

# Introduction to Computer Security Lecture Slides

© 2024 by [Mihai Chiroiu](#) & [Florin Stancu](#)

is licensed under [Attribution-NonCommercial-ShareAlike 4.0  
International](#)

# Authentication and key establishment

# Who do we authenticate

- **Users:**
  - The human operator
  - Authentication is typically slow
  - Local or over-the-wire
  - **Authentication** only
    - human brain cannot do proper cryptography just yet :((
- **Principals**
  - User's digital identity
  - Authentication should be fast and scalable
  - Mostly over-the-wire
  - **Goal: Authentication & key establishment**



# Identification vs authentication

- Identification means one-from-many
  - Find your fingerprints in a police database
- Authentication means one-to-one relations
  - Compare your (based on the username) input to a previously saved one
  - Enrollment (can be slow, must be precise) vs Recognition (must be quick)
- Cooperation
  - In identification, the user does not cooperate
  - In authentication, the user is cooperative

# AAA framework

- Identify
  - Map a real-person/subject to a virtual account
- **Authenticate**
  - Request a proof from the account
- Authorize
  - Verify if the account can access a resource
- Accounting
  - Log/monitor what the account is doing

# Just authentication?

- Is authentication alone enough?
  - Yes, for local systems (e.g., console/GUI login)
  - Not very good for remote systems (e.g. telnet) -> session hijack
- Key establishment only?
  - For anonymity purposes
  - Not very practical (e.g. plain D-H over MitM channel)
- We need both!

# Attacks?

Attack	Short description
replay	reusing a previously captured message in a later protocol run
reflection	replaying a captured message to the originating party
relay	forwarding a message in real time from a distinct protocol run
interleaving	weaving together messages from distinct concurrent protocols
middle-person	eavesdropping on communication
bruteforce	for short credentials (e.g., PIN codes) – without rate limiting
dictionary	using a heuristically prioritized list in a guessing attack
forward search	feeding guesses into a one-way function, seeking output matches
pre-capture	extracting client OTPs by social engineering, for later use

