Authentication: an introduction

Scenario

- Alice needs to show her identity to Bob; she needs to get a service, or to access information, or a resource etc
- Bob needs to be sure of Alice's identity
  - Who is Alice?
- Trudy tries to impersonate Alice
  - Once
  - Many times

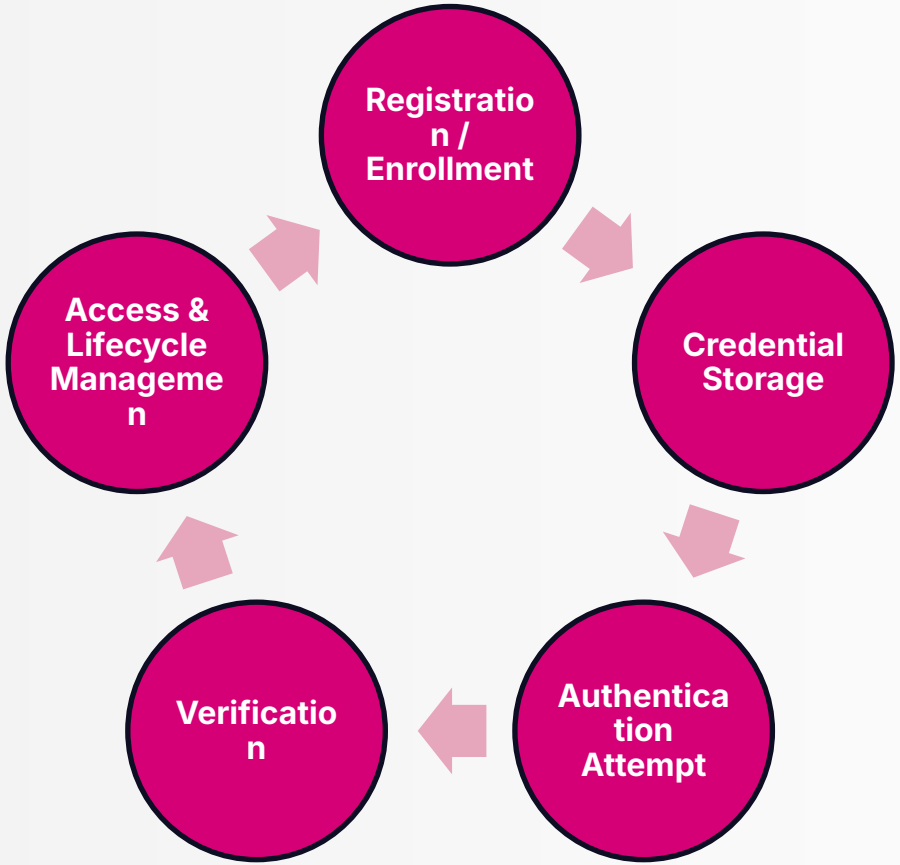
Definition of authentication

- Authentication is the process of establishing and verifying that a subject attempting to access a system is the same subject who originally registered with it
- It is performed by validating one or more authentication factors against trusted reference data captured during registration
- Authentication increases assurance that the claimed identity is valid, but it does not provide absolute or fully legal identification
    - absolute identification $\Longrightarrow$ authentication

In other words...

- Authentication is the process of reliably verifying the identity of someone (or something)
  - Not legally
- In human interaction
  - People who know you can recognize you based on your appearance or voice
  - A guard might authenticate you by comparing you with the picture on your badge
  - A mail order company might accept as authentication the fact that you know the expiration date on your credit card
- Two interesting cases
  - a computer is authenticating another computer
  - a person is using a public workstation

Cycle of life of authentication



More structured view

- **Registration / enrollment**
  - Subject creates credentials or registers suitable information
- **Credential storage**
  - System securely stores authentication data (hashed, encrypted, or in secure hardware)
- **Loop**
  - Authentication attempt
    - Subject presents credentials
  - Verification
    - System validates credentials against stored reference
  - Access & lifecycle management
    - Grant or deny access, manage changes, renewals, revocation etc.

