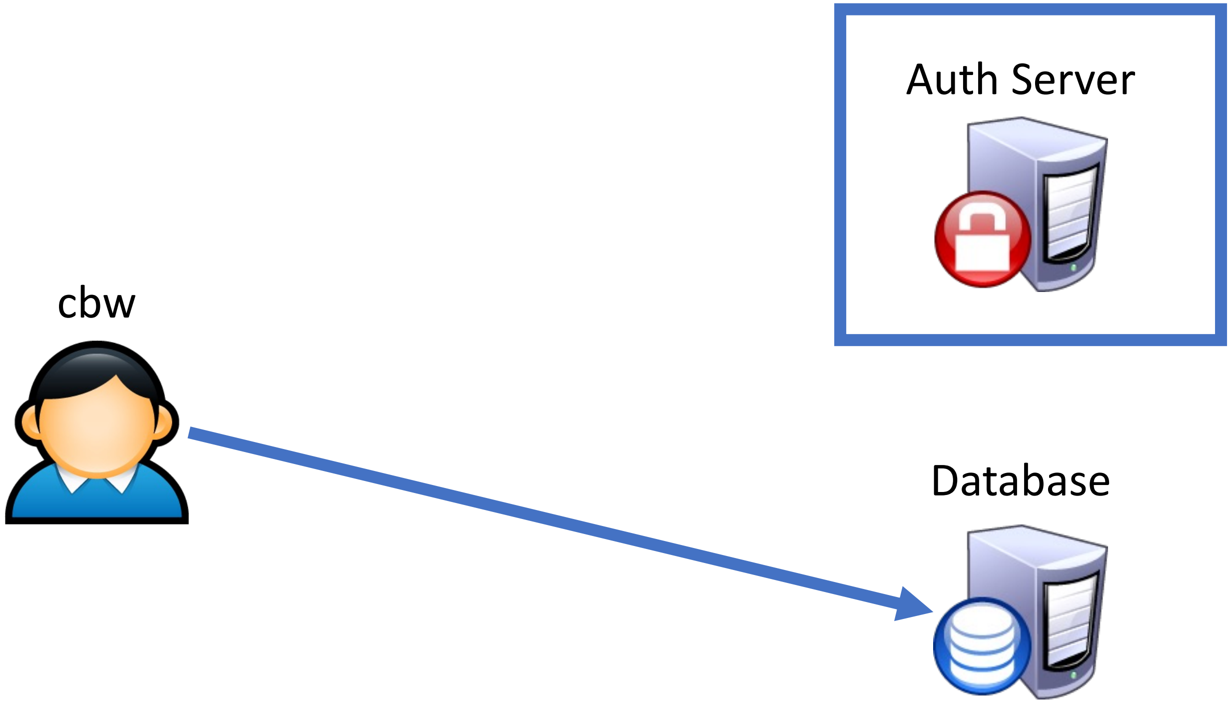
NIS Password Hashes

- *Crypt* based password hashes
- Is this secure?

```
[cbw@workstation ~] ypcat passwd
afbjune:qSAH.evuYFHaM:14532:65104::/home/afbjune:/bin/bash
philowe:T.yUMej3XSNAM:13503:65104::/home/philowe:/bin/bash
bratus:2omkwsYXWiLDo:6312:65117::/home/bratus:/bin/tcsh
adkap:ZfHdSwSz9WhKU:9034:65118::/home/adkap:/bin/zsh
amitpoon:i3LjTqgU9gYSc:8198:65117::/home/amitpoon:/bin/tcsh
kcole:sgYtUsOtyk38k:14192:65104::/home/kcole:/bin/bash
david87:vA06wxjJEUgBE:13055:65101::/home/david87:/bin/bash
loch:6HgIQrVkcBeiw:13729:65104::/home/loch:/bin/bash
ppkk315:s6CTSAkqqr/nU:14061:65101::/home/ppkk315:/bin/bash
haynesma:JYWaQUARSqDQE:14287:65105::/home/haynesma:/bin/bash
ckubicek:jYpwYhqqvr3tA:10937:65117::/home/ckubicek:/bin/tcsh
mwalz:wPIa5Bv/tFVb2:9103:65118::/home/mwalz:/bin/tcsh
sushma:G6XNe18GpeQj.:13682:65104::/home/sushma:/bin/bash
guerin1:n0Da2TmO9MDBI:14512:65105::/home/guerin1:/bin/bash
```

Distributed Authentication Revisited

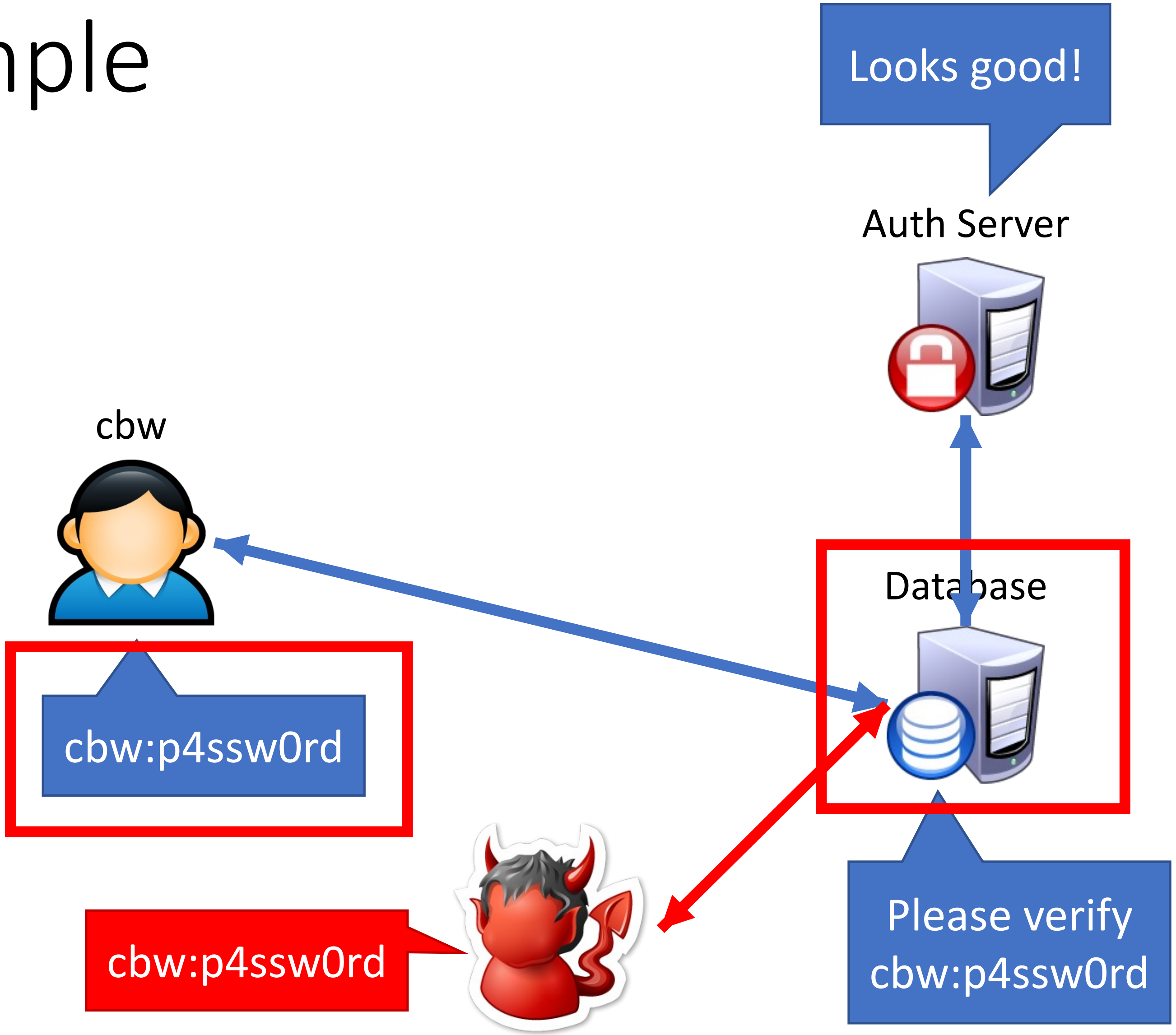
- Goal: a user would like to use some resource on the network
  - File server, printer, database, mail server, etc.
- Problem: access to resources requires authentication
  - Auth Server contains all credential information
  - You do not want to replicate the credentials on all services



What's the threat model here?

