Chair for Network Architectures and Services
Department of Informatics
TU München – Prof. Carle

Peer-to-Peer Systems and Security IN2194

Chapter 2 Security
2.1 Basics and Fundamental Issues

Dr. Heiko Niedermayer
Christian Grothoff, PhD
Prof. Dr.-Ing. Georg Carle

Motivation – Why do we need security?

Motivation

- Should someone else be able to read what you write?
- Should someone else be able to pretend he is you?

- Security addresses many facets of allowing only the right things to happen – even in the presence of non-cooperative or malicious entities (attackers).

Overview

- Security Basics
  - Security Goals
  - Cryptography
  - Open vs Closed Systems
- Authentication
  - Authentication Protocols
  - Boyd‘s Theorem
  - Other methods
- Key Distribution
- Trust and Reputation
  - Trust in Key Distribution
  - Reputation of / Trust in an entity (e.g. associated with a key)

Security Basics

Security Basics – Some Security Goals

Confidentiality

- Only the designated receivers are able to read the message.

Integrity

- Message cannot be modified without the receiver being able to detect it.

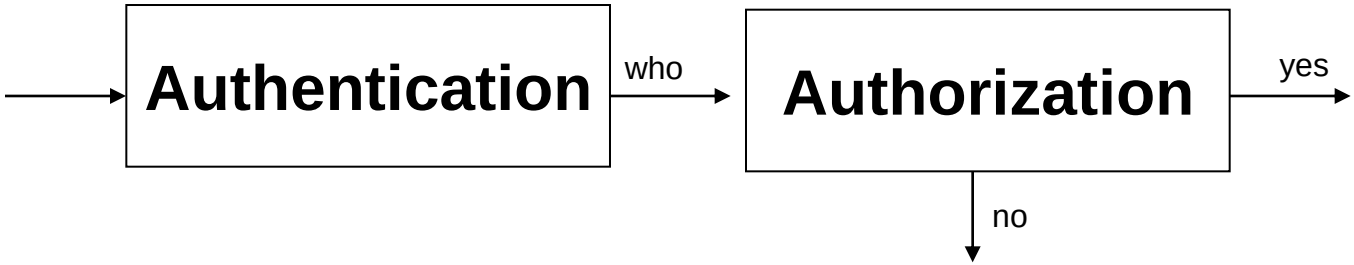
Authentication

- An entity proves its identity to other entities.
- Mutual authentication
  - A and B both prove their identity to each other.
- We may believe we have an intuitive understanding of the meaning of authentication – but the term is actually very difficult to define.
  - Give it a try, if you like.
  - See, e.g., the work of Menezes et al. and Lowe

Security Basics – Security Goals

Authentication vs. Authorization

- Authentication: an entity proves its identity
- Authorization: decide whether an entity is allowed to perform a certain action.

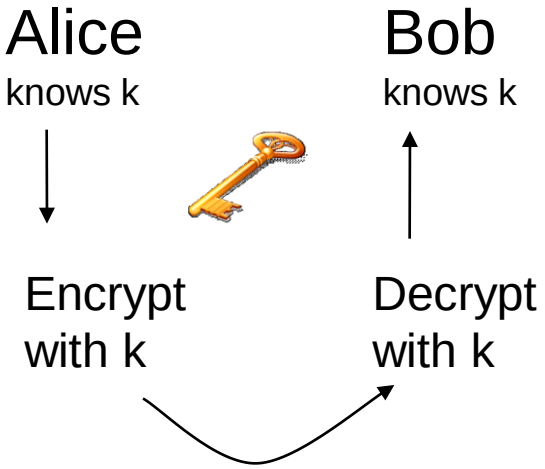


- Authentication is a pre-requisite for most other security goals!

Security Basics – Symmetric Cryptography

Symmetric cryptography

- A and B share a **common key** K
- Symmetric ciphers provide two functions
  - „encrypt“: `cdata = enc(k,data)`
  - „decrypt“: `data = dec(k,cdata)`
- The advantage of symmetric cryptography is that it is comparatively *fast*.
- Well-known ciphers are, e.g.:
  - AES: Advanced Encryption Standard
  - Twofish (B. Schneier)
- Today key length >= 128 bit recommended
  - Else brute-force attacks feasible

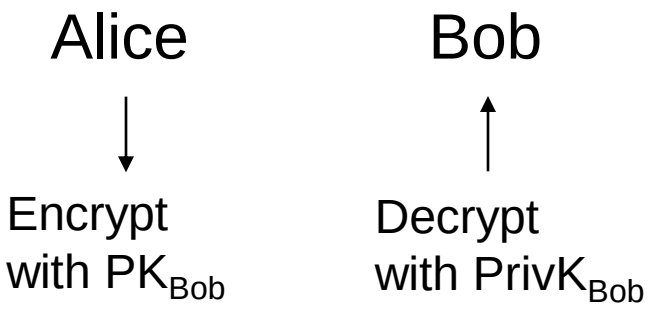


Security Basics – Public Key Cryptography

Public Key Cryptography

- Public/private key pair (PK,PrivK)
  - Public key PK can be revealed to the world / other entities.
  - Private key PrivK must be kept secret.
  - Private key and public key are inverse to each other.
- Operation
  - Anyone can encrypt with the public key – only the owner of private key can decrypt.
  - Owner can encrypt with private key – others can decrypt with public key, e.g. to sign a message.
- Asymmetric ciphers are usually based on concept of one-way functions.
  - Easy to solve in one direction, but hard to reverse.
- Well-known ciphers are, e.g.:
  - RSA (based on factoring mod n)
  - ElGamal (based on log(x) mod n)
  - Elliptic Curve Cryptogaphy (ECC)
- Note: it is unknown whether one-way functions exist. We only know some candidates, but no proof.

Safeguard private key!



Security Basics – Forward Secrecy / Diffie-Hellman

Lifetime of secrets and keys

- Usually,
  - Longterm keys for authentication
  - Session keys for data
- Problem
  - If longterm key is broken, session keys and data may get disclosed.