Closed world assumption

Presumption that what is not currently known to be true, is false

The opposite of the closed world assumption is the open world assumption, stating that lack of knowledge does not imply falsity

Negation as failure is related to closed world assumption, as it believes false every predicate that cannot be proved to be true

A particular case: the closed environment

- Authentication in a company, home, etc.
- We often use a third-party Carole (trusted server) distributing required information for authentication (before authentication or using secret communication)
- Trudy is a legal user and might use the connection Alice-Bob

What's the difference between authenticating a human and authenticating a computer?

- computer can store a high-quality secret, such as a long random-looking number, and it can do cryptographic operations
- a workstation can do cryptographic operations on behalf of the user, but the system must be designed so that all the person must remember is a password
- password can be used to acquire a cryptographic key in various ways
    - directly, as in doing a hash of the password
    - using the password to decrypt a higher-quality key, such as an RSA private key, that is stored in some place like a directory service

Authentication of people

- Use
  - What you know (passwords)
  - What you have (smart card)
  - Who you are (biometric tools)
  - Where you are (network address - weak, because of spoofing)
- These are authentication factors

Passwords

- Humans
  - Short keys, that possibly allow to obtain long keys
  - Sometimes easy to guess
- Computers
  - Long keys
  - Hidden (not stored in clear): either stored encrypted or indirectly
- One-time password
- A well-known threat is the login Trojan horse: the attacker prompts a fake login window that induces the user to prompt the password that is collected by the attacker

Login Trojan

Biometric

- There are many devices that are used to authenticate
  - Retina examination, fingerprint reader, voice, or based on other characteristics (ex. handwritten signatures, timing in using the keyboard or the mouse etc.)
- Accuracy
  - Errors: false positive and false negative
  - They are used in specific scenarios (sometimes to enforce other methods)
  - In last years accuracy improved

Biometric examples

Fingerprinting

- Fake fingerprint, dead people
- Not stable over time: children, people doing manual work (possible damage)

Voice

- Use of tape
- Voices are affected by flu and colds, noise in the background, telephone etc.

Biometric examples (continued)

- Keystroke timing
  - Each person has different speed in typing
  - Typing faster than personal limit is impossible
- Handwritten signature
  - Poor quality
  - Electronic tablet, for signature + timing
    - Largely used in banks

Authentication: attacker model

Why model attackers?

- Clarify protection scope
  - Define who the system is defending against
  - Focus resources on realistic threats, not edge cases
- Improve protocol design
  - Select authentication methods that address the right risks
  - Avoid unnecessary complexity for low-risk scenarios
- Guide mitigation strategy
  - Prioritize controls for the most probable and damaging attacks

Attacker dimensions

- **Motivation**
  - Financial gain (fraud, theft)
  - Espionage (corporate, political)
  - Disruption or sabotage
  - Ideological or personal revenge
- **Resources**
  - Computational power (CPU/GPU clusters, botnets)
  - Time and patience for long-term campaigns
  - Budget for buying exploits or stolen data
- **Skills**
  - Low-skill “script kiddies” using pre-made tools
  - Skilled professionals or Advanced Persistent Threats (APTs)
- **Access level**
  - Physical access to devices or networks
  - Local network access
  - Remote internet-based access
- **Persistence**

Capability levels

- Passive adversary
  - Observes network traffic without altering it
  - Collects metadata (timestamps, IPs) to infer behaviour
- Active adversary
  - Injects, modifies, or deletes authentication messages
  - Replays captured valid credentials
- Insider threat
  - Has legitimate credentials or privileged access
  - Can bypass many external defenses
  - May collude with external attackers

Core attack vectors

- Credential theft
  - Phishing campaigns targeting users directly
  - Malware or keyloggers capturing keystrokes
  - Theft of stored credentials from compromised devices
  - Reusing username/password pairs from past breaches
- Brute force and guessing
  - Online attempts with rate limits to bypass
  - Offline cracking of stolen hashed password databases
- Replay attacks
  - Capturing valid authentication data and resending it
  - Exploiting lack of session freshness checks
- Man-in-the-Middle (MITM)
  - Intercepting and modifying live authentication sessions
  - Using forged certificates or compromised routers

Specialized threats

- Biometric spoofing
  - Creating artificial fingerprints or facial models
  - Using photos, videos, or 3D masks
- Token cloning
  - Duplicating physical smartcards or copying OTP seeds
  - Exploiting weak hardware protection on tokens
- Session hijacking
  - Stealing session cookies or tokens via malware or XSS
  - Using them to bypass re-authentication
- Social engineering
  - Impersonating legitimate users to IT support
  - Coercion or tricking users into revealing credentials