Distributed Auth Example

- Idea: client forwards user/password to service, service queries Auth Server



Symmetric Key Agreement among Multiple Parties

- For a group of N parties, every pair needs to share a different symmetric key
  - What is the number of keys?
  - What secure channel to use to establish the keys?

- How to establish such keys
  - Symmetric Encryption - Use a central authority, a.k.a. (TTP).
  - Asymmetric Encryption – PKI.

Needham-Schroeder Protocol

- Which message authenticates Alice and the Server?
- What purpose does the challenge nonce N_j have?
- How can Bob be sure that he is receiving a session key from the trusted server? (note that Bob does not talk to the trusted server at any point)

- Let Alice A and Bob B be two parties that trust server S
- K_{AS} and K_{BS} are shared secrets between [A, S] and [B, S]
- K_{AB} is a negotiated session key between [A, B]
- N_i and N_j are random nonces generated by A and B

1) $A \rightarrow S: A, B, N_i$
2) $S \rightarrow A: \{N_i, K_{AB}, B, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$
3) $A \rightarrow B: \{K_{AB}, A\}_{K_{BS}}$
4) $B \rightarrow A: \{N_j\}_{K_{AB}}$
5) $A \rightarrow B: \{N_j - 1\}_{K_{AB}}$

K_{AS} is not sent in the clear, authenticates S and A

K_{BS} is not sent in the clear, authenticates B

Challenge nonce forces A to acknowledge they have K_{AB}

Needham-Schroeder Example

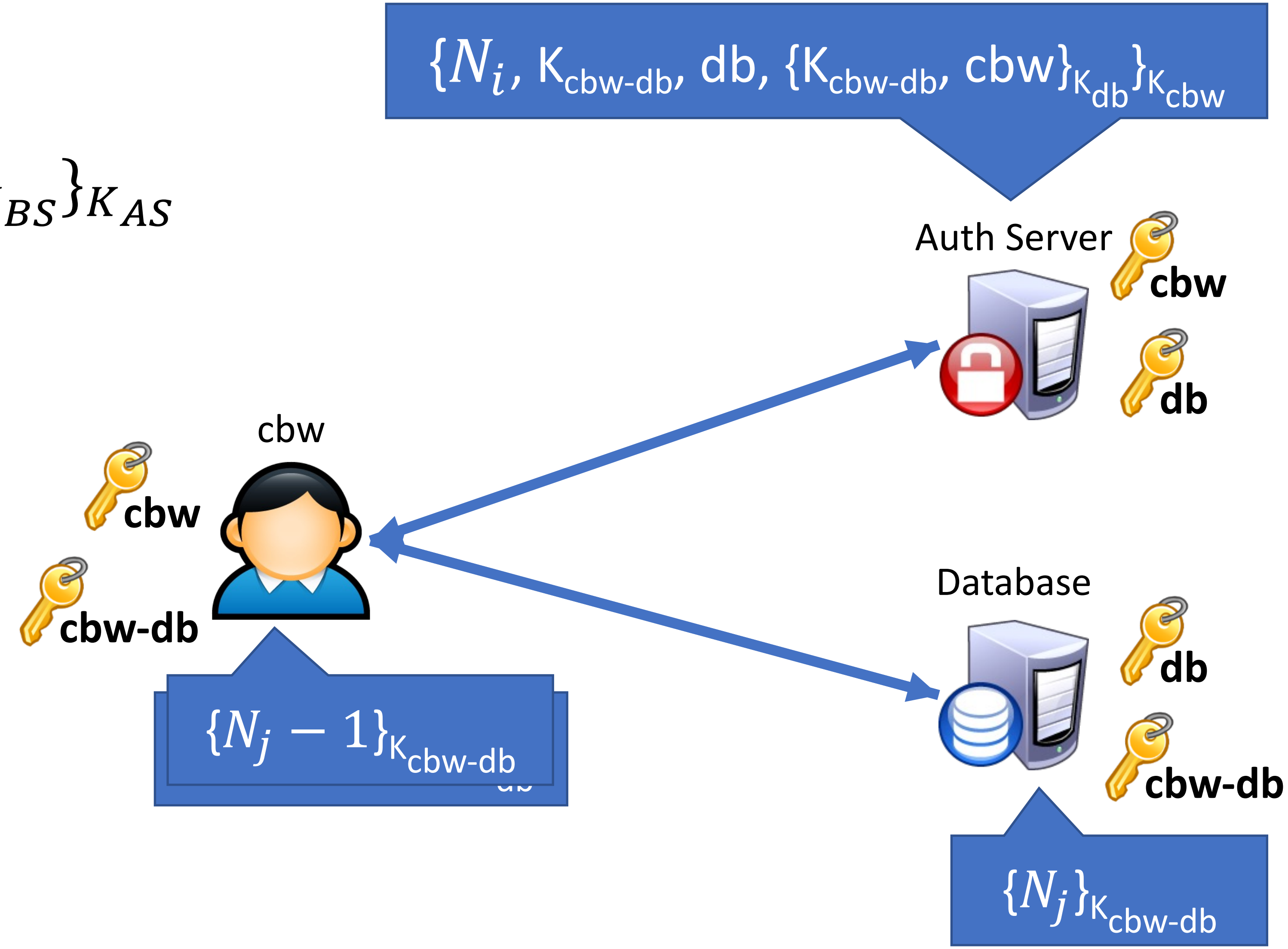
1) $A \rightarrow S: A, B, N_i$

2) $S \rightarrow A: \{N_i, K_{AB}, B, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$

