Table of contents

Trusted Third Parties (TTP) and Certification **1**
TTP 1
TTP Operating Modes 1
Key Distribution Centers (KDCs) 3
Certification Authorities (CAs) 4
Functional Entities Related to Certification 8
Other TTPs 10
Public Key Authentication 11
Certificates 11
Certificate Revocation Lists (CRLs) 12
Authentication Trees 14
Certification Topologies 17
Public Key Infrastructure (PKI) 19
Main Functional Entities 19
Advantages and Disadvantages 21

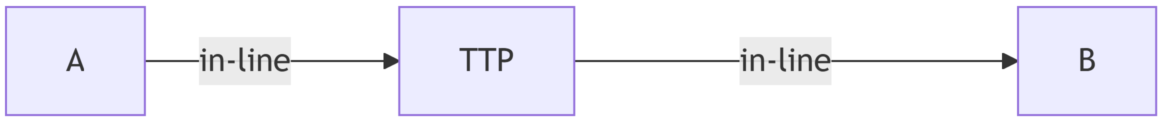
Trusted Third Parties (TTP) and Certification

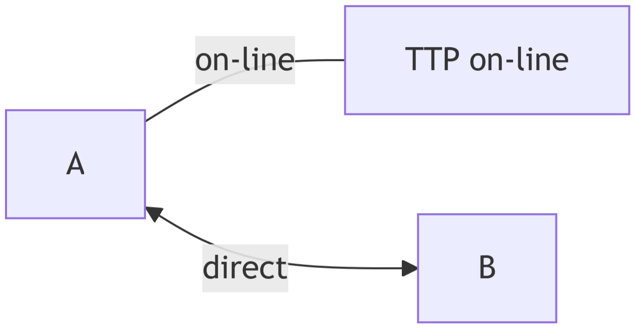
TTP

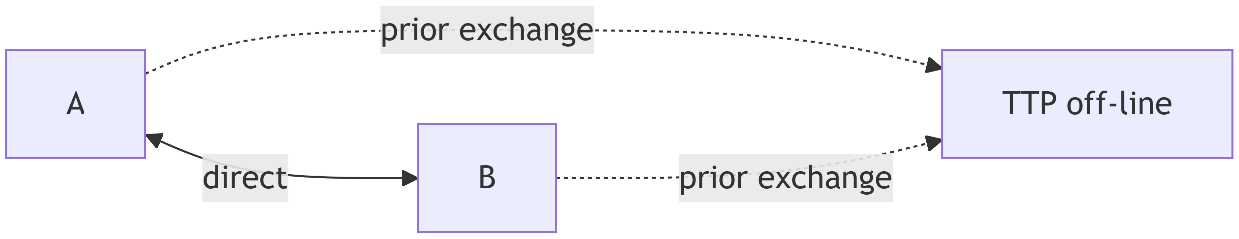
TTP Operating Modes

Three Operational Modes:

- **In-line:** TTP acts as an intermediary, relaying all exchanges in real time (e.g., Proxies, Secure Gateways).
- **On-line:** TTP participates in real time, but A and B communicate directly (e.g., KDC).
- **Off-line:** TTP does not participate in real-time exchanges but makes information available *a priori* (e.g., CA).







Comparison:

- **Off-line:** Facilitates exchanges, no need for permanent availability.
- **In-line/On-line:** Requires permanent availability.
- **Off-line:** Revocation of privileges is more complex.

Original Text

TTP: Operating Modes

- **In-line:** The TTP acts as an **intermediary** to relay exchanges between A and B in real time. Examples: Proxies, Secure Gateways.
- **On-line:** The TTP **participates in real time** in exchanges between A and B, but A and B communicate directly (without passing through the TTP). Example: Key Distribution Center.
- **Off-line:** The TTP **does not participate** in real-time exchanges but makes information available **a priori**. Example: Certification Authorities.

Comparison In-line/On-line/Off-line: Exchanges are facilitated, and there is no need for permanent TTP availability in off-line mode (unlike the other two), but **privilege revocation** (e.g., when a secret key is compromised) is more complex.
[Diagrams showing the three modes]

Quick Review

TTP Modes:

- **In-line:** Intermediary relay
- **On-line:** Real-time participation
- **Off-line:** Information available *a priori* (e.g., CA)
- Off-line: No availability required but revocation complex

Key Distribution Centers (KDCs)

Objective: Solve the n^2 key distribution problem.

Principle:

- Without KDC: $\frac{n(n-1)}{2} \approx n^2$ keys for n entities.
- With KDC: Only n keys (each entity shares a key with the KDC).
- Session keys dynamically generated by the KDC.