Example scenarios

- Untrusted Wi-Fi hotspot
  - Passive attacker captures unencrypted traffic
  - Active attacker injects fake login pages
- Corporate insider abuse
  - Admin accesses confidential areas without authorization checks
  - Manipulates audit logs to hide traces
- Compromised mobile device
  - Malware bypasses biometric authentication by replaying stored data
  - User unaware device is acting as a proxy for an attacker

Mapping threats to defenses

- MITM
  - Mutual authentication between client & server
  - TLS with certificate pinning
- Replay
  - Use of nonces and timestamps in authentication messages
  - One-time passwords (OTP) and short-lived tokens
- Credential theft
  - Multi-factor authentication (MFA)
  - Hardware tokens or secure mobile apps
- Brute force
  - Rate limiting and account lockouts
  - Salted, iterated password hashing (e.g., bcrypt, Argon2)
- Biometric spoofing
  - Liveness detection and challenge-response biometrics

Realistic assumptions

- Networks are not trusted
  - Assume traffic can be monitored by unauthorized parties
- Endpoints may be compromised
  - Include malware, rootkits, or misconfigured devices
- Users are fallible
  - Susceptible to phishing and social engineering
- Implementations are imperfect
  - Bugs, weak randomness, and misconfigurations are realistic risks

Security design

- Attacker models are context-specific
    - One size does not fit all
- Define “who, what, how” before designing defences
    - Understand the attacker’s capabilities and goals
- Update models regularly
    - Threat landscapes evolve over time
- Authentication security = protocol soundness + endpoint security + human factor resilience

Authentication: scenarios & techniques

Authentication scenarios

In the following we will consider authentication using knowledge of a key (public key or secret key, possibly stored in a smart card)

1. Users (Alice and Bob) share a secret key
2. Users share a key with Carole, trusted authority (authentication server)
3. Users have a public key

Techniques

Users are authenticated using a key

Techniques

- timestamp
- nonce (or challenge): random number chosen by the person that authenticates to verify the other knows the key
- sequence number in protocols

Use of keys

- Keys must be kept as a secret
    - Except public keys
- Keys can (should) be ephemeral (dynamic) but also master keys (static)
    - Different consequences in case of compromission
- Quality key = random-looking sequence of bits
    - The longer, the better (compromises needed)
- Keys can be generated by suitable KDF or locally created and then securely exchanged
    - But DH-like techniques are different

Timestamps

- They are strings that mark current time
  - Date, time of day, up to milliseconds
  - Weakness: clock should be synchronized
- They have a validity interval
  - There is a validity period that considers network problems and limitations
  - Weakness: a fast adversary can replay a timestamped packet with the interval
- TSAs: timestamping authorities
  - They produce trusted timestamps and signs them
  - In this case it is a longer string
    - Cryptographic hash of the timestamped information
    - Timestamp
    - Signature (using public key cryptography)

Nonces

- Random numbers to be used once
    - Good CS-PRNG needed
- Typically needed for creating fresh challenges
- They are not permanently stored
    - Too many
- Often coupled to timestamps
    - They are temporarily stored during validity of timestamps to prevent fast replay
- They are sometimes used alone

Sequence numbers

- Inspired to sequence numbers of TCP protocol
- They create long sequences of consecutive data, hard to be forged
- They typically have a max value
    - After this value security parameters are often re-negotiated