3) $A \rightarrow B: \{K_{AB}, A\}_{K_{BS}}$

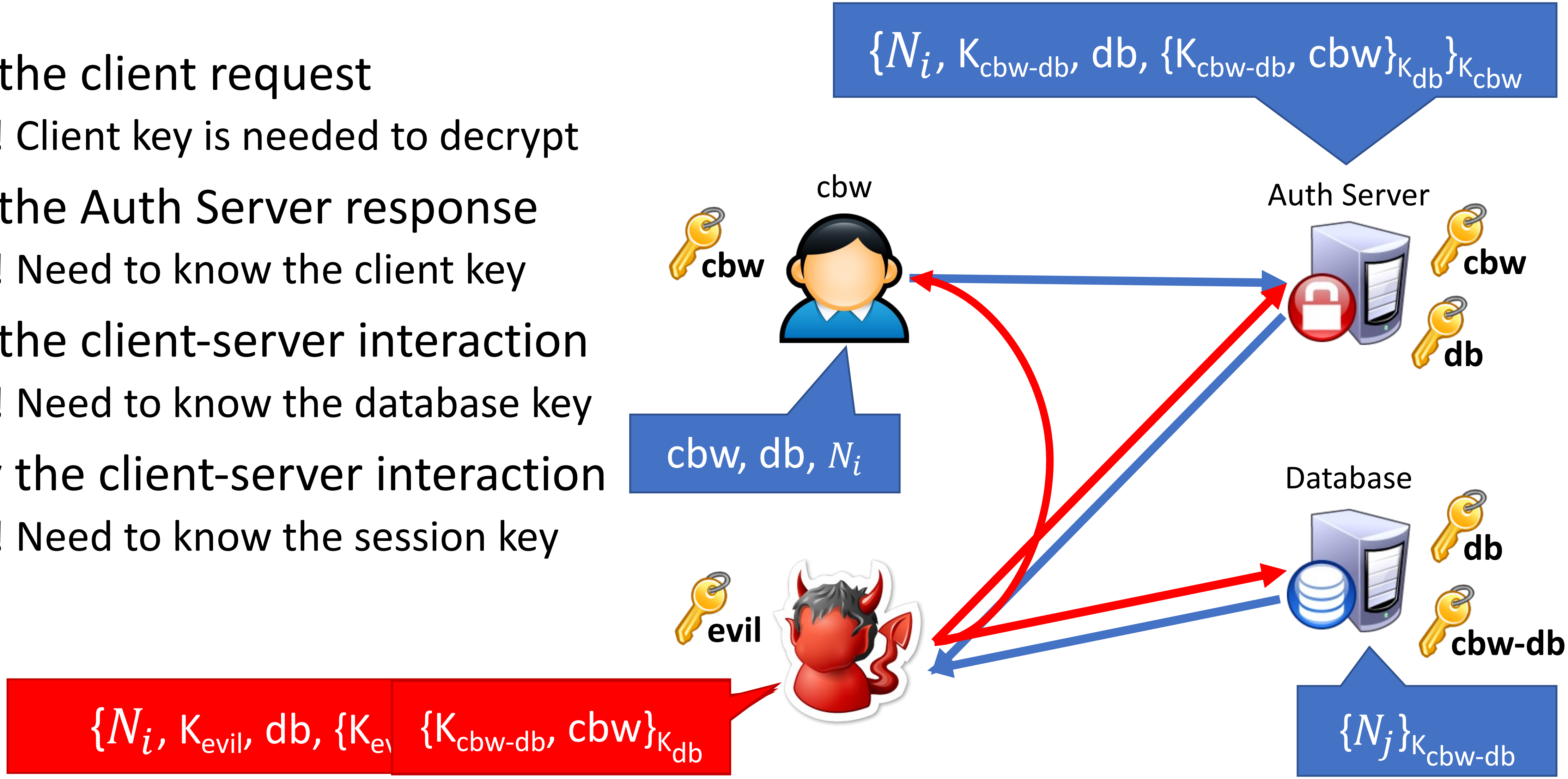
4) $B \rightarrow A: \{N_j\}_{K_{AB}}$

5) $A \rightarrow B: \{N_j - 1\}_{K_{AB}}$



Attacking Needham-Schroeder

- Spoof the client request
  - Fail! Client key is needed to decrypt
- Spoof the Auth Server response
  - Fail! Need to know the client key
- Spoof the client-server interaction
  - Fail! Need to know the database key
- Replay the client-server interaction
  - Fail! Need to know the session key



Replay Attack

Typical, Benign Protocol

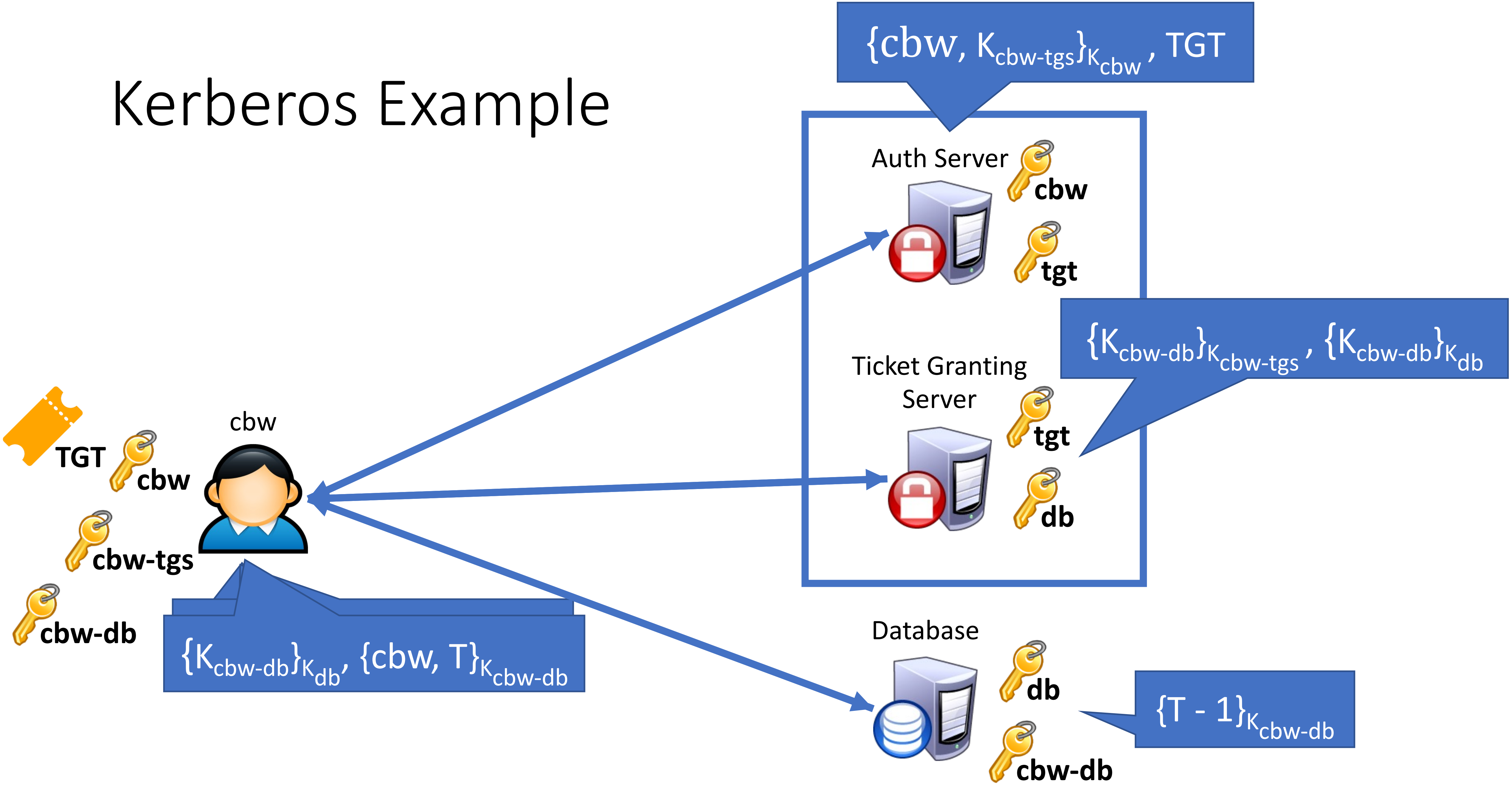
1) $A \rightarrow S: A, B, N_i$
2) $S \rightarrow A: \{N_i, K_{AB}, B, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$
3) $A \rightarrow B: \{K_{AB}, A\}_{K_{BS}}$
4) $B \rightarrow A: \{N_j\}_{K_{AB}}$
5) $A \rightarrow B: \{N_j - 1\}_{K_{AB}}$

- Let's say an attacker hacks Alice and steals K_{AS} and K_{AB} what damage can the attacker do?
- In particular, which steps of the protocol can they spoof? (act as Alice)?
- Let's say Alice discovers that she has been hacked and changes K_{AS} to $K_{ANEW'S}$ will the attack still succeed?

Kerberos

- Created as part of MIT Project Athena
  - Based on Needham-Schroeder
- Provides mutual authentication over untrusted networks
  - Tickets as assertions of authenticity, authorization
  - Forms basis of Active Directory authentication in Microsoft networks
- Principals
  - Client
  - Server
  - Key Distribution Center (KDC)
    - Authentication server (AS)
    - Ticket granting server (TGS)

Kerberos Example



Attacking Kerberos

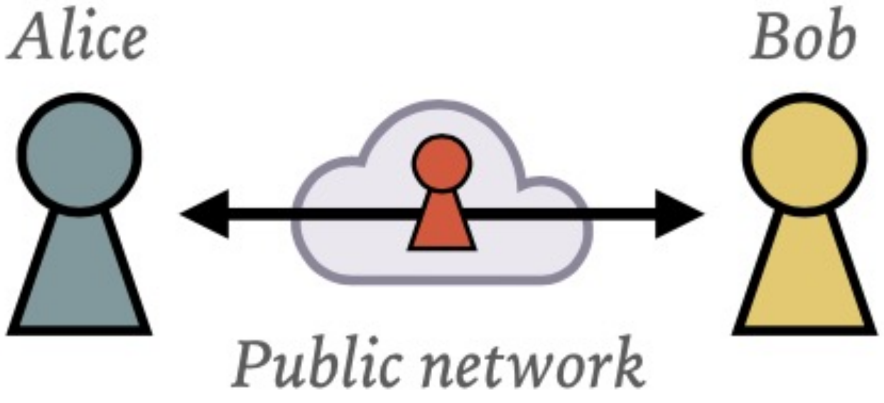
- Don’t put all your eggs in one basket
  - The Kerberos Key Distribution Server (KDS) is a central point of failure
  - DoS the KDS and the network ceases to function
  - Compromise the KDS leads to network-wide compromise
- Time synchronization
  - Inaccurate clocks lead to protocol failures (due to timestamps)
  - Solution?
  - Use NTP (Network Time Protocol)