(Perfect) Forward Secrecy

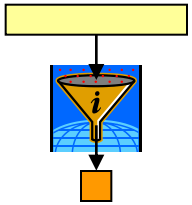
- Forward Secrecy is the property of a key establishment protocol that even if a longterm key is compromised in the future, the derived session keys will not be compromised.

Diffie-Hellman Key Exchange

- Given an authenticated channel, Diffie-Hellman is a protocol to derive session keys so that forward secrecy can be achieved.
- Alice A and Bob B exchange numbers g^a and g^b and create shared secret g^{ab} that cannot be guessed by an observer. The calculations are done in group p and therefore all values are modulo p.

Cryptographic Hash functions

- Hash function:
  - Map arbitrary (often large) value to small one (e.g., 160bit long)
  - Implies collisions: two values can map to the same hash value.
  - A good hash function makes collisions as unlikely as possible.
    = not significantly more likely than 1 / 2^bitlength



- Cryptographic hash functions satisfy additional requirements:

  a) Pre-image resistance

  For given y, hard to compute x with h(x)=y

  b) 2nd pre-image resistance (also: weak collision resistance)

  For given x‘, hard to compute x with h(x) = h(x‘) and x<>x‘

  c) Collision resistance (also: strong collision resistance)

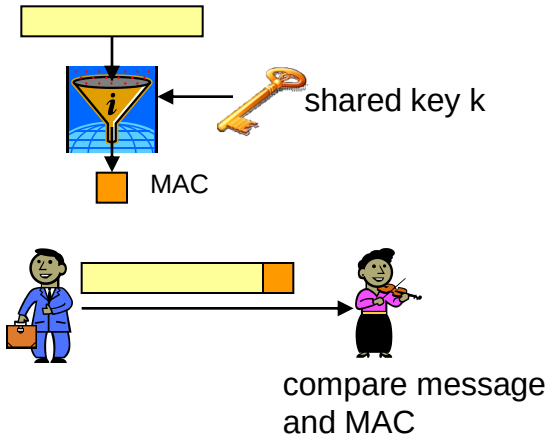
  Hard to find a pair x,x‘ with h(x)=h(x‘) and x<>x‘

  - We know candidates, but again have no proof that a-c) hold for them

Security Basics – Cryptographic Hash Functions

Cryptographic Hash functions

- Can be used to fingerprint data to provide data integrity
  - Message Authentication Code (MAC)
    - Use key to protect hash value
    - e.g. HMAC
      `hash(padding_1,K,hash(padding_2,K,m))`
  - Integrity with Message Digest Code (MDC)
    - Encrypt MDC (hash of data) with shared key or public key cryptography to protect it.
- Well-known functions used as cryptographic hash functions:
  - SHA-1 (collision resistance now doubtful, to be replaced)
  - MD5 (shown to be flawed)



Security Basics – Strategies

Strategies for communication systems can be

- Closed
  - Do authentication and access control.
  - Only allow a group of legitimate users.
  - Practically all security goals can be achieved in some way or other.
- Open
  - Anyone may participate.
  - React to misbehaviour.
    - Reputation systems
    - Incentives for good behaviour
    - Attack / Intrusion detection
  - Or be robust against misbehaviour.
    - Misbehaviour should not have devastating effect.

Authentication

Authentication

- The first step in achieving the standard security goals is usually **authentication**.
  - Many other security goals pointless without it – e.g., no confidentiality if we accidentally give the secret key to the wrong receiver.
- Many definitions of authentication
  - **Entity authentication:** “Entity authentication is the process whereby one party is assured of the identity of a second party involved in a protocol, and that the second has actually participated”
    [Menezes et. al]
  - There are better and stronger definitions, but this will suffice in this context.

Authentication – Goals

Aspects of Authentication

- Authentication Decision
  - We need to obtain information from the other entity so that we believe that it is the entity with a certain ID.
    - E.g. some knowledge that only the other entity can have.
- Freshness
  - Did the other entity participate in the protocol?
    - Replay attacks: an attacker reuses messages from previous correct protocol runs.
  - We need to ensure that the other entity is actively participating in the current run of the protocol.
- Key Establishment
  - In most cases, the result of an authentication protocol is the establishment of a shared secret (key).
    → Authentication and Key Establishment protocols
  - A „good key“ should be *fresh* and *authenticated*.