Use of the techniques

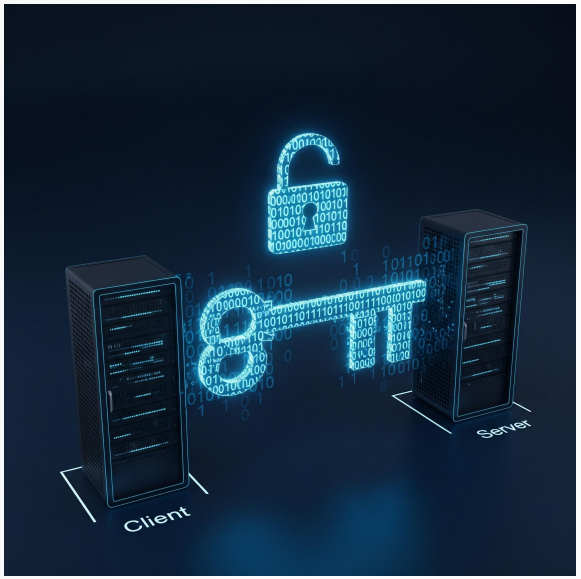
The presented techniques

- Aren't necessarily all used at the same time
- They provide good basis for security requirements, but they may need additional measures
- Their smart combination may produce widely good results
- Are orthogonal to the type/factor of authentication

Encryption vs. authentication

- Trudy attacks the protocols using non authentic messages (possibly messages that have been exchanged between Alice and Bob in previous application of the protocol)
- Need to guarantee authentication and integrity
- **Encryption DOES NOT guarantee authenticity of the message**

Authentication based on symmetric keys



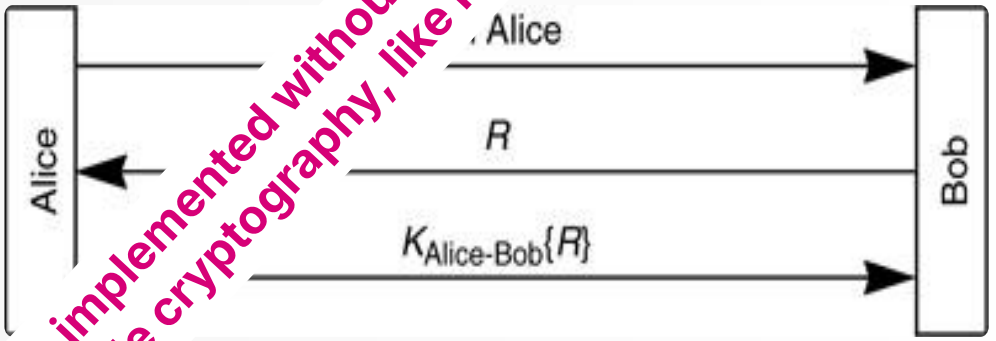
Authentication by symmetric key

- Alice and Bob share a secret key $K_{Alice\text{-}Bob}$
- K{M} denotes M encrypted with secret key K
- One-way authentication using nonce (challenge)
- One-way authentication using timestamps
- Two-ways (mutual) authentication using nonce

1- or 2-wayness

- 1-way: one part is authenticated to the other part
- 2-ways: each is authenticated to the authentication partner
    - It is obvious that a 2-ways auth. can be implemented as two 1-way auth. in opposite directions
    - A better and more robust construction is the near-simultaneous authentication

Challenge/response based on shared secret



can be implemented without a reversible cryptography, like HMAC

- Authentication is not mutual. Bob authenticates Alice, but Alice does not authenticate Bob
- an eavesdropper could mount an off-line password-guessing attack (assuming $K_{Alice-Bob}$ is derived from a password), KPA - such an attack allows to guess the key
- R is a nonce

Challenge-response symmetric key

How to guarantee data integrity

- timestamps (a message is valid only in a small time-window)
- sequence number (A and B remember sequence number of exchanged messages to avoid replay attacks in which the attacker sends old messages)
- nonce should be used carefully

Relay attack

1. T(A) → B: "I am Alice"
2. B → T(A): R
3. A → T(B): "I am Alice" *(T tricks A into opening a connection)*
4. T(B) → A: R *(nonce from B)*
5. A → T(B): K{R}
6. T(A) → B: K{R} *(T forwards to B)*

Result: B accepts the proof that T is Alice, even though T never knew K

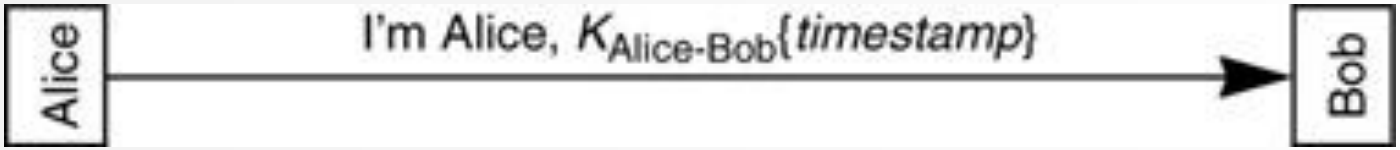
Preventing relay attacks

- **Mutual authentication**: Alice must first confirm she's talking to the real Bob before sending K{·}
- **Channel binding**: bind K{·} to something T can't forward
- **Out-of-band verification** (physical proximity, secure device pairing, etc.)