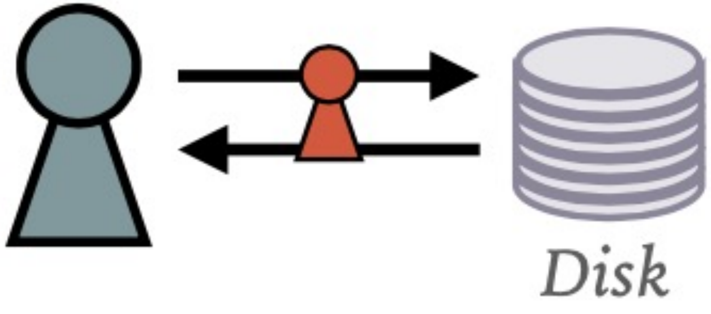
Sources

1. Many slides courtesy of Wil Robertson: https://wkr.io
2. Honeywords, Ari Juels and Ron Rivest: http://www.arijuels.com/wp-content/uploads/2013/09/JR13.pdf

- For more on generating secure passwords, and understanding people's mental models of passwords, see the excellent work of Blas Ur: http://www.blaseur.com/pubs.htm

Symmetric Key Cryptography





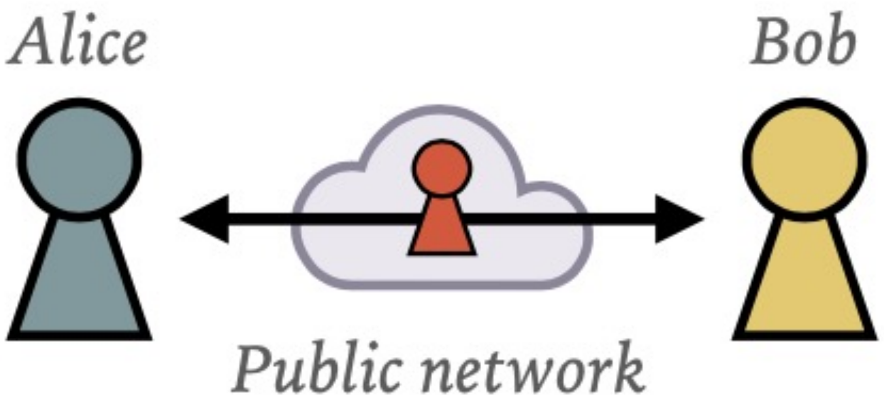
Confidentiality	Keep others from reading Alice's messages/data
Integrity	Keep others from undetectably tampering with Alice's messages/data
Authenticity	Keep others from undetectably impersonating Alice (keep her to her word too!)

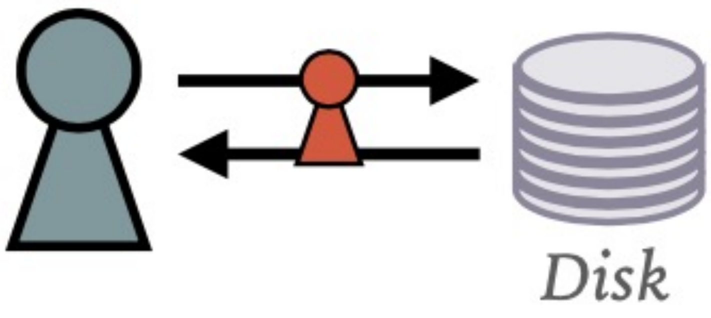
Block Ciphers

Limitations?

- what if Eve modifies the packet in transit?
- How do we share keys?

Scenarios and Goals



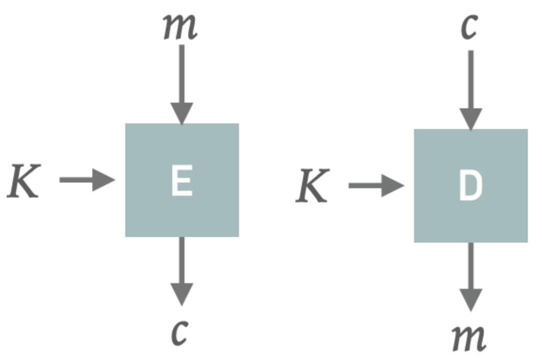


Confidentiality	Keep others from reading Alice’s messages/data
Integrity	Keep others from undetectably tampering with Alice’s messages/data
Authenticity	Keep others from undetectably impersonating Alice (keep her to her word too!)

Message Authentication Codes (MACs)

BLACKBOX #2:
MESSAGE AUTHENTICATION CODE (MAC)

Symmetric Key Cryptography

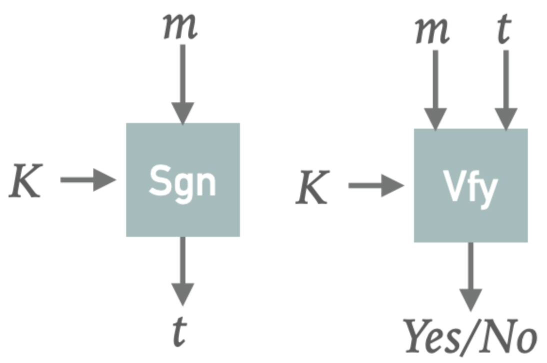


CONFIDENTIALITY

Block ciphers

Deterministic $\implies$ *use IVs*

Fixed block size $\implies$ *use encryption "modes"*

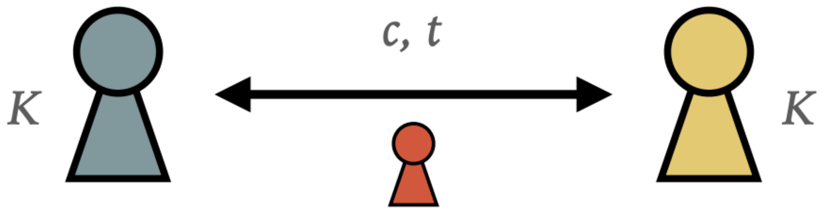


INTEGRITY

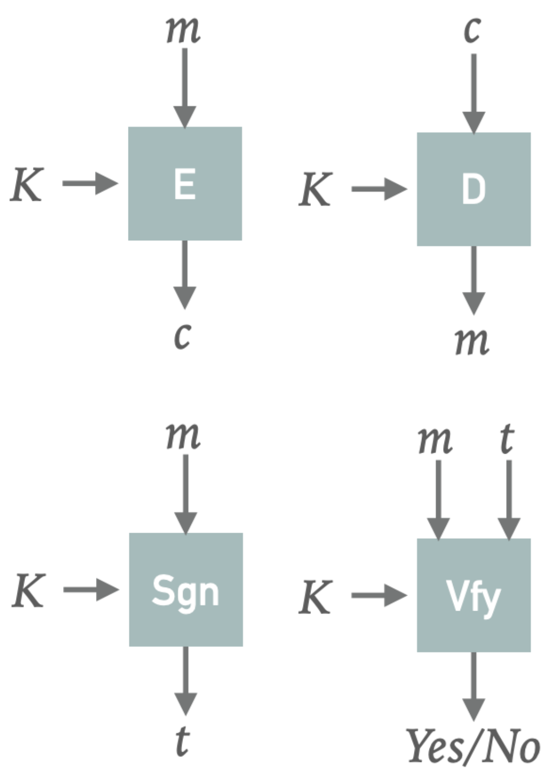
Message Authentication Codes (MACs)

Send (message, tag) pairs

Verify that they match



Could we simply use symmetric key cryptography (i.e. block ciphers) to achieve integrity?



CONFIDENTIALITY

Block ciphers

Deterministic ⟹ use IVs

Fixed block size ⟹ use encryption "modes"

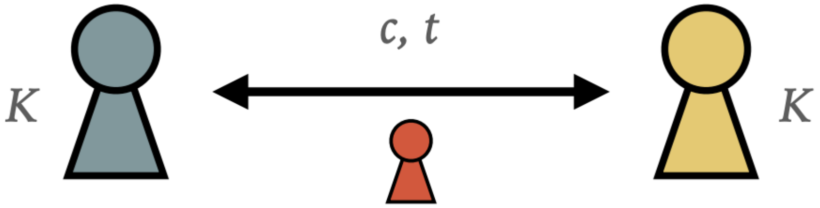
INTEGRITY

Message Authentication Codes (MACs)

Send (message, tag) pairs

Verify that they match

A. Yes
B. No
C. Maybe
D. Under some circumstances



General adversarial goals

- Total Break: Adversary is able to fund the secret key for signing and forge any signature of any message
- Selective forgery: Adversary is able to create valid signatures on a message chosen by someone else, with a significant probability.
- Existential Forgery: Adversary can create a pair of (message, signature) such that the signature of the message is valid.