# Authentication



# The concept

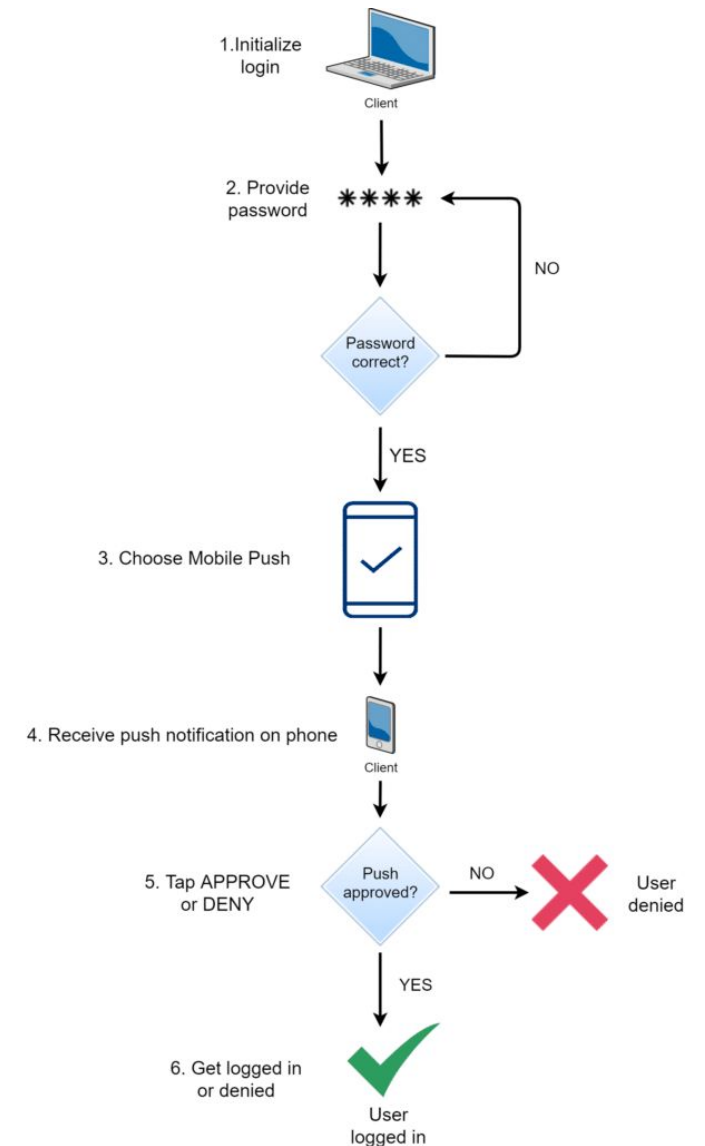
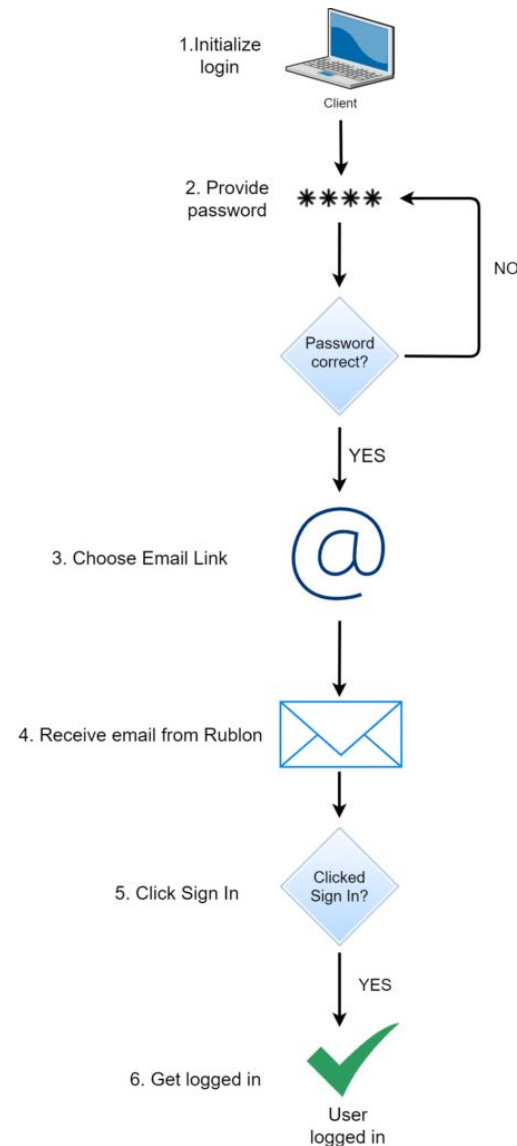
- The authenticator (e.g. server, website) asks to prove that you are who you pretend to be based on one or more pieces of evidence called **factors**.
  - May also be mutual: server also authenticates to the client!
- The evidence can be presented either directly (e.g. password authentication) or indirectly by using it in cryptographic calculation (e.g. challenge authentication protocol).
  - Indirect proof use some form of cryptographic algorithms.
  - Indirect proof also known as implicit authentication.

# Types of factors

- Something you know (Knowledge Factor)
- Something you have (Possession Factor)
- Something you are (Inherence Factor)
  
- Other authentication attributes that can be used:
  - Somewhere you are
  - Someone you know

# Chaining factors

- N-factor authentication
  - Factors should be different
- N-step verification
  - Can be same factor



<https://rublon.com/blog/2fa-2sv-difference/>

# Something you know - Passwords

- Require people to remember them
  - Used on multiple occasions
- Shoulder surfing / key logging
- Can be enhanced through policies
  - E.g. Minimum 20 characters