Challenge/response variant

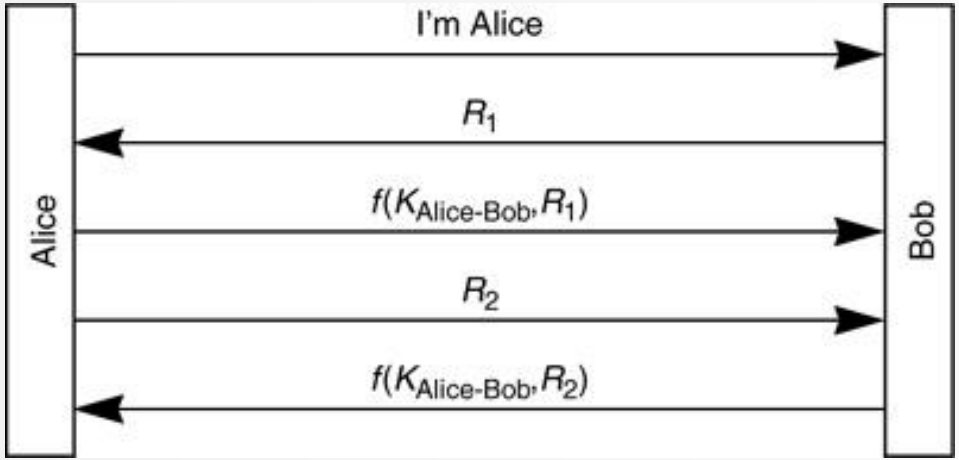
- It requires reversible cryptography
- Still possible the dictionary attack by eavesdropper
- If R has limited lifetime (e.g., random number + timestamp) Alice authenticates Bob because only someone knowing $K_{Alice-Bob}$ could generate $K_{Alice-Bob}\{R\}$
  - Limited lifetime required to foil replaying of an old $K_{Alice-Bob}\{R\}$

Timestamp based



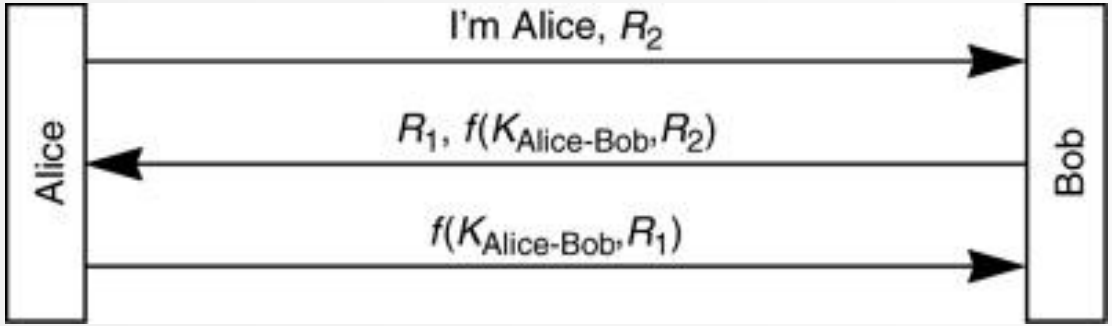
- Bob and Alice have reasonably synchronized clocks (can be a weakness)
- Alice encrypts the current time, Bob decrypts the result and makes sure the result is acceptable (i.e., within an acceptable clock skew)
- Efficient, no intermediate states
- If Bob remembers timestamps until they expire, then no replaying attacks
- if multiple servers with same secret K, then Alice can send K{Bob||timestamp}

Mutual authentication



- many messages!
- perhaps a few ones can be saved...