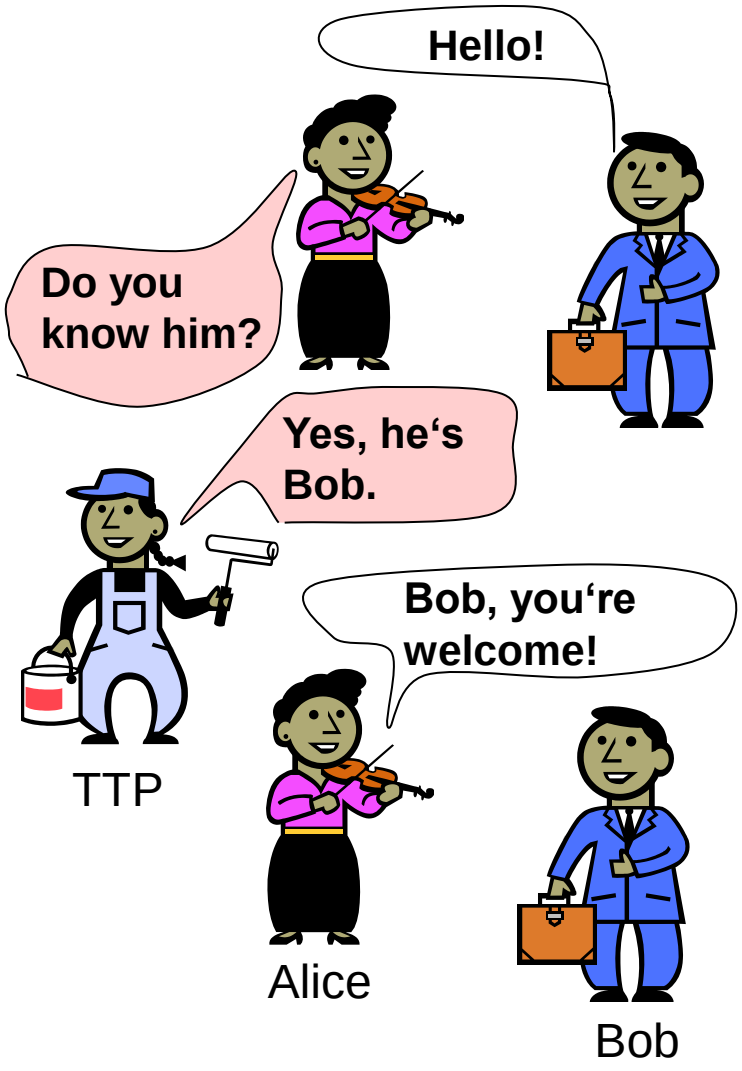
Authentication – Trusted Third Parties

Trusted Third Party (TTP)

- A TTP is an authority that all entities of a protocol trust. The TTP is expected to know the identities.
- In case of symmetric cryptography, the TTP usually has a shared secret with each legitimate entity.
- In case of asymmetric cryptography, the TTP knows the public keys of all legitimate entities.

Certification Authority (CA)

- Entities of an authentication protocol obtain certificates from an authority prior to the protocol run.
- In most cases, the CA will not participate actively in the authentication protocols.



Important: All entities need to trust the TTP on a human or legal level for use within the technical system.
Otherwise, the TTP is of no use.

Certificate

- Generated by Certificate Authority (CA) for an entity
- Purpose
  - The CA states that an entity and a public key correspond.
- A certificate contains
  - Cleartext
    - **Name of the entity (e.g. Bob)**
    - **Public Key of entity**
    - Name of the CA
    - further data about the entity
    - (optionally) more data about CA (like Public Key)
    - for all the cryptographic operations the algorithms that are used
  - **Signature by the CA**
    - Hash value of cleartext signed with private key of CA

Certificate
--- for ----
Name: Bob
Public Key:
RSA 47399844398
….
--- by ---
CA: GlobalCA Inc.
Public Key:
RSA 10499339940
--- Signature ---
10493850405

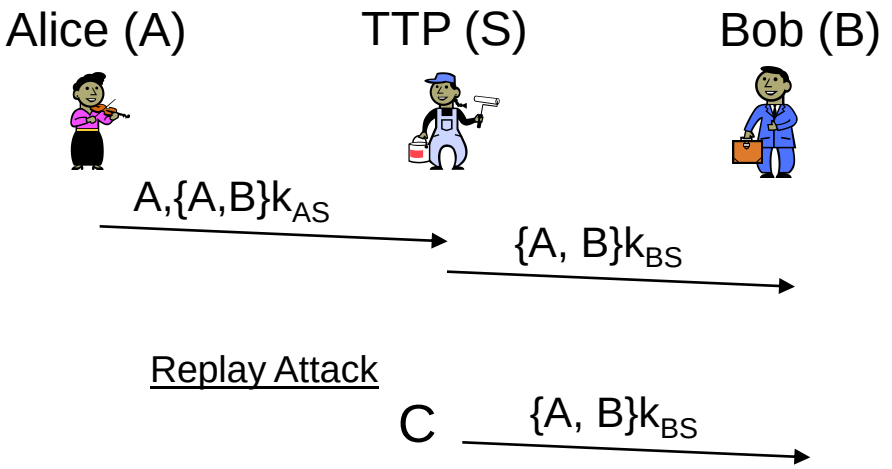
Terms and Notation

- N_X
  - Nonce = fresh random number chosen by X („number used once“).
  - Usually presented by other party to X in later protocol steps, to show it was actively participating and knows the correct keys.
- {M}k
  - M is encrypted and integrity protected with symmetric key K.
- [M]k
  - Keyed-hash of M with key K.
- $E_X(M)$
  - M is encrypted with public key of X.
- $Sig_X(M)$
  - M is signed with private key of X.

Authentication using TTP and attacks

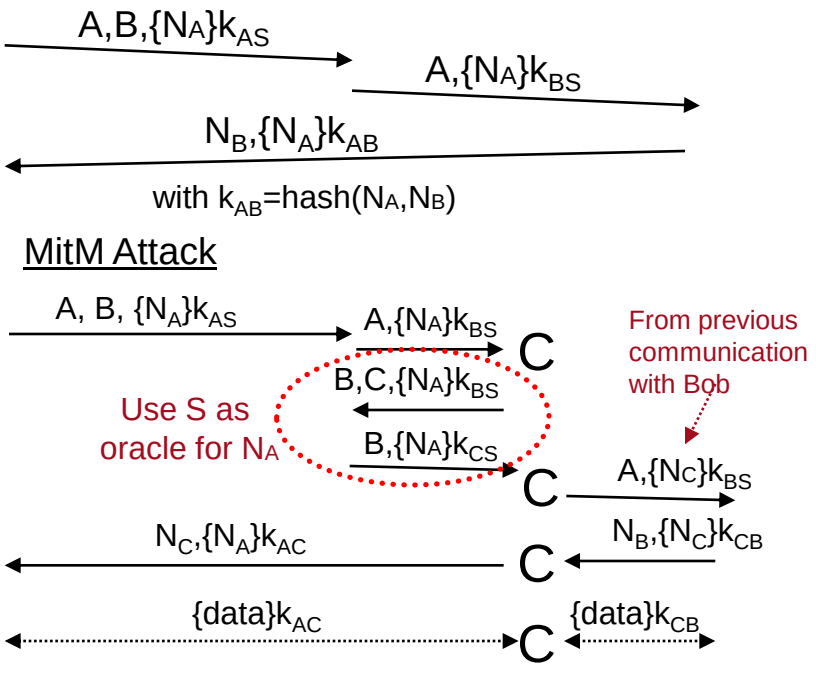
Replay Attack

- An attacker C can resend the second message.
- Bob cannot decide whether the message is fresh or not.
- Reacting to an old message can result in security compromise!