**44 million Microsoft users reused passwords in the first three months of 2019**

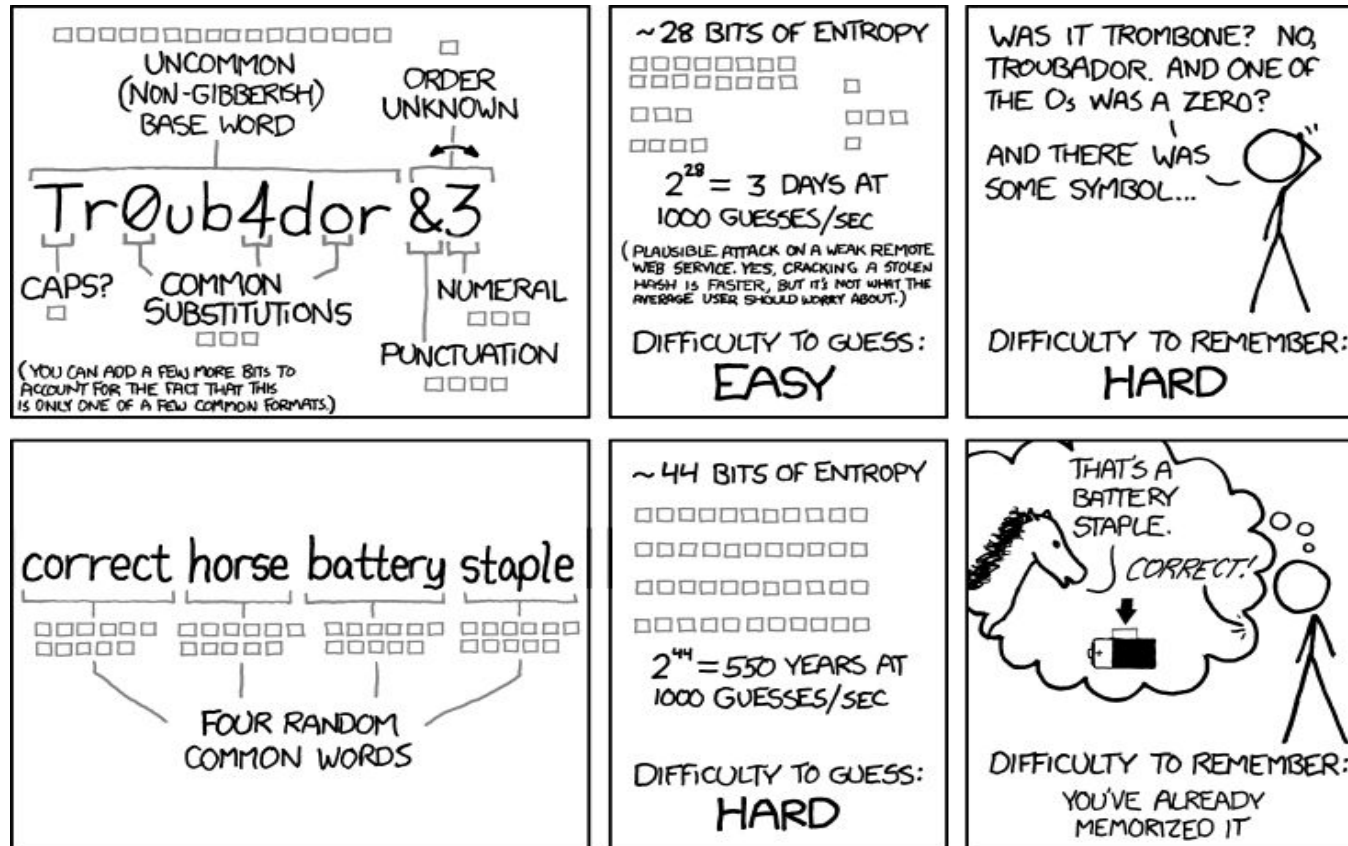
Microsoft used a database of three billion publicly leaked credentials to identify users who reused passwords.

# Something you know - Passwords

Findings			
All countries			
Get the 2019-2021 password list			
RANK	PASSWORD	TIME TO CRACK IT	COUNT
1	password	< 1 Second	4,929,113
2	123456	< 1 Second	1,523,537
3	123456789	< 1 Second	413,056
4	guest	10 Seconds	376,417
5	qwerty	< 1 Second	309,679
6	12345678	< 1 Second	284,946
7	111111	< 1 Second	229,047
8	12345	< 1 Second	188,602
9	co1123456	11 Seconds	140,505

<https://nordpass.com/most-common-passwords-list/>

# Something you know - Passwords



THROUGH 20 YEARS OF EFFORT, WE'VE SUCCESSFULLY TRAINED EVERYONE TO USE PASSWORDS THAT ARE HARD FOR HUMANS TO REMEMBER, BUT EASY FOR COMPUTERS TO GUESS.

<https://xkcd.com/936/>

# Something you have

- Phone number / **email?**
- Public / private key
- Symmetric key



***Best: On a hardware token***  
**(write+execute-only private keys)**

# Something you are

- Fingerprint
- Facial recognition
- Speech recognition
- Odour recognition
- Gait



# Biometric properties

- Not 100% accurate
  - Because of sensors
  - Because of changes in biometrics
- Not 100% applicable
  - E.g. Fingerprints w/o hands
- Typically hard to profile, easy to collect/verify
  - E.g. Scanning of face multiple times to enable FaceID on Apple

# Storing factors

# Storing passwords

- Plain text – **just don't**
- Hash(Password)
- Hash(Salt + Password)
- Hash(Salt + Password + Pepper)

# Attacks on stored passwords

- Offline
- Online
  - Rate-limit
  - Lock out after N failed attempts
  - Some cryptographic hardware devices are “online”!