- Ciphertext only Attack: Adversary knows only the verification function
- Known Plaintext Attack: Adversary knows a list of messages previously signed by Alice
- Chosen Plaintext Attack: Adversary can choose what messages they want Alice to sign, and knows both the smessage and the corresponding signature

Attacker Goal: Existential Forgery

- A MAC is secure if an attacker cannot demonstrate an existential forgery despite being able to perform a chosen plaintext attack:

- Chose plaintext:
  - Attacker gets to choose m1, m2, m3, …
  - And in return gets a properly computed t1, t2, t3, …

- Existential forgery:
  - Construct a new (m,t) pair such that Vfy(k, m, t) = Y

BLACKBOX #3:
HASH FUNCTIONS

Hash Function Properties

- Very fast to compute
- Takes arbitrarily-sized inputs, returns fixed-sized output
- Pre-image resistant:

  Given H(m), hard to determine m
- Collision resistant

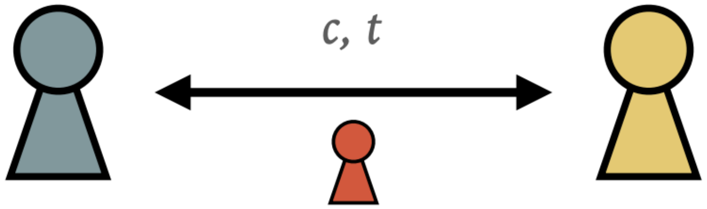
  Given m and H(m), hard to find m′≠ m s.t. H(m) = H(m′)

Good hash functions: SHA family (SHA-256, SHA-512, …)

Authenticated Encryption: Secrecy + Integrity

We have seen how we can achieve two independent goals: encryption and authentication. How about putting them together?

k1, k2



k1, k2

```
m
Enc(k1, m) = c
Mac(k2, m) = t
```

```
Dec(k1, c) = m
Verify(k2, m, t) = 1?
```

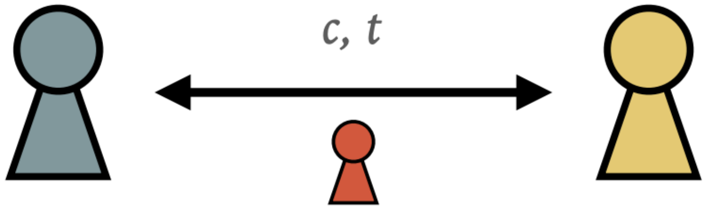
Encrypt and Authenticate: Is it secure?

A. Yes, encryption is randomized with proper K, IV
B. No the tag might leak information
C. No the MAC is deterministic

Encrypt then authenticate

We have seen how we can achieve two independent goals: encryption and authentication. How about putting them together?

k1, k2



k1, k2

```
m
Enc(k1, m) = c
Mac(k2, c) = t
```

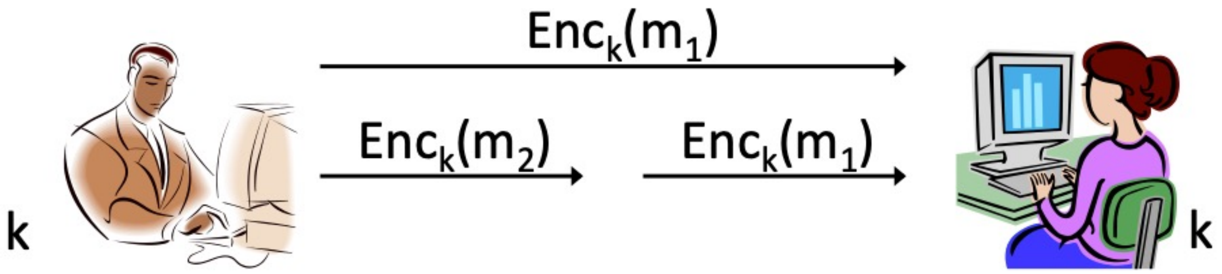
```
Verify(k2, c, t) = 1?
Dec(k1, c) = m
```

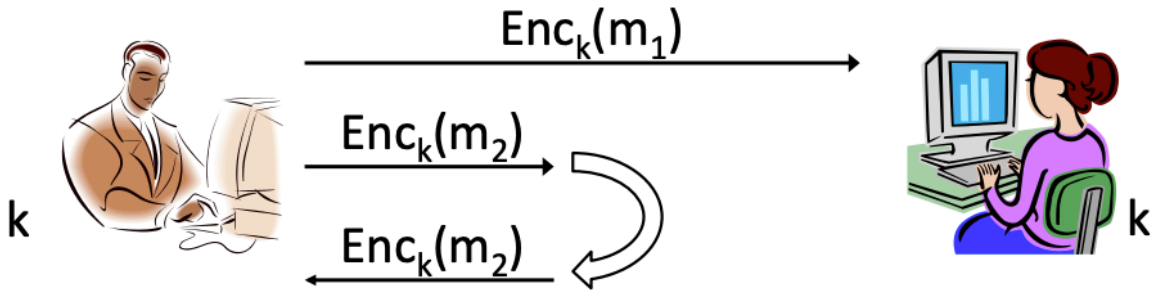
Encrypt then Authenticate: Is it secure?

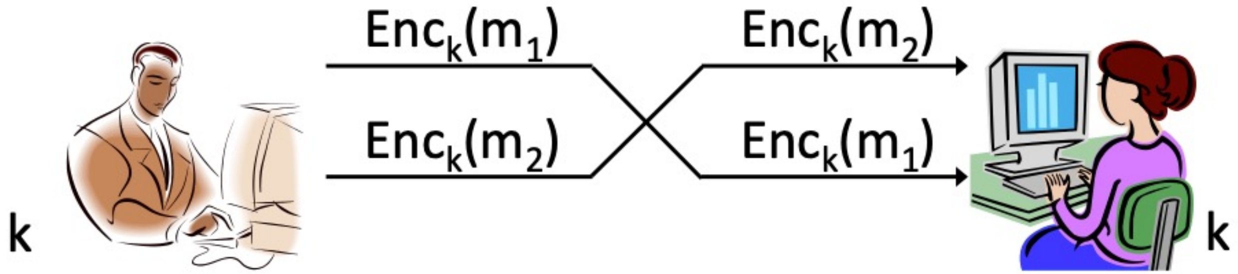
A. Yes, encryption is randomized with proper K, IV
B. No the tag might leak information
C. No the MAC is deterministic

Secure Sessions: Consider parties who wish to communicate securely over the course of a session using authenticated encryption. Are they immune to the following attacks?

- Securely = secrecy and integrity
- Session = period of time over which parties are willing to maintain state.