Man-in-the-Middle attack

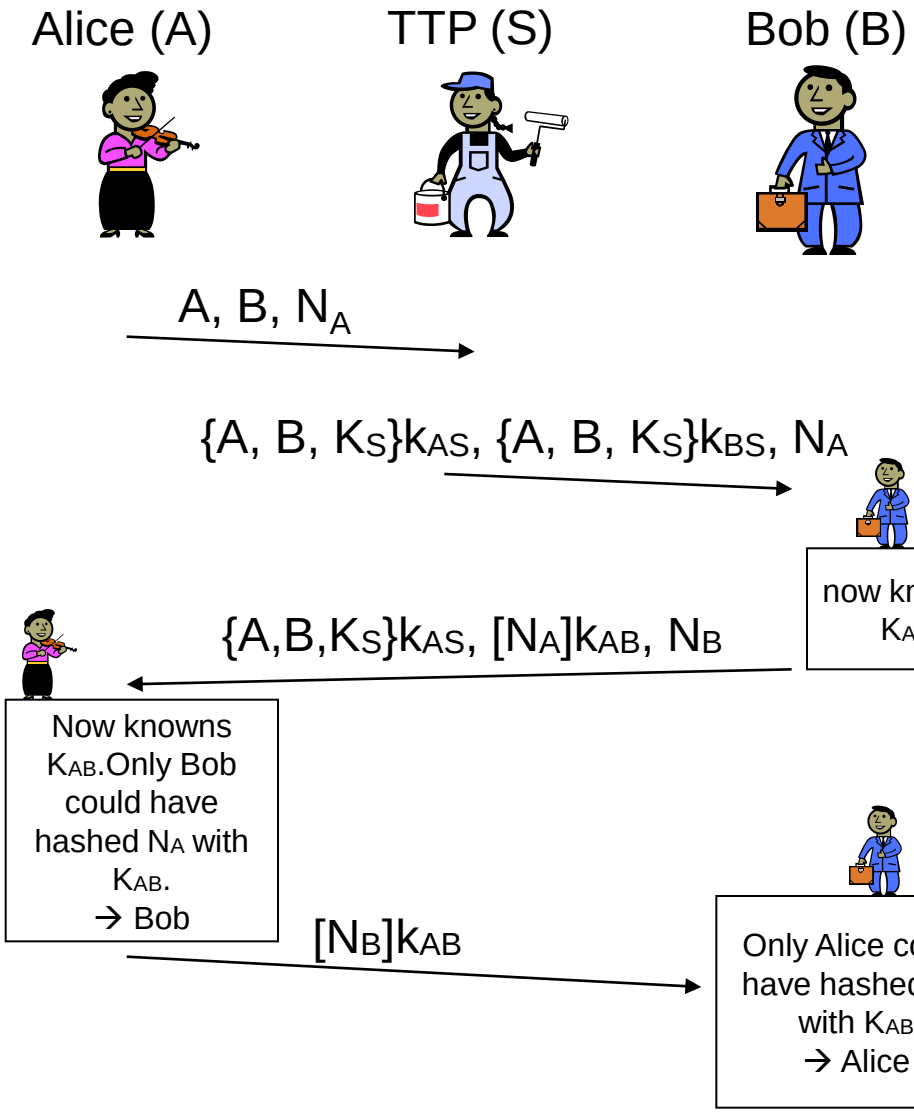
- C positions itself between Bob and Alice, and between Bob and the TTP.
- In this example, we assume that C has once talked to Bob and seen the second message containing $\{N_C\}k_{BS}$.



Authentication using TTP and Symmetric Keys

Example:
Boyd Key Agreement Protocol

- Provides
  - Mutual authentication
  - Key $K_{AB} = MAC_{K_S}(N_A, N_B)$
  - Key is authenticated, fresh, and confirmed.
  - All 3 entities contribute to key.
  - TTP does not know K_{AB}.
- Assumptions
  - A and B each share a secret key with TTP (K_{AS}, K_{BS}).
- No known attack.