# Storing keys

- Which factor is a random key?
- Storage: software vs hardware
  - Software-protected memory / files
    - `chmod 600`
    - E.g.: SSH keys, WebAuthn “passkeys”;
    - Weakest ***something you have*** factor!
  - Hardware Security Tokens / Trusted Platform Module
    - Key becomes a stronger ***something you have***!
    - Requires online attacks => rate limiting, auto-wipe after 10 failures etc.!

# Password managers

- *One ring to rule them all*
  - Master key can be derived from password
  - **Use multiple factors** (e.g., tokens)!
- Database storage: local or cloud
- Encourage different password per service:  
password generators, integration with browsers
- Back it up / don't forget/lose the keys!

CYBERSECURITY

## LastPass Hacked: Password Manager With 25 Million Users Confirms Breach

**Davey Winder** Senior Contributor   
Co-founder, Straight Talking Cyber

Follow

Aug 25, 2022, 11:08pm EDT

5 

# Password-based key derivation

- Problem: passwords have arbitrary lengths
- Cryptographic algorithms require keys of specific lengths!
  - E.g., AES-256 requires 256-bits key => 32 bytes
- Solution: Key Derivation Functions (KDF):
  - *DerivedKey = KDF(password, salt, iterations)*
- Algorithms: PBKDF2, Argon2

# FIDO2 / passkeys

- Previously: U2F: Universal Second Factor
- FIDO2 WebAuthn => asymmetric crypto!
  - Give a unique public key to the web server (no reuse!)
  - Use private key instead of password
- Private key must be stored on secure hardware!
  - FIDO-certified security keys: Yubikey, SoloKey, NitroKey ;)
  - Hardware validates 2nd factor:
    - Something you know (PIN – rate limited!)
    - Something you are (fingerprint, FaceID etc.)
- Must always enroll backup keys!



# Key establishment protocols

# Authentication protocols

- Symmetric (shared secret)
  - How to ask for a known secret over insecure channels?
    - Hash the password?
  - Challenge-Response
- **Asymmetric protocols**
  - Diffie-Hellman!
  - Forward Secrecy

# Burrows–Abadi–Needham logic (notation)

- $ID_A, ID_B, ID_S$ 
  - An unique identifier for A, B and S (Trusted Server)
- $k_{A,B}$ 
  - A key shared between A and B
- $\{ID_A\}_{K_A}$ 
  - Encryption/signature of  $ID_A$  under the key of A
- $A \rightarrow B : \{ID_A\}_{k_{A,B}}$ 
  - A send to B the message  $ID_A$  encrypted by the shared key of A and B

# Plain Diffie-Hellman

1.  $A \rightarrow B : DH\_A (g^a \bmod p)$
2.  $B \rightarrow A : DH\_B, \{ID_B\} K_{A,B}$   
*(both obtain the same  $K_{A,B}$ )*

1.  $A \rightarrow B : \{ID_A\} K_{A,B}$

*Classic MitM attack:*

1.  $A \rightarrow T : DH\_A$
  2.  $T \rightarrow B : DH\_T$
  3.  $B \rightarrow T : DH\_B, \{ID_B\} K_{T,B}$
  4.  $T \rightarrow B : \{ID_A\} K_{T,B}$
  5.  $T \rightarrow A : DH\_T, \{ID_B\} K_{A,T}$
- ... etc

# Protocol for asymmetric encryption (STS simplified)

1.  $A \rightarrow B : DH\_A$
  2.  $B \rightarrow A : DH\_B, \{ \{DH\_A, DH\_B\} pub\_A \} K_{A,B}$
  3.  $A \rightarrow B : \{ \{DH\_A, DH\_B\} pub\_B \} K_{A,B}$
- We assume each party has private/public keys
    - Public key being know to all entities
  - The problem is how to distribute public keys
    - Public Key Infrastructure
    - Pretty Good Privacy

# Symmetric authentication

- Given A and B who trust S, A and B should be able to create a shared key  $k_{A,B}$  for secret communication
- $k_{A,B}$  should be known only to A and B (and possibly to S)
- A and B should know that  $k_{A,B}$  is newly generated
- A and B should authenticate each other
- Why? **Enterprise authentication!**