Optimized mutual authentication

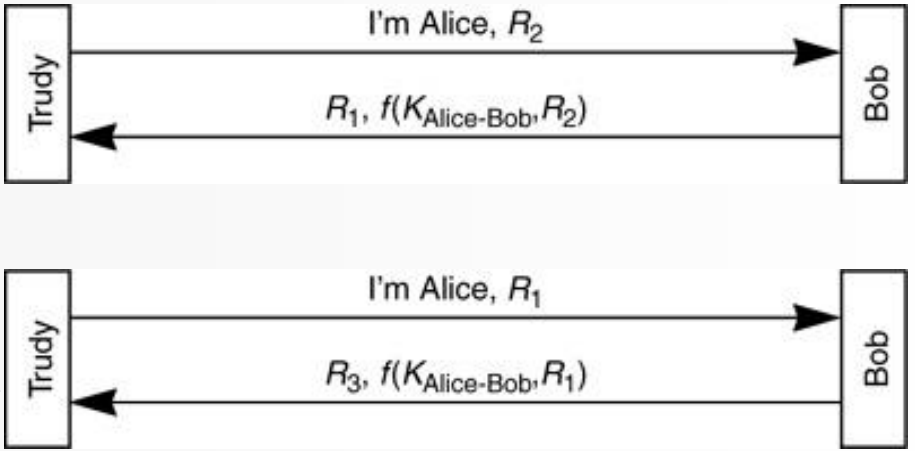


save messages but weak to reflection attack

f can be a HMAC

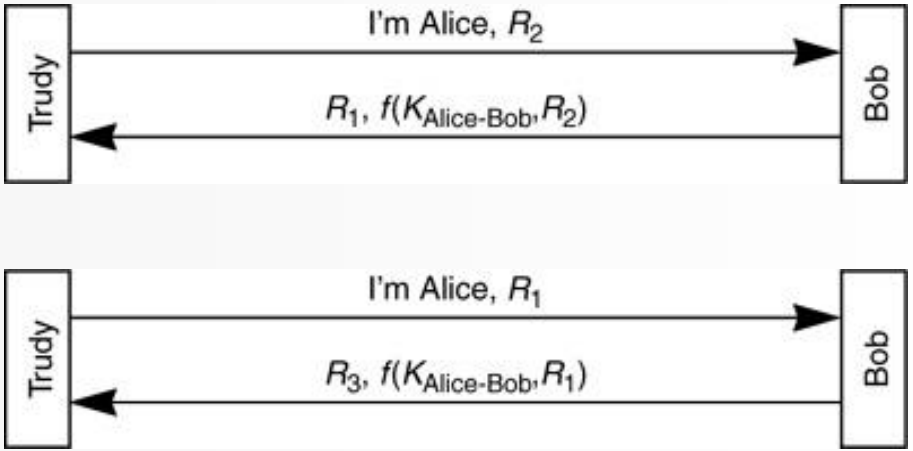
Reflection attack

- Trudy wants to impersonate Alice to Bob
- Two sessions, the 2nd will be incomplete but will allow Trudy to complete the 1st one



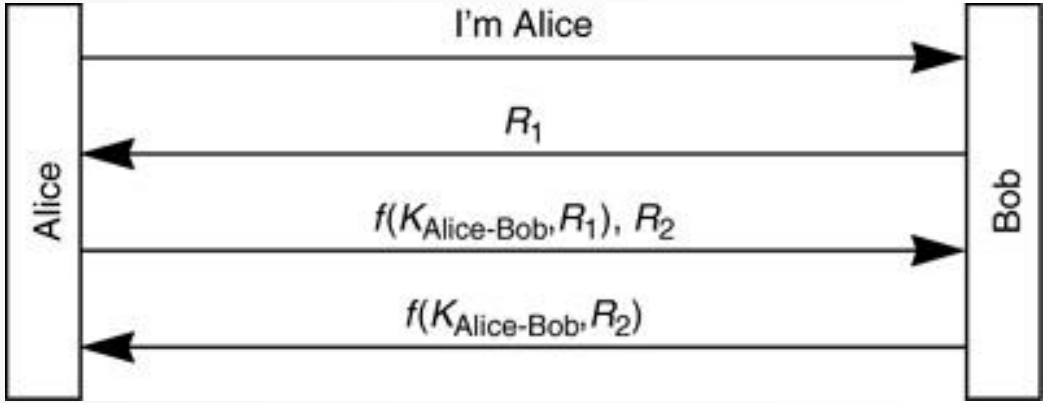
How to prevent reflection attack?