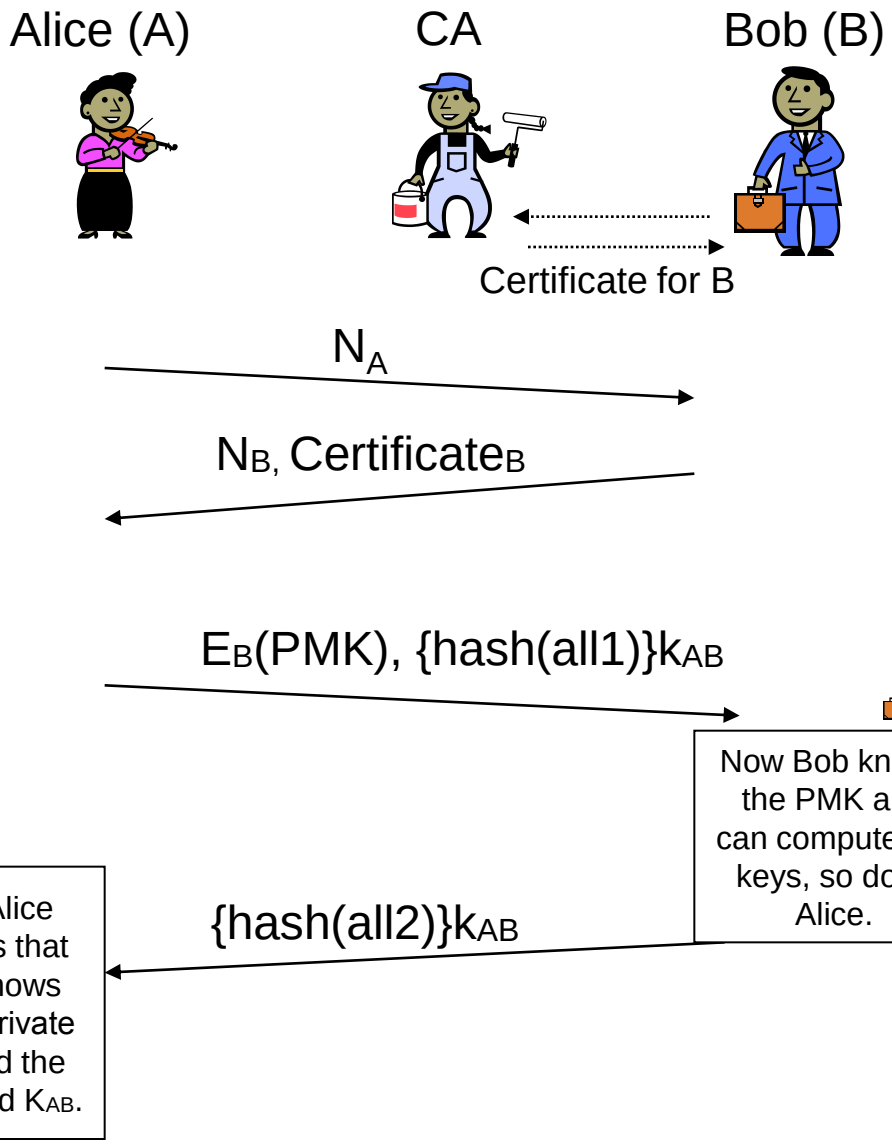
Authentication using CA and Public Keys

Example: TLS / SSL (simplified)

- Alice = Client & Bob = Server
- This is a simplified version of the key transport or key exchange protocol in TLS.
- Per default, only the server (Bob) is authenticated.
- PMK is a random secret created by the client. The keys for the further communication are derived from the PMK, e.g.

$$K_{AB} = Hash_{PMK}(N_A, N_B)$$

- In case of key exchange, messages 2 and 3 would contain the Diffie-Hellman numbers g^a, g^b and PMK = $g^{(ab)}$
- all1 and all2 = all messages till moment of use

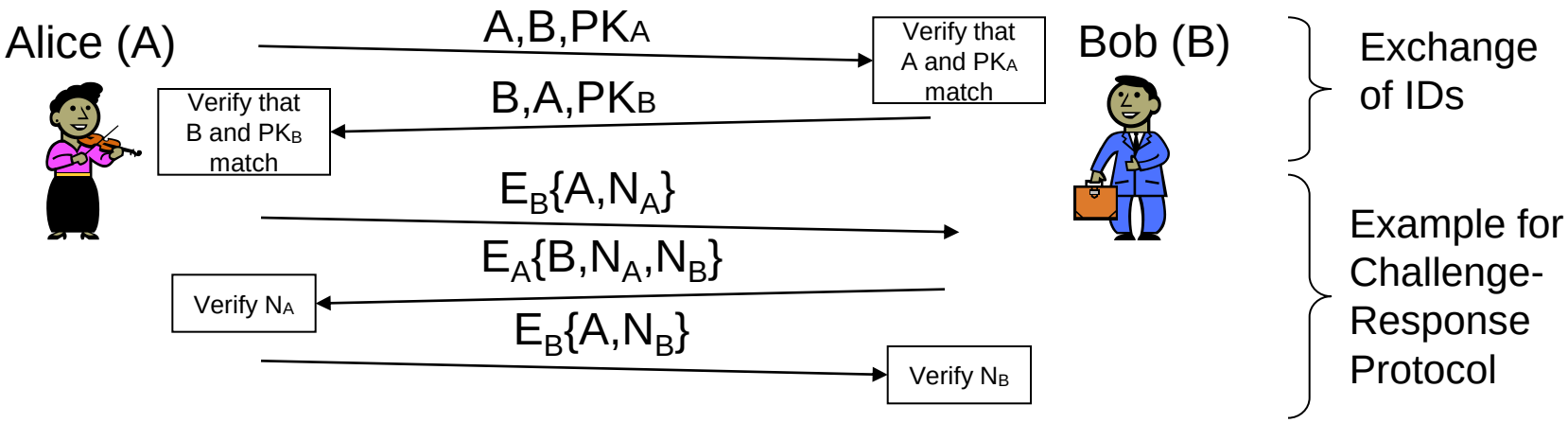


Authentication without global CA?

Cryptographic Identifiers

Cryptographic Identifiers (also called Self-Certifying IDs)

- Idea: Use a public key as identity (usually a hash of a public key)
  - $ID_X = hash(PK_X)$
- A node can sign its messages with its ID.
  - e.g. A → B: $A,PK_A,Sig_A\{Message\}$
- Nodes can verify IDs of other entities with a challenge-response protocol:



Ascertains that A and B are communicating with the correct ID owner, and not a man-in-the-middle. Nonces N_A and N_B are used as challenge.

- A and B can additionally establish a shared key via Diffie-Hellman protocol etc.
- But does not solve the problem: "Who ***is*** the real entity (person?) behind the ID"?

Can we avoid CAs/TTPs? – Boyd‘s Theorem

Theorem 1: „Suppose that a user has either a confidentiality channel to her, or an authentication channel from her, at some state of the system. Then in the previous state of the system such a channel must also exist. By an inductive argument, such a channel exists at all previous states.“

„Another way to interpret the theorem is that no secure channels may be formed between any users who do not already possess secret or shared keys. The result seems quite natural – it is not expected to get something from nothing.“

Discussion of Boyd's Theorems

Theorem 2: „Secure communication between any two users may be established by a sequence of secure key transfers if there is a trusted chain from each one to the other."