# Protocol for symmetric encryption (1)

## Protocol steps

1.  $A \rightarrow S : ID_A, ID_B$   
*I am A, give me key for B*
1.  $S \rightarrow A : k_{A,B}$   
*(key get transferred in unencrypted form)*
1.  $A \rightarrow B : ID_A, k_{A,B}$   
*(key get transferred in unencrypted form)*

## Possible attacks

- An attacker with MITM capabilities gets  $k_{A,B}$

# Protocol for symmetric encryption (2)

## Protocol steps

1.  $A \rightarrow S : ID_A, ID_B$   
*I am A, give me key for B*
1.  $S \rightarrow A : \{k_{A,B}\} k_{A,S}, \{k_{A,B}\} k_{B,S}$   
*(key encrypted with common secret between A,S)*
1.  $A \rightarrow B : ID_A, \{k_{A,B}\} k_{B,S}$   
*(key also encrypted with common B,S secret)*

## Possible attacks (1)

1.  $A \rightarrow S : ID_A, ID_B$
2.  $S \rightarrow A : \{k_{A,B}\} k_{A,S}, \{k_{A,B}\} k_{B,S}$
3.  $A \rightarrow T : ID_A, \{k_{A,B}\} k_{B,S}$
4.  $T \rightarrow B : ID_T, \{k_{A,B}\} k_{B,S}$   
*(Trudy replaces the identity A presented to B!)*



# Protocol for symmetric encryption (2)

## Protocol steps

1.  $A \rightarrow S : ID_A, ID_B$
2.  $S \rightarrow A : \{k_{A,B}\} k_{A,S}, \{k_{A,B}\} k_{B,S}$
3.  $A \rightarrow B : ID_A, \{k_{A,B}\} k_{B,S}$

## Possible attacks (2)

1.  $A \rightarrow T : ID_A, ID_B$
2.  $T \rightarrow S : ID_A, ID_T$
3.  $S \rightarrow T : \{k_{A,T}\} k_{A,S}, \{k_{A,T}\} k_{T,S}$
4.  $T \rightarrow A : \{k_{A,T}\} k_{A,S}, \{k_{A,T}\} k_{T,S}$
5.  $A \rightarrow T : ID_A, \{k_{A,T}\} k_{T,S}$

*Trudy in the middle...*

# Protocol for symmetric encryption (3)

## Protocol steps

1.  $A \rightarrow S : ID_A, ID_B$
2.  $S \rightarrow A : \{k_{A,B}, ID_B\}_{k_{A,S}}, \{k_{A,B}, ID_A\}_{k_{B,S}}$
3.  $A \rightarrow B : \{k_{A,B}, ID_A\}_{k_{B,S}}$

## Possible attacks

*Replay of old broken key*

1.  $A \rightarrow T : ID_A, ID_B$
2.  $T \rightarrow A : \{k'_{A,B}, ID_B\}_{k_{A,S}}, \{k'_{A,B}, ID_A\}_{k_{B,S}}$
3.  $A \rightarrow B : ID_A, \{k_{A,B}, ID_A\}_{k_{B,S}}$

# Protocol (4) - Needham-Schroeder (1978)

## Protocol steps

1.  $A \rightarrow S : ID_A, ID_B, \text{Nonce}_A$
2.  $S \rightarrow A : \{k_{A,B}, ID_B, N_A, \{k_{A,B}, ID_A\} k_{B,S}\} k_{A,S}$
3.  $A \rightarrow B : \{k_{A,B}, ID_A\} k_{B,S}$
4.  $B \rightarrow A : \{N_B\} k_{A,B}$
5.  $A \rightarrow B : \{N_B - 1\} k_{A,B}$

## Possible attacks - Denning Sacco

*Replay of old broken key*

1.  $T \rightarrow B : \{k'_{A,B}, ID_A\} k_{B,S}$
2.  $B \rightarrow T : \{N_B\} k'_{A,B}$
3.  $T \rightarrow B : \{N_B - 1\} k'_{A,B}$

# Protocol for symmetric encryption (5)

## Protocol steps

1.  $B \rightarrow A : ID_B, N_B$
2.  $A \rightarrow S : ID_A, ID_B, N_A, N_B$
3.  $S \rightarrow A : \{k_{A,B}, ID_B, N_A\}_{k_{A,S}}, \{k_{A,B}, ID_A, N_B\}_{k_{B,S}}$
4.  $A \rightarrow B : \{k_{A,B}, ID_A, N_B\}_{k_{B,S}}$

## Possible attacks

- None of the above

# Notes on protocols - Abadi and Needham [2]

- If the identity of a principal is essential to the meaning of a message, it is prudent to mention the principal's name explicitly in the message.
- Be clear about why encryption is being done.
- When a principal signs material that has already been encrypted, it should not be inferred that the principal knows the content of the message.
- Be clear about what properties you are assuming about nonces.
- If timestamps are used as freshness guarantees, then the difference between local clocks at various machines must be much less than the allowable age of a message.