Advantages:

- Scalability: A new entity = only one new key.
- Secure channel establishment via tickets (as in Kerberos).

Disadvantages:

- **Single point of security failure:** KDC compromise $\rightarrow$ entire system vulnerable.
- **Single point of operational failure:** KDC unavailability (DoS) $\rightarrow$ system paralysis.
- **Performance bottleneck:** Costly operations (encryption, random generation).

Solutions: Mirroring, load balancing.

Original Text

TTP: Key Distribution Centers (KDCs)
Goal: Solve the n^2 **key distribution problem**:

- In a symmetric environment with n entities without an intermediary: $n(n-1)/2 \sim n^2$ different keys are needed for all pairs of entities to share a different key.
- Moreover, such a system is not **scalable** because adding an entity results in the generation of n new keys.

If each entity shares a key with a KDC, only n **keys** are needed for the system to function, and one key suffices for each new entity. The establishment of secure channels is ensured by the generation of session keys and the presence of tickets as in Kerberos.
Problems:

- **Single point of security failure:** By construction, the KDC can impersonate all nodes in the network. If it is compromised, the entire system becomes vulnerable.
- **Single point of operational failure:** The usual mode of operation of a KDC is on-line (possibly in-line). If it becomes unavailable (e.g., due to a denial-of-service attack), the entire system is paralyzed.
- **Performance bottleneck:** KDC operations are often computationally expensive (encryption/decryption, random generation, etc.). Classic solutions (e.g., mirroring) must be considered to distribute the KDC load.

Quick Review

KDC:

- Solves n^2 problem $\rightarrow n$ keys
- Scalable: +1 entity = +1 key
- Risks: Single point of failure (security + operational), bottleneck

---

Certification Authorities (CAs)

Role: Authenticate the association between an **entity** and its **public key**.

Operation:

1. CA verifies identity (passport, etc.).
2. CA creates and signs a **certificate** containing this association.
3. Certificates are accessible to entities (may be cached).

Verification: Requires an authentic copy of the CA's public key.

Advantages:

- **Off-line mode:** Short unavailability is acceptable.
- Simpler then safer: No complex protocols needed.

Disadvantages:

- **Asynchronous revocation:** Certificate may become invalid (private key theft).
- **Solution:** Signed Certificate Revocation Lists (CRLs).

CA Compromise:

Serious consequences if the private signing key is compromised.

Original Text

TTP: Certification Authorities (CAs)
The primary role of a **Certification Authority (CA)** is to **authenticate the association between an entity and its public key** (think of Man-in-the-Middle attacks!).
The CA will **create and sign certificates** containing this association (using proof of identity such as a passport) and make them accessible to the relevant entities.
Once signed, copies of the certificates (**cached certificates**) can be stored in unprotected locations (e.g., on the user's disk space). However, to verify the signature of the certificates, the relevant entities require an **authentic copy of the CA's public key**.
Simpler then safer: There is no need to implement complex protocols in a CA.
The usual mode of operation of a CA is **off-line**, which reduces the impact of short periods of unavailability.
Problem associated with off-line mode: The validity of cached certificates may be questioned **asynchronously** due to a private key theft.
Remedy: CAs also publish signed lists of invalid certificates (**Certificate Revocation Lists** or **CRLs**).
Compromising a CA has less obvious but almost as harmful consequences as compromising a KDC, especially if the private key used to sign certificates is also compromised.

Quick Review

CA:

- Authenticates entity   public key association
- Signs certificates (off-line mode)
- CRLs for revocations
- Compromise = serious consequences

Proof of Possession (PoP)

Problem: Identity verification is not sufficient; possession of the private key must also be verified.

Attack without PoP:

1. A signs a document and sends it to B (notary): $S_{priv_A}(\text{Invention}), S_{priv_{CA_A}}(Cert_A)$.
2. C intercepts and requests from CA_C (without PoP) a certificate associating C with pub_A.
3. C sends to B: $S_{priv_A}(\text{Invention}), S_{priv_{CA_C}}(Cert_C)$.
4. C becomes the inventor!

Simple PoP Protocol:

1. $CA \rightarrow A : A, r$ (random number + identity).
2. $CA \leftarrow A : S_{priv_A}(A, r)$.
3. CA verifies the signature with pub_A.

Consequences:

- Introduces **trust levels** for CAs.
- Criteria: PoP, CRL updates, signing key security.
- Problem exacerbated by uncontrolled CA proliferation.