[Colin Boyd, „Security Architectures using Formal Methods", IEEE Transactions on Communication, 1993]

→ Authentication cannot be solved within a system alone. It needs an out-of-band mechanism (e.g. personal contact), beyond the scope of the technical system.

Can we achieve secure in-system authentication without CAs or TTPs and no prior contacts?

→ No.

→ The only way around CAs or TTPs is out-of-band communication.

Discussion of Boyd's Theorems

CAs/TTPs are central components, slightly contradictory to the P2P principle. Sometimes, we would like to avoid them.

In such a case, there are some practical ways to improve the situation:

- Add out-of-band mechanisms
  - Voice (Zfone), SMP from Off-the-record messaging, …
- Use social properties
  - Combine security graph with social network graph

  → Try to detect attackers and ignore them
- Use network properties
  - Robust routing or limitations of IP adresses (restrict to subnet etc.)

  → Try to avoid attackers
- Raise the costs for an attacker
  - Make it expensive to join or to get an identity
  - E.g. by Group Decisions
    - Multiparty Computation (a group of peers needs to cooperate to allow a node to join)
- Key Continuity
  - Assume first contact to be secure, and remember key
  - Similar: Duckling Security
- …

Duckling Authentication / Authentication of known entities

Observation: *The main problem for authentication is the first contact when no previous context exists yet. If there is a context, say the shared key of the last session, this can be used for authentication without TTP.*

Baby Duck / Duckling Security Model

- E.g. SSH establishes a relationship by exchanging public keys in the initial session → "host keys"
  - Assumption: no attacker is present.
- Initial contact problem solved for subsequent sessions.
- A successful man-in-the-middle attacker has to be present in the first session and every other session till now to compromise the current session.

→ Once we know an entity and share a secret, we can authenticate. However, we do not want to share a secret with all people in the world or even a small fraction of it. → not a universal solution

Zfone Authentication

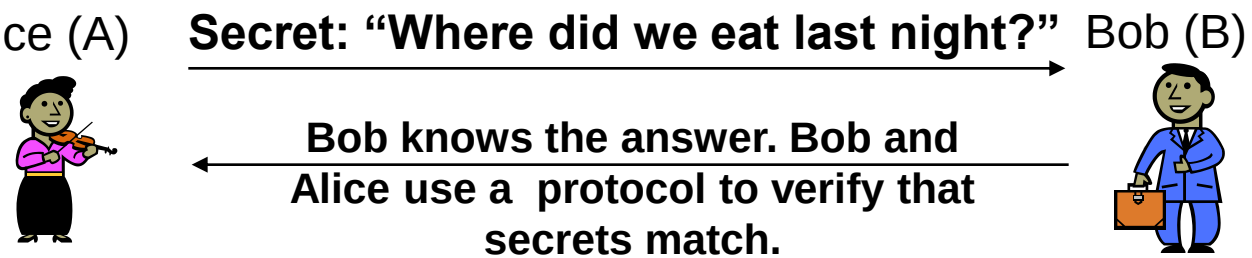
Zfone

- VoIP software by Phil Zimmerman (→ PGP)
- No Public Key Infrastructure (→ difficult to manage securely on a large scale, due to social attacks).

Zfone Authentication (ZRTP)

- Idea: combine human interaction proof and baby duck approach.
- How it works
  - A and B perform Diffie-Hellman exchange (= exchange numbers g^a and g^b and create shared secret g^{ab} that cannot be guessed by an observer).
  - Keying material from previous sessions is used according to duckling idea.
  - A *Short Authentication String* (SAS) is generated as a cryptographic hash of both Diffie-Hellman numbers.
  - For authentication, both users read the SAS and the voice is transmitted to the other user. If the spoken number is correct, the users can confirm the authentication.
    A man-in-the-middle attacker usually needs to intercept and change the Diffie-Hellman numbers to perform the attack on the initial exchange. Thus, he cannot perform a standard man-in-the-middle attack.

Exploiting personal / human secrets for authentication



- If Alice and Bob know each other, they might use their personal knowledge about the other and their meetings to authenticate.
  - Problem: weak secrets (like well-known birthday)
  - Requirements
    - Protocol should resist Man-in-the-Middle attackers.
    - Protocol should not disclose information to potential attackers.
- Example: Socialist Millionaires Protocol (SMP) in Off-the-Record-Messaging (OTR)
  - Off-the-record: term from journalism = “namentlich nicht genannte Quelle”
    - Confidentiality with Authentication, Repudiation (or Deniability) and Forward Secrecy.
  - Socialist Millionaires Protocol (SMP):
    - Two parties, each having a datum x or y, wish to compare whether x and y are equal, without disclosing them to each other.
    - Can use SMP in OTR to allow verification of secret between two parties
  - SMP and OTR use Diffie-Hellman-like exponentiations

Key Distribution

Key Distribution

Authentication Protocols

- Entities prove their identity on the basis of keys and/or certificates.
- How do they „know“?
  → Authentication is intrinsically linked to ***Key Distribution***.