# Digital Identity

## European Digital Identity

### PAGE CONTENTS

Digital Identity for all Europeans

Benefits of the European Digital Identity

Why is it needed?

Key principles

Practical use

Making things easier for citizens and businesses

Documents

## Digital Identity for all Europeans



The European Digital Identity will be available to EU citizens, residents, and businesses who want to identify themselves or provide confirmation of certain personal information. It can be used for both online and offline public and private services across the EU.

Every EU citizen and resident in the Union will be able to use a personal digital wallet.



*"Every time an App or website asks us to create a new digital identity or to easily log on via a big platform, we have no idea what happens to our data in reality. That is why the Commission will propose a secure European e-identity. One that we trust and that any citizen can use anywhere in Europe to do anything from paying your taxes to renting a bicycle. A technology where we can control ourselves what data is used and how."*

Ursula von der Leyen, President of the European Commission, in her State of the Union address, 16 September 2020

## Digital Identity Guidelines

The four-volume SP 800-63 *Digital Identity Guidelines* document suite is available in both PDF format and online.

PDF versions of the documents are available from:

Document	Title	URL
SP 800-63-3	Digital Identity Guidelines	<a href="https://doi.org/10.6028/NIST.SP.800-63-3">https://doi.org/10.6028/NIST.SP.800-63-3</a>
SP 800-63A	Enrollment and Identity Proofing	<a href="https://doi.org/10.6028/NIST.SP.800-63a">https://doi.org/10.6028/NIST.SP.800-63a</a>
SP 800-63B	Authentication and Lifecycle Management	<a href="https://doi.org/10.6028/NIST.SP.800-63b">https://doi.org/10.6028/NIST.SP.800-63b</a>
SP 800-63C	Federation and Assertions	<a href="https://doi.org/10.6028/NIST.SP.800-63c">https://doi.org/10.6028/NIST.SP.800-63c</a>

Links to the online version of the SP 800-63 suite are below.



**SP 800-63-3**  
Digital Identity Guidelines



**SP 800-63A**  
Enrollment & Identity Proofing



**SP 800-63B**  
Authentication & Lifecycle Management



**SP 800-63C**  
Federation & Assertions

# Single Sign-On (SSO)

- Password managers
- Enterprise level SSO
  - Same-domain authentication
  - Kerberos, RADIUS with LDAP / Active Directory databases
- Federated Identity
  - Cross-domain authentication
  - Based on assertions containing the result of authentication
    - Factors cannot be shared between domains
  - RADIUS, OpenID Connect, SAML etc.

# Kerberos

- Developed by MIT in 1983
  - Was banned for export till 2000 by US
- Used for key establishment between multiple entities
  - The Kerberos server is trusted by all entities
  - Assumes existing pre-shared keys between entities and Kerberos server
- Can be adapted to multiple symmetric encryption algorithms
  - Kerberos v5 uses AES



# Kerberos

- The adversary can compromise the network, not the host (e.g. secrets, keys)
- Based on fixed Needham–Schroeder protocol
- Uses tickets to create a legitimate session key