- use different keys
  - $K_{Alice\text{-}Bob}$ and $-K_{Alice\text{-}Bob}$, or $K_{Alice\text{-}Bob}+1$
- different challenges, i.e., challenge from initiator different from challenge from responder
  - e.g., even number at initiator's side, odd number at responder's side

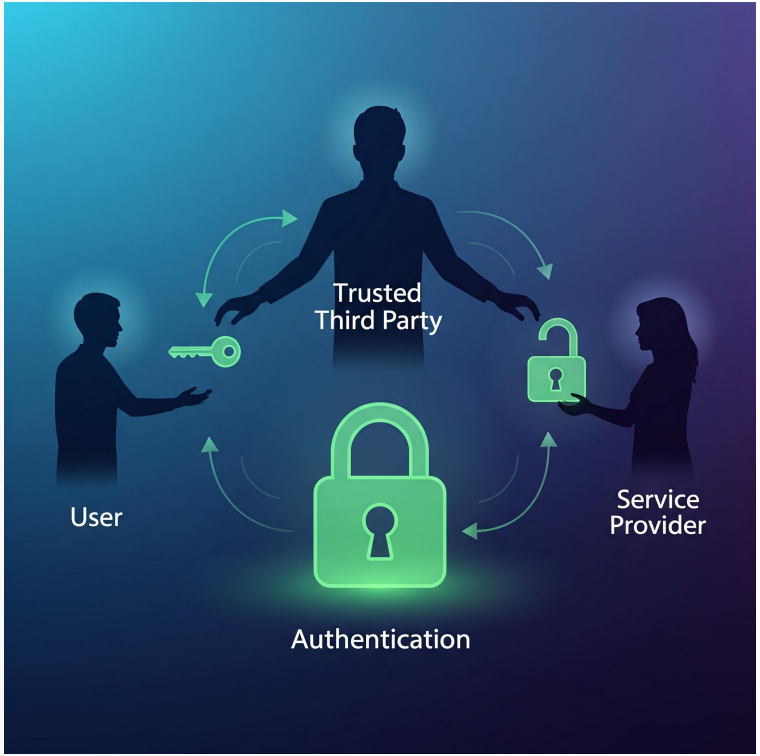


Less optimized mutual authentication

Trudy can mount an off-line password-guessing attack (KPA) by impersonating Bob's address and tricking Alice into attempting a connection to her



Authentication based on a trusted third party



Base idea

- Method in which two entities rely on an **intermediary**, **trusted by both**, to verify their identities
- The third party issues cryptographic credentials that serve as proof of identity, enabling secure communication without prior direct trust between the entities
- **Advantage**: simplifies trust management, **avoids many pairwise trust links**
- **Risk**: single point of failure if the third party is compromised or unavailable

Trusted party

- It is not possible that all users share a secret key (quadratic number of keys, each user must have a different key for everyone)
- **Scenario**
  - users share a key with trusted authority C
  - **C authentication server** (aka Key Distribution Center (KDC))
  - A and B share a secret key with C (K_{AC} e K_{BC})
  - A and B might not be human users but also entity of the system (e.g., printer, databases etc.)
- **Goals**
  - authenticate A (or A and B)
  - optional: decide on a session key

Authentication server

Goal: Alice and Bob must authenticate and choose a secret session key K - used for short time (session)

At the end of the authentication protocol

1. Only A and B know K (besides C - trusted server)
2. Each should be sure of fact 1 above
3. K is a new key (randomly chosen, not used before)

Attacker's strength

Trudy (attacker) can

- be a legitimate user of the system (share a key with C)
- sniff and spoof messages
- concurrently run more than one session with A, B and C
  - different execution of the protocol can be done interleaved
  - T can convince A and/or B to start a new session with T
- might know old session keys

Attacker's limits

Trudy

- **cannot guess** random numbers chosen by A or B
- does not know keys K_{AC} and K_{BC} (in general does not know secret keys of other users)
- **cannot decrypt** in a short time messages encrypted with unknown keys

Authentication with trusted server

A and B share a key with C (K_{AC} and K_{BC}, respectively)