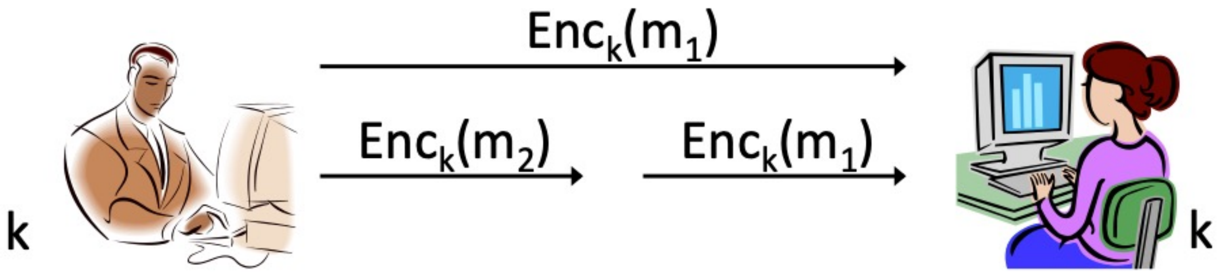


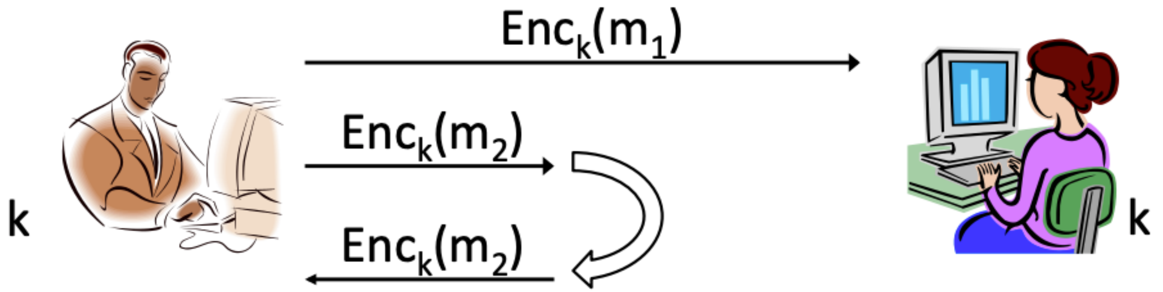


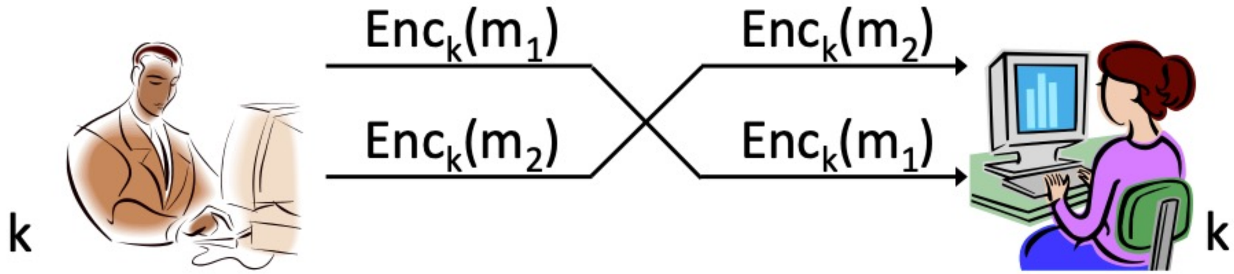
A. Yes
B. No

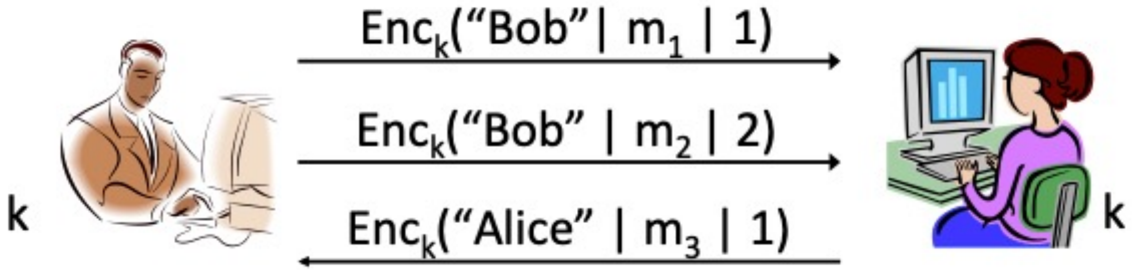
Secure Sessions: Consider parties who wish to communicate securely over the course of a session using authenticated encryption. Are they immune to the following attacks?

- Securely = secrecy and integrity
- Session = period of time over which parties are willing to maintain state.

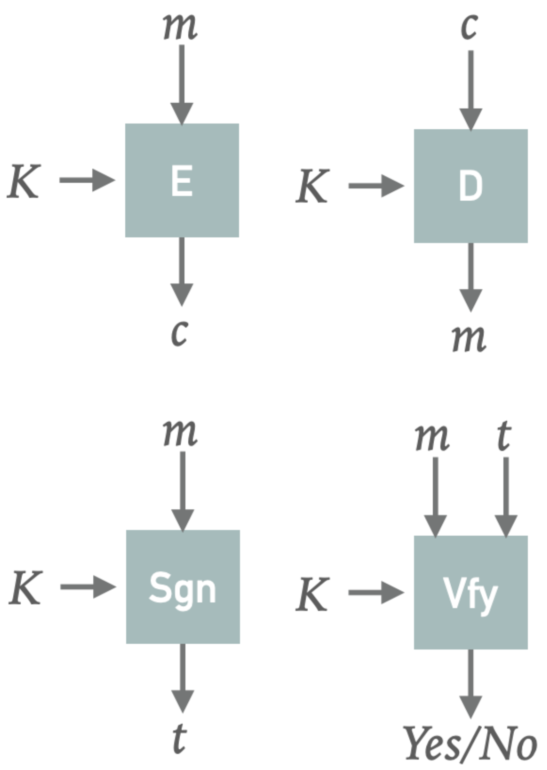








Symmetric Key Cryptography



CONFIDENTIALITY

Block ciphers

Deterministic $\Longrightarrow$ *use IVs*

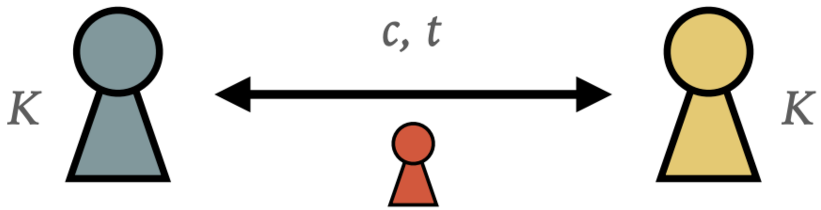
Fixed block size $\Longrightarrow$ *use encryption "modes"*

INTEGRITY

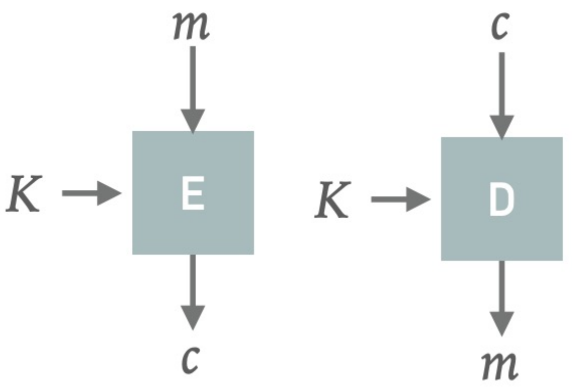
Message Authentication Codes (MACs)

Send (message, tag) pairs

Verify that they match



Symmetric Key Cryptography

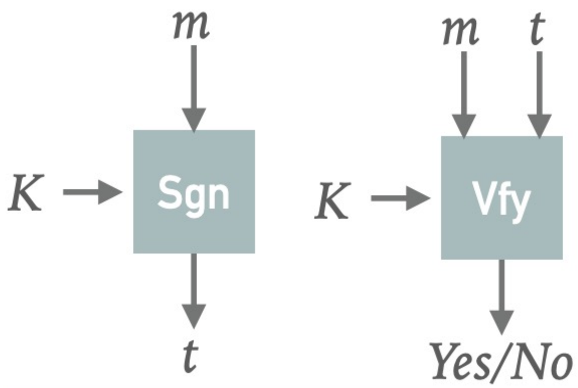


CONFIDENTIALITY

Block ciphers

Deterministic ⟹ use IVs

Fixed block size ⟹ use encryption "modes"



INTEGRITY

Message Authentication Codes (MACs)

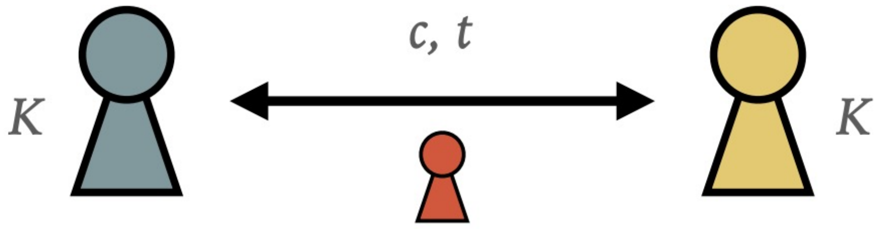
Send (message, tag) pairs

Verify that they match

Next

How do we establish K?

How do we know with whom we are communicating?