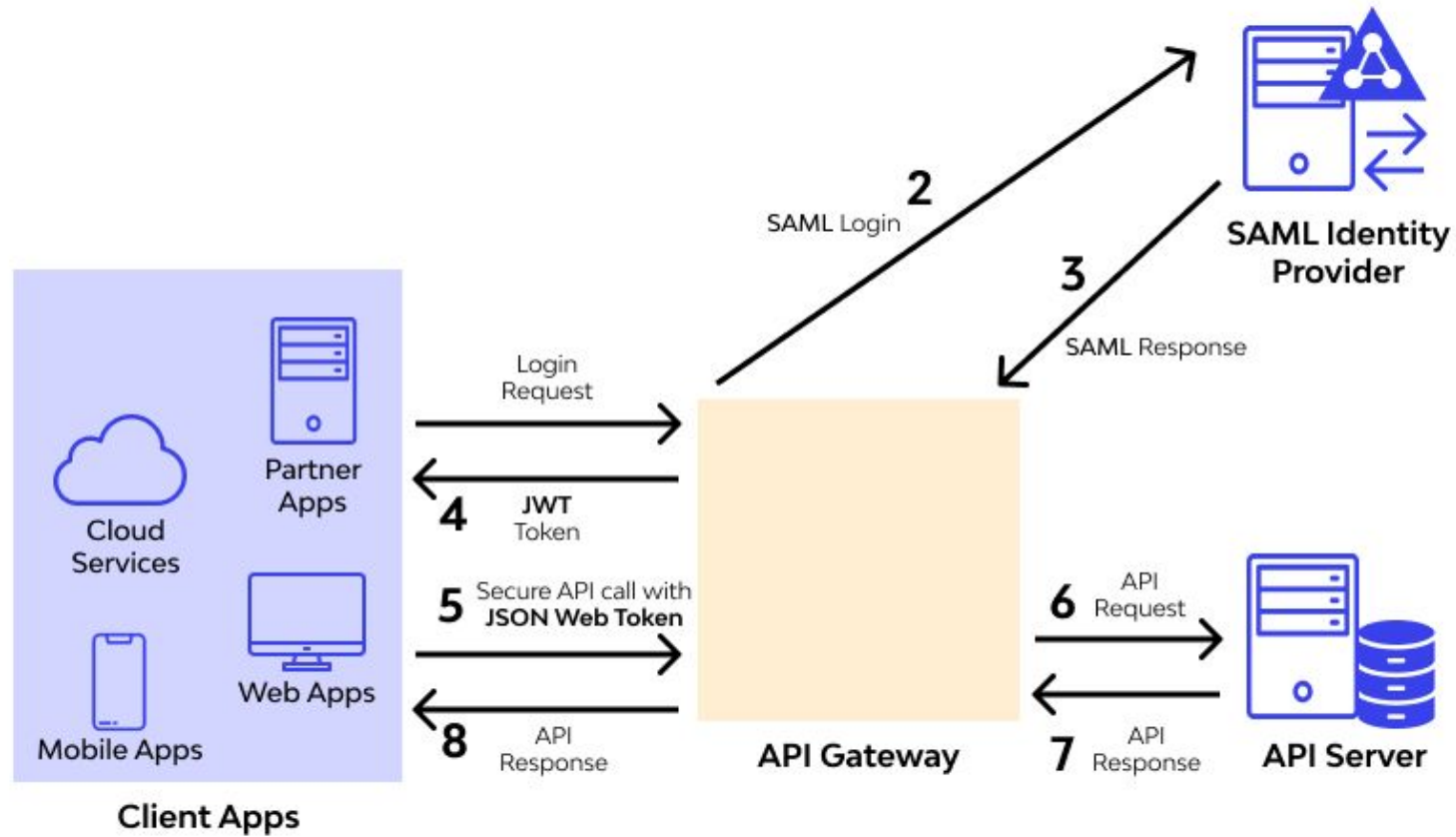
# Kerberos (v1)

1.  $A \rightarrow KAS : ID_A, ID_B, N_A$ 
  - KAS = Kerberos Authentication Server
2.  $KAS \rightarrow A : \{k_{A,B}, ID_B, T_{KAS}, N_A\} k_{A,KAS}, \{k_{A,B}, ID_A, T_s\} k_{B,KAS}$ 
  - $T_s$  = Timestamp server
3.  $A \rightarrow B : \{k_{A,B}, ID_A, T_{KAS}\} k_{B,KAS}, \{ID_A, T_a\} k_{A,B}$ 
  - $k_{A,B}$  : session key between A and B
  - $\{k_{A,B}, ID_A, T_{KAS}\} k_{B,KAS}$  : ticket for A to used when contacting B
  - $T_a > T_{KAS}$  : B needs to validate time window by comparing  $T_a$  and  $T_{KAS}$