1. A sends to C: (A,B)
2. C chooses K - session key - and sends to A:
   (K_{AC}(K),K_{BC}(K))
3. A decrypts, computes K and sends to B:
   (C,A,K_{BC}(K))
4. B decrypts K_{BC}(K), finds K and sends to A:
   K("Hello A","this is B")

Attack 1

- Assume T can sniff, spoof and make MITM attack (T is in the middle of both A to C and A to B communication)
- A sends to T (instead of A sends to C) : (A,B)
- Immediately T sends to C: (A,T)
- C chooses K and sends to A: (K_{AC}(K),K_{TC}(K))
- A sends to B: (C,A,K_{TC}(K)) - T intercepts
- T sends to A: K("Hello A","this is B")
- Possible modification of step 1 (previous slide):
  - A sends to C: (A,K_{AC}(B))
  - in this way C still knows that A wants to talk to B

Modified protocol

- A sends to C: $(A,K_{AC}(B))$
- C chooses K, and sends to A: $(K_{AC}(K),K_{BC}(K))$
- A decrypts K and sends to B: $(C,A,K_{BC}(K))$
- B decrypts K and sends to A: K("Hello A","this is B")

- Note: T does not know that A wants to talk to B

Attack 2

Consider modified protocol

1. A sends to C : (A,K_{AC}(B))
   T gets A's message and sends to C (as A): (A,K_{AC}(T))
   (**REPLAYS** part of some previously exchanged message)
   Note: T does not know whom A wants to talk to
2. C chooses K, and sends to A: (K_{AC}(K),K_{TC}(K))
3. A decrypts K and sends to B : (C,A,K_{TC}(K))
4. T gets the message and finds that A wants to talk to B; T now acts in place of B and sends to A: K("Hello A","this is B")

Note: T knows the identity the person A wants to talk to only at the end of the protocol

Protocol Needham-Schroeder

basis for the Kerberos protocol and aims to establish a session key between two parties on a network, typically to protect further communication

1. A chooses N (nonce) and sends to C: (A,B,N)
2. C chooses K and sends to A: $K_{AC}(N,K,B,K_{BC}(K,A))$
3. A decrypts, checks N and B, and sends to B: $K_{BC}(K, A)$
4. B decrypts, chooses nonce N′ and sends to A: K(N′)
5. A sends to B: K(N′ - 1) *(now B has checked A)*

Note: B does not directly communicate with C

Attacking N.–S. protocol

1. A chooses N and sends to C: (A,B,N)
2. C chooses K and sends to A: $K_{AC}(N,K,B,K_{BC}(K, A))$
3. A decrypts, checks N and B and sends to B: $K_{BC}(K,A)$
   T intercepts and (as A) **replays** to B: $K_{BC}(K',A)$, where K′ is an older compromised session key (it is an old message)
4. B decrypts, chooses nonce N′ and sends to T (not to A): K′(N′)
5. T sends to B: K'(N′-1)

Note:

1. T uses old compromised session key and acts in place of A
2. B is not sure that C is active: there is no direct exchange between B and C

Attack clarification

- It's an attack to authentication (Dennis, Sacco 1981)
- two sessions
  - assume that Trudy has recorded session 1 and that K' is compromised
  - after session 2, B is convinced that they share the secret key K' only with A

session 1, session 2

1. A → C: (A, B, N)
2. C → A: K_{AC}(N, B, K', K_{BC}(K', A))
3. A → T(B): K_{BC}(K', A) *T pretends to be B*

assume that K' is compromised

4. T(A) → B: K_{BC}(K', A) **replay**
5. B → T(A): K'(N')
6. T(A) → B: K'(N' - 1)

B is now convinced that he shares secret key K' only with A

Needham-Schroeder protocol (variant)

Modified Challenge-Response (nonce and timestamp)

- C exchanges messages with both A and B
- timestamp t used only between B and C
- K session secret key

1. A chooses N and sends to B: (A, N)
2. B chooses N′ and sends to C: (B, N′, K_{BC}(N, A, t))
3. C sends to A: (K_{AC}(B, N, K, t) , K_{BC}(A, K, t), N′)
4. A sends to B: (K_{BC}(A, K, t), K(N′))

Basis for the Kerberos protocol

Expanded Needham-Schroeder