Key Distribution

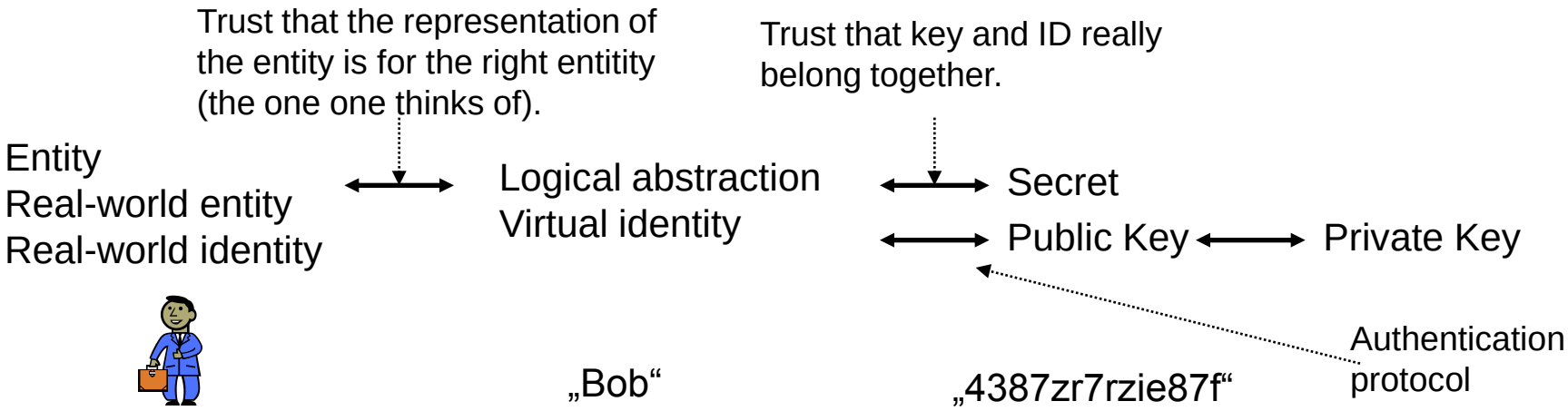
- Fundamental problem for network security.
  - All entities in a system need to know the right keys and need to be able to understand the authentication proofs of other entities on the basis of their keys.
- Linkage of ID and key
  - ID and key are usually not related. Key Distribution also needs to distribute the knowledge about this relation.
- A-priori knowledge
  - There exists an initial key distribution that was established out-of-band (e.g. configured by administrators).
- Direct exchange
  - Entities directly exchange their ID and key and subsequently know the relation.
- Via Trusted Party
  - A trusted party that both entities know introduces the entities to each other.
- Distribute new keys on the basis of existing key distribution.

Key Distribution

Key Distribution and Cryptography

- Symmetric Cryptography
  - Two parties share a key to identify each other.
    - $O(n^2)$ keys if no TTP is used.
  - Groups share a key to identify group.
- Public Key Cryptography
  - One public/private key pair per entity.
    - This corresponds to n keys.
  - Public keys are usually not the IDs. Thus, the relation ID and public key has to be distributed.
  - If a trusted Certificate Authority exists, key distribution is simplified.
    - All entities need to establish a context (keys, IDs) with the CA.
    - Two entities only need to exchange their public keys and certificates when they establish a session.

Key Distribution and Trust

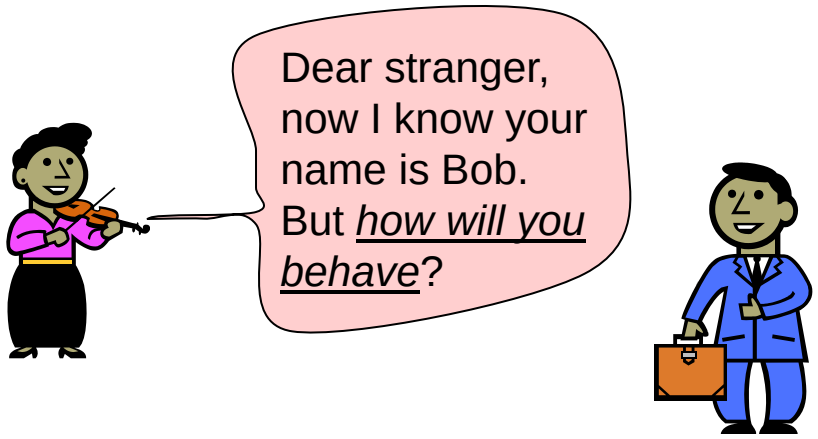


Key Distribution and Trust

- Key Distribution and Authentication implicitely include at least two relations.
  - Virtual ID to Secret/Public Key
  - Entity to Virtual ID
  - Both relations are arbitrary and cannot be established by purely technical means.
  - CAs, TTPs or other peers create them by checking passports or mail addresses, by experience, by knowledge, etc.
- So, the belief that K_B is the correct key for party B is a form of **trust**.
  - As not all entities who sign and distribute such relations are equally trustworthy, entities may have different levels of trust into keys for other entities.

Trust

Trust in general



Peers are now in the network

- Will they behave?
- Will they share?
- Will they participate actively?
- What can I safely tell a peer?
- Will the peer sue me?
- Is the other peer my friend?
- Can I be sure?
- …

„Ich will nicht wissen, wer Sie sind, sondern wie Sie sind!“

Trust

Trust

- The term „trust“ has various slightly different meanings.
  - A trusted party is a party that we trust completely for making decisions (within the technical system).
  - It may define the trust we have on a human or organizational level, say for important or private information.
  - Trust within the technical system can be trust in the correct behaviour with respect to protocol and data usage.
  - Trust within the technical system can also be limited to the expected cooperation for providing a desired service.
    „The entity may be bad, but it will give us what we want.“
    → see also *reputation* and *incentives*

Trust Mechanism

- Compute a trust value on the basis of experience, acquired knowledge and a-priori knowledge.

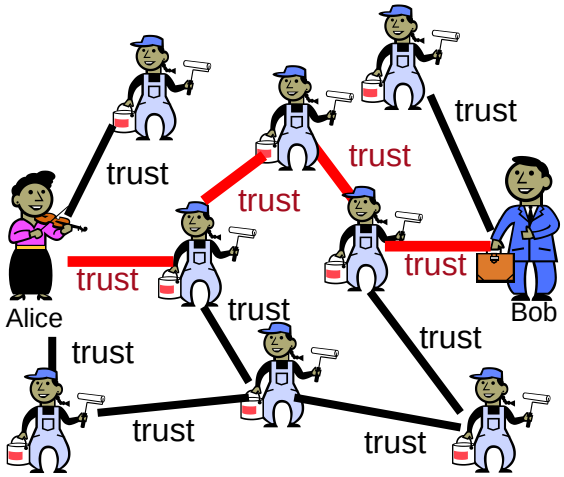
Trust into a key

Trust into a key

- A entity is sure that the key belongs to and is known by the right person.
- How to achieve this?
  - Personal contact (Alice gave Bob her key directly in a personal meeting)
  - An intermediate entity assures that the key belong to a certain person.
    - Certificate Authority
    - Web-of-Trust

Web of Trust

- Instead of a single TTP, the entities in a protocol trust in a chain of trusted peers to establish a trust context between them.
  - E.g., GPG: Public key ←→ ID verification
  - Not necessarily bidirectional.
  - Size of chain may be limited or trust may be rated lower for longer chains, etc.
  - Problem: Trust is often not transitive.