BLACKBOX #4:
DIFFIE HELLMAN KEY ESTABLISHMENT

Asymmetric/Public-key Cryptography

- main insight: separate keys for different functions
- Keys come in pairs, and are related to each other by a specific algorithm.
  - Public key (PK): used to encrypt or verify signatures
  - Private key (SK): used to decrypt and sign
- Encryption and decryption are inverse operations
- Secrecy: ciphertext reveals nothing about the plaintext
  - computationally hard to decrypt in polynomial time without key

Diffie-Helman Key Exchange

$x \bmod N$

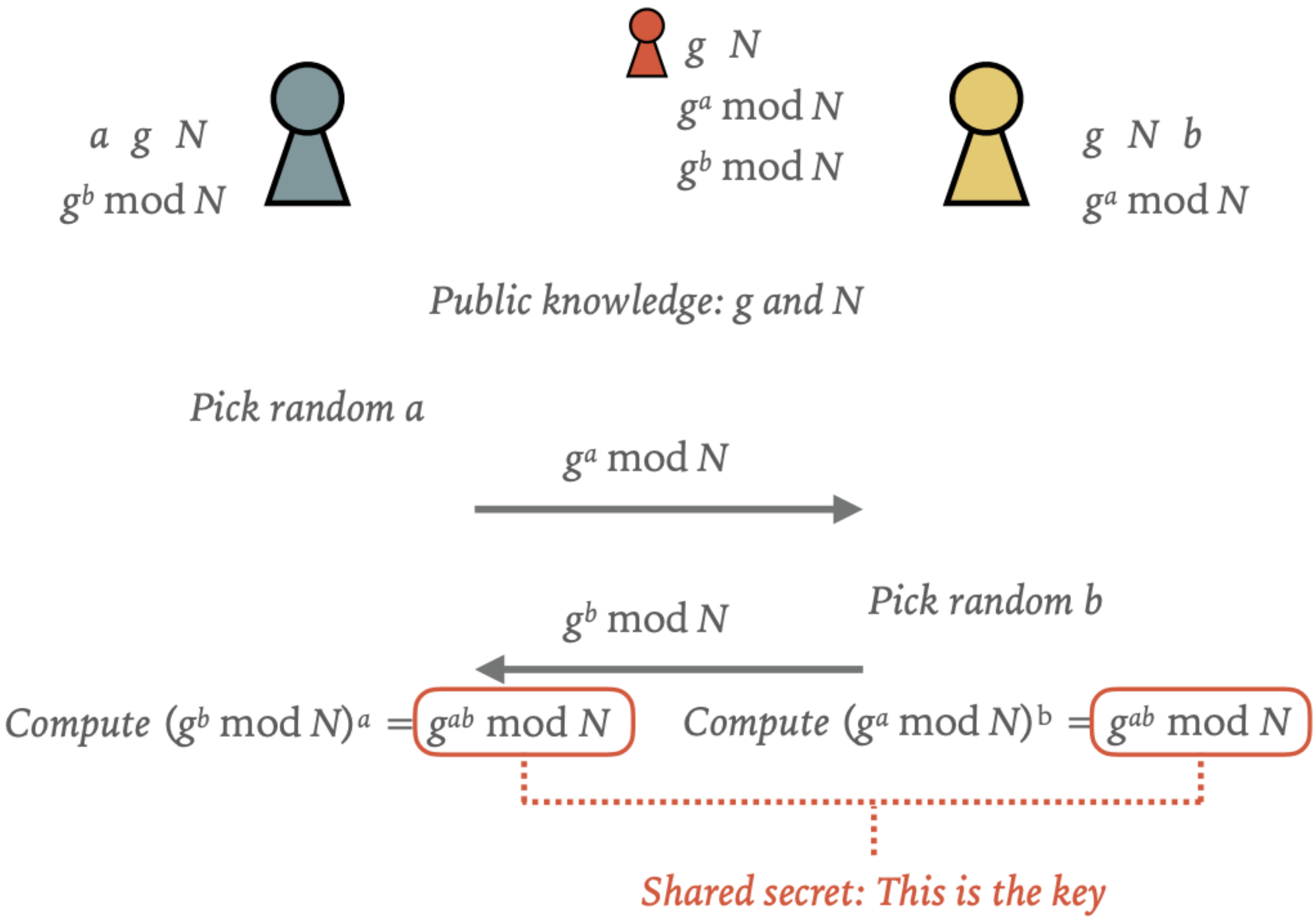
g is a generator of mod N if
$\{1, 2, \ldots, N\text{-}1\} = \{g^0 \bmod N, g^1 \bmod N, \ldots, g^{N-2} \bmod N\}$

N=5, g=3
$3^0 \bmod 5 = 1$ $3^1 \bmod 5 = 3$ $3^2 \bmod 5 = 4$ $3^3 \bmod 5 = 2$

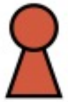
Given x and g, it is efficient to compute
$g^x \bmod N$

Given g and g^x, it is efficient to compute x
(simply take $\log_g g^x$)

Given g and g^x mod N it is *infeasible* to compute x
Discrete log problem



g N
$g^a \bmod N$
$g^b \bmod N$
a g N
$g^b \bmod N$
g N b
$g^a \bmod N$
Public knowledge: g and N
Pick random a
$g^a \bmod N$
Pick random b
$g^b \bmod N$
Compute $(g^b \bmod N)^a = g^{ab} \bmod N$
Compute $(g^a \bmod N)^b = g^{ab} \bmod N$
Shared secret: This is the key



$g \quad N$

$g^a \bmod N$

$g^b \bmod N$

$g^{ab} \bmod N$

Note that just multiplying g^a and g^b won't suffice:

$$g^a \bmod N * g^b \bmod N \quad = g^{a+b} \bmod N$$

Key property:

An *eavesdropper* cannot infer the shared secret (g^{ab}).

But what about *active intermediaries*?

$g \quad N$

$g^a \bmod N$

$g^b \bmod N$

$g^{ab} \bmod N$

Given g and $g^x \bmod N$ it is *infeasible* to compute x
Discrete log problem

Note that just multiplying g^a and g^b won't suffice:

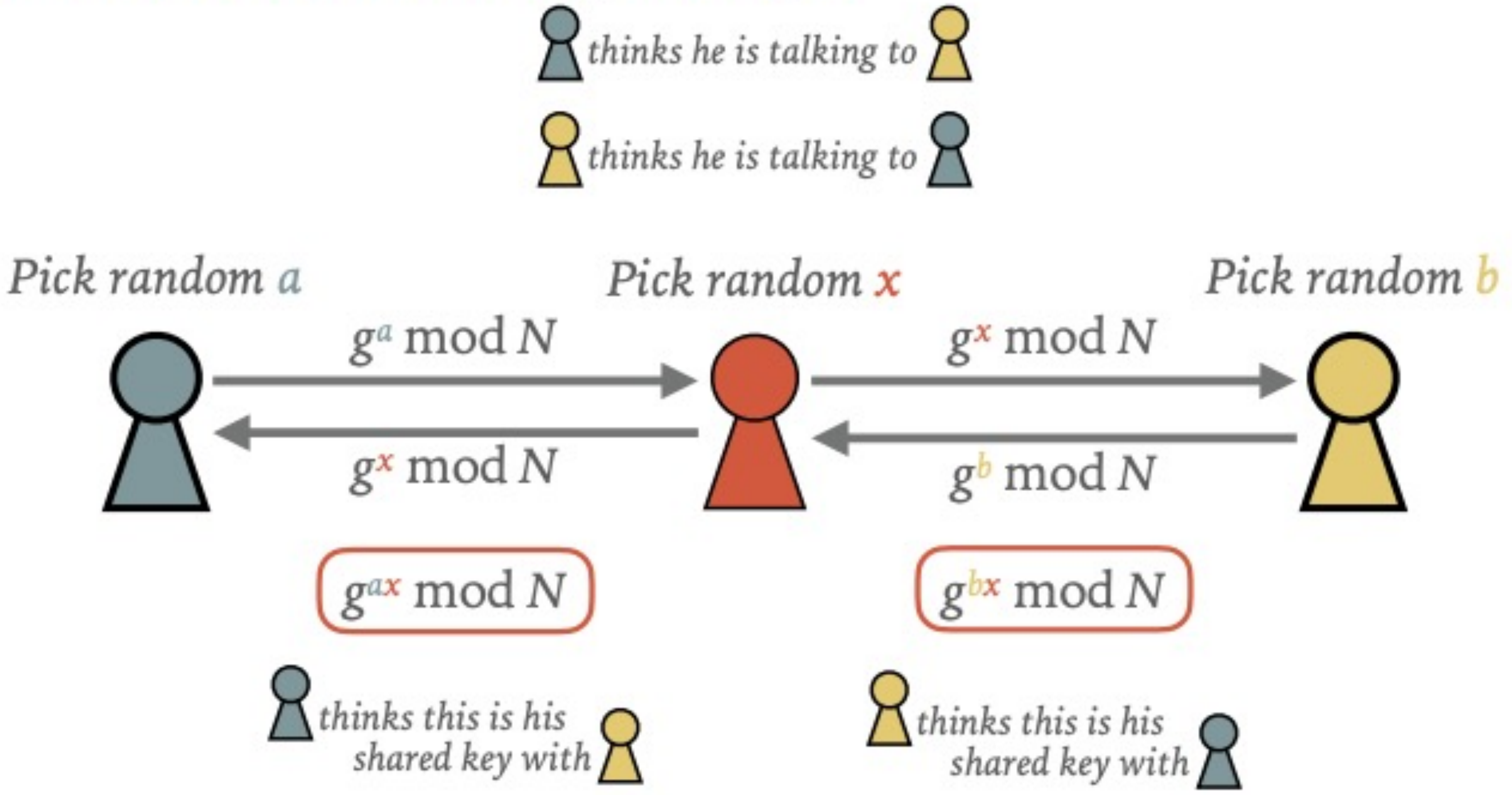
$$g^a \bmod N * g^b \bmod N \quad = g^{a+b} \bmod N$$