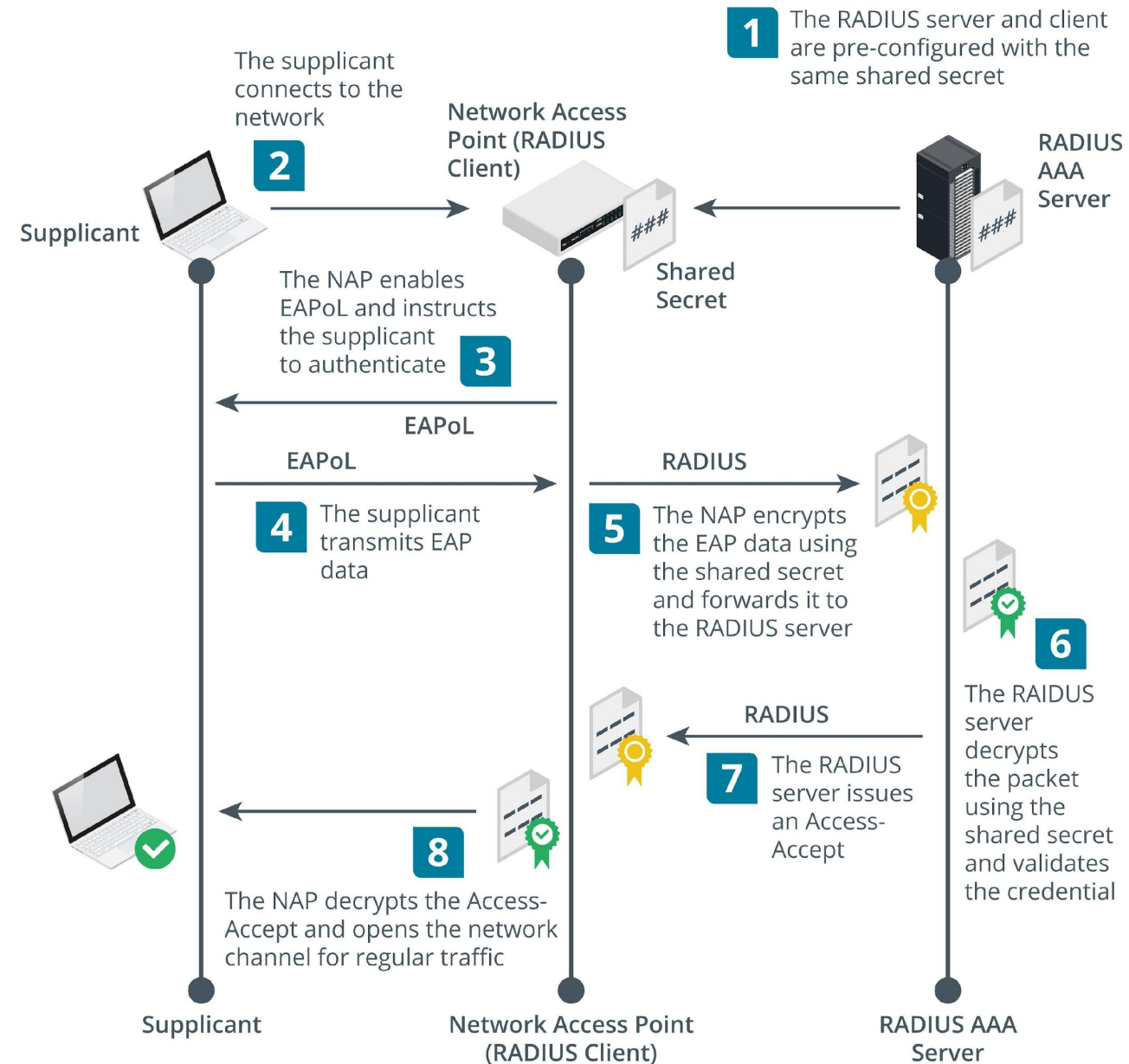
# Kerberos (v2)

- Usually the shared key between A and KAS is deducted from a user input/password
- The previous version of the protocol requires credentials input every connection to a new application server (B)
- Ticket granting separate from user authentication
  - User authenticates using passwords with KAS and receives session key for Ticket Granting Service
  - Entities use session key to require tickets for multiple applications

# SAML



# RADIUS / EAP



# References

- [1] “Protocols for Authentication and Key Establishment”, Colin Boyd, Anish Mathuria, Douglas Stebila, 2020
- [2] Abadi, M., Needham, R.: Prudent engineering practice for cryptographic protocols. In: IEEE Symposium on Research in Security and Privacy, pp. 122–136. IEEE Computer Society Press (1994)
- [3] Computer Security and the Internet: Tools and Jewels, Paul C. van Oorschot. Springer, 2021.  
<https://people.scs.carleton.ca/~paulv/toolsjewels.html>