Trust into an entity / Reputation / Incentives

Trust into an entity

- Problem: Will the other entity cooperate?

Option: Incentives

- Incentives are mechanisms to make a peer cooperate by giving it benefit from cooperation
- Utilized by systems like BitTorrent (Tit-for-Tat).
  - Peers upload (= cooperate) in order to increase their own download rate.

Option: Reputation System

- Trust into a service or peer based on experience or a-priori knowledge
  - Often expressed by a trust rating.
- Global vs Local
  - Global reputation: Reputation is determined system-wide. Each peer has an identical rating of, say, Alice.
  - Local reputation: Each node locally computes a reputation value for Alice, based on its local knowledge. The rating for Alice may differ among the peers.

Requirements for Reputation Systems

To determine trust within a technical system, we need to be able to

- To observe an action of the entity
  - e.g. we successfully downloaded a desired item from Alice
- To evaluate the action of the entity
  - e.g. Alice cooperated
- To store or aggregate evaluation reports
  - e.g. increase the counter for „Alice cooperated“ at reputation server
- General rating / make projections about future behaviour
  - e.g. Alice cooperates in 57 % of the cases.

→ There are attacks or misuses that cannot be observed within the system. In such cases, we cannot use technical reputation schemes.

→ E.g. will the anonymizer perform traffic analysis?

Only a-priori information/trust can be used in such cases.

Reputation Systems – Problems and Attacks

Time-dependancy of Trust

- Trust values that are not based on personal social knowledge, but on historic behaviour in the system, can become invalid when the peer changes its strategy.
- Problem: Unable to tell change in behaviour ***in advance***.
- Attack: Acquire trust by cooperating for cheap tasks („sell lots of CDs“), deny or attack valuable transfers („sell some non-existing cars“).

Whitewashing

- A badly-rated peer may simply leave the system, and return with a new innocent idenity.
- Mitigation
  - Fixed unchangable identities (→ how?)
  - Rate new peers like bad peers. → Barrier for new peers to join and stay („bad experience“), attack may still be profitable, …

Collusion of attackers

- Attackers can cooperate and give peers of other attackers good ratings.

A (centralized) global reputation scheme: Ebay

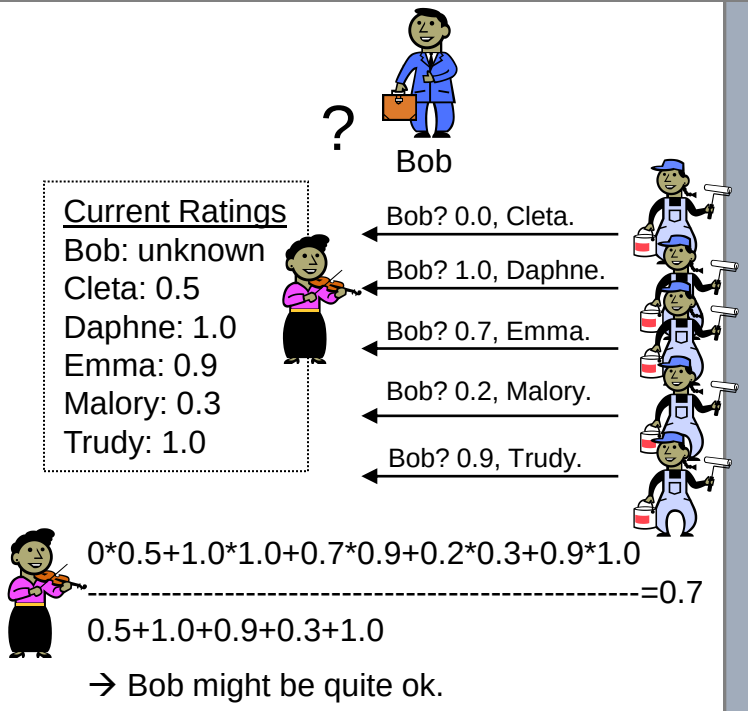
Ebay

- A well-known online platform on client/server basis for auctions with Peer-to-Peer exchange of money and goods.
  - Usually, payment in advance. Will the vendor cooperate?
  - Ensure that the vendor behaves correctly by using a reputation system.
- Repuation Scheme
  1. After an auction, vendor and buyer observe their behaviour.
  2. The buyer sends a rating to the server („+“ positive, „o“ acceptable, „-“ negative). Since 2008 the vendor can only rate positive.
  3. The server stores and aggregates the ratings.
  4. The rating is computed as sum of positive (+1) and negative (-1) ratings. Additionally, a percentage of positive ratings is computed.
     → „394 points and 96 % positive“
- Thus, the rating combines the significance (number of samples) and the quality of the behaviour (sum over „+“ and „-“, and percentage of „+“s)

Decentralized Trust (P2P)

Decentralized Reputation Mechanisms

- Basic idea
  - Use your own experience
  - Use ratings of other peers and combine their rating with your rating for them.
  - Combine knowledge for a new rating.
- Example
  - Use the weighted average on the trust reports with trust as weight.



$$\frac{0*0.5+1.0*1.0+0.7*0.9+0.2*0.3+0.9*1.0}{0.5+1.0+0.9+0.3+1.0}=0.7$$

→ Bob might be quite ok.