Key property:

An *eavesdropper* cannot infer the shared secret (g^{ab}).

But what about *active intermediaries*?

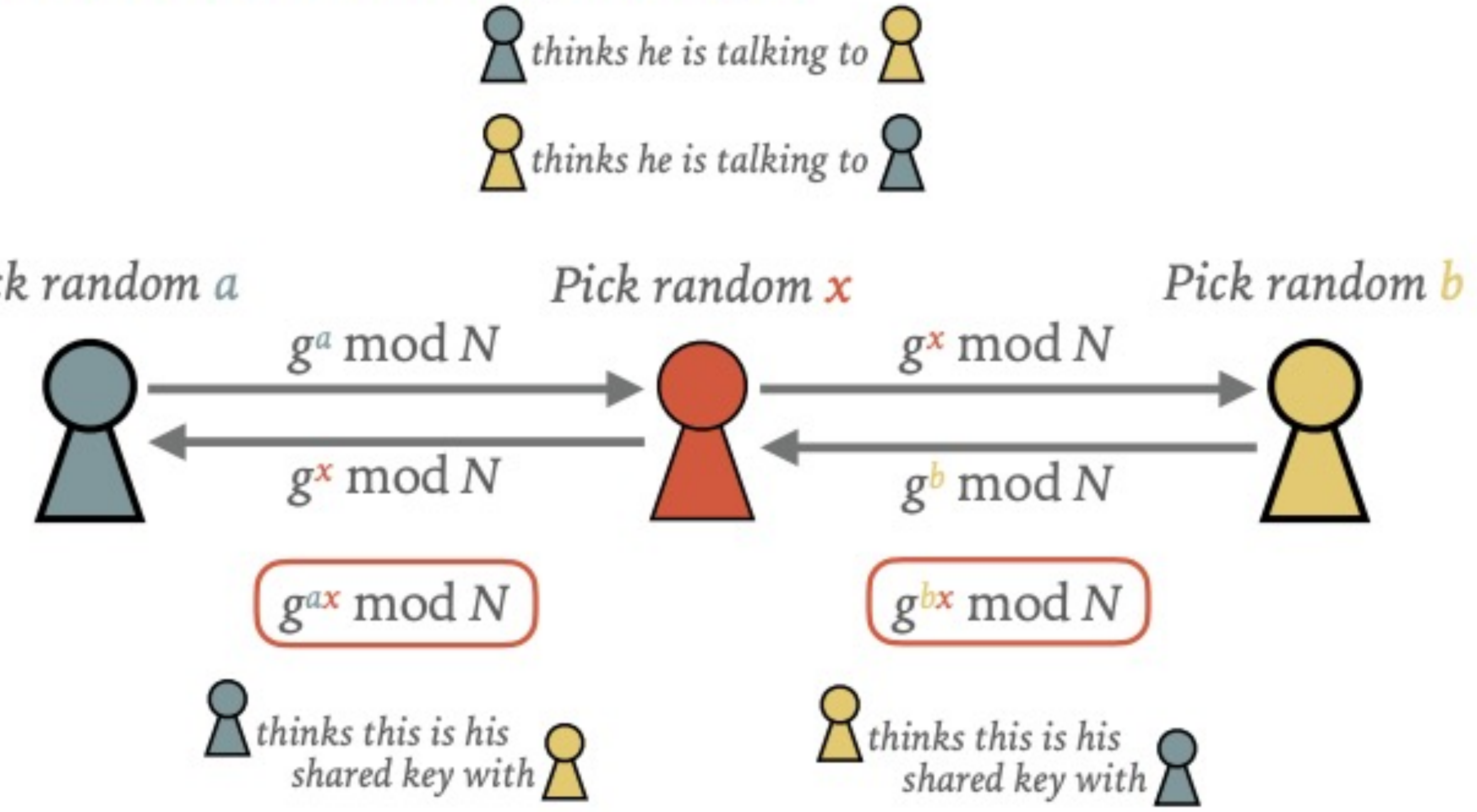
The attacker can interpose between the two communicating parties and insert, delete, and modify messages.



The attacker can now eavesdrop on the conversation.

Key property: Diffie-Hellman is *not* resilient to a MITM attack

The attacker can interpose between the two communicating parties and insert, delete, and modify messages.



The attacker can now eavesdrop on the conversation.

Key property: Diffie-Hellman is *not* resilient to a MITM attack

Fix: Need to authenticate messages

Computational complexity for integer problems

- Integer multiplication is efficient to compute
- There is no known polynomial-time algorithm for general purpose factoring.
- Efficient factoring algorithms for many types of integers. *Easy to find small factors of random integers.*
- Modular exponentiation is efficient to compute
- Modular inverses are efficient to compute

Textbook RSA Encryption

Public Key pk

N = pq *modulus*

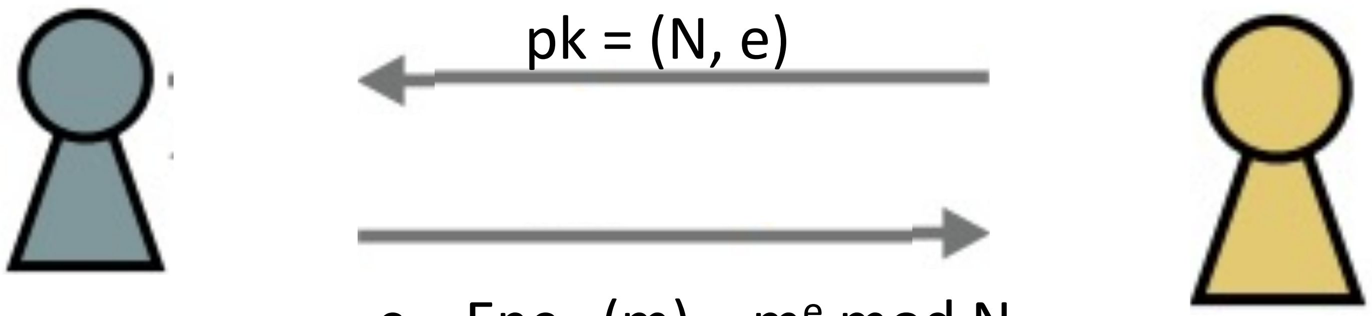
e *encryption exponent*

Secret key sk

p, q *primes*

d *decryption exponent*

$d = e^{-1} \bmod (p-1)(q-1) = e^{-1} \bmod \Phi(N)$



$c = Enc_{PK}(m) = m^e \bmod N$

$d = Dec_{SK}(c) = c^d \bmod N$

RSA Security

- Best algorithm to break RSA: Factor N and compute d
- Factoring is not efficient in general
- Current key size recommendations: N >= 2048 bits
- *Do not implement this yourself. Factoring is hard only for some integers, and textbook RSA is insecure.*