Original Text

CA: Proof of Possession (PoP)
Verifying the identity of A to create (or modify) a certificate associating A with its public key is not a sufficient criterion. It is also necessary to **verify that A truly possesses the corresponding private key**.
Let A and its CA be CA_A. Let's see what an active attacker C can do in "collaboration" with a CA_C that does not verify PoP:
A signs a document containing the description of a revolutionary invention and sends it to B (the notary) with its certificate signed by CA_A:
$A \rightarrow B : S_{priv_A}(\text{Invention}), S_{priv_{CA_A}}(Cert_A)$
C intercepts this packet, approaches CA_C, and asks it to create a certificate associating its identity C with A's public key, and sends to B:
$C \rightarrow B : S_{priv_A}(\text{Invention}), S_{priv_{CA_C}}(Cert_C)$
C thus becomes the revolutionary inventor...
Simple PoP Verification Protocol:
$CA \rightarrow A : A, r$; r: random number, A to protect A from chosen message attacks.

$CA \leftarrow A : S_{priv_A}(A, r)$; CA only needs to verify the signature with pub_A.
This criterion and other behavioral criteria, such as CRL updates or the security of the signing key, introduce **trust levels** for CAs and for the certificates they sign.
This phenomenon is exacerbated by the **uncontrolled proliferation of CAs**!

Quick Review

PoP (Proof of Possession):

- Verify possession of private key (not just identity)
- Without PoP: Attacker can impersonate identity
- Protocol: CA verifies signature on challenge
- Introduces trust levels for CAs

Separation of Certification and Revocation

Problem: If the same key signs certificates and CRLs, an adversary possessing the CA's private key can:

1. Publish a CRL revoking victim A's certificate.
2. Create a fake certificate associating A with a key controlled by the adversary.
3. Decrypt confidential transactions for A.
4. Impersonate A (signatures, authentication).

Solution: Separation of Duties

Clear separation of tasks:

- **Certificates** and **CRLs** signed with **different keys**.
- By **different functional entities**:
  - **Certification Authority (CA)**
  - **Revocation Authority (RA)**
- Residing on **different machines**.
- Subject to **independent security policies**.

Original Text

CA: Certification and Revocation
Problem: If the same key is used to sign certificates and CRLs, an adversary possessing the private signing key of a CA can attack a "victim" A under the authority of this CA

as follows:

- Publish a CRL containing A's revoked certificate.
- Create a certificate associating A with a public key for which it controls the private key to then:
  - act as a Man-in-the-Middle to decrypt confidential transactions for A;
  - impersonate A for authenticated transactions or signed documents.

Solution: Separation of duties: Certification and revocation become clearly differentiated tasks:

- Certificates and CRLs are signed with **different keys**,
- by **different functional entities** (Certification Authority and Revocation Authority);
- if possible, residing on **different machines** subject to **independent security criteria (security policies)**.

Quick Review

Separation of Duties:

- Certificates   CRLs (different keys)
- CA   Revocation Authority
- Separate machines and policies
- Prevents post-compromise attacks

Functional Entities Related to Certification

Name Server:

- Manages a unique and consistent namespace.
- Combined with certification if authentication is required.
- Example: **DNSSec** (authenticated DNS for the Internet).

Registration Authority (RA):

- Tasks requiring direct contact with entities.
- Identity verification, PoP, certificate requests/modifications.
- Detached from CA for geographical reasons.

Key Generator:

- Generates public/private key pairs.
- **Advantages:** User simplicity, enhanced key security.
- **Disadvantage:** Private key known to another entity → loss of non-repudiation.

Certificate Directory:

- Read-only access directory for certificates.

Original Text

Functional Entities Related to Certification

- **Name Server:** Responsible for managing a **unique and consistent namespace**. When authentication is required, name management must be complemented by the certification of public keys associated with these names.

  Example of a pilot solution combining both concepts: **DNSsec**: authenticated name management environment for the Internet.

- **Registration Authority:** Entity responsible for performing tasks related to certificate management requiring **direct contact** with the relevant entities. These tasks include verifying the parameters necessary for the initial request or modification of certificates (identity verification, PoP, etc.). Detaching this functionality from the CA is typically due to geographical considerations.

- **Key Generator:** Allows delegation of the public/private key pair creation process to a dedicated entity:
  - **Advantages:** Simplicity for users; possibility to enhance the security of chosen pairs.
  - **Disadvantage:** Private key known to another entity! Loss of non-repudiation.

- **Certificate Directory:** The directory allowing users to access (read-only) the certificates of correspondents.

Quick Review

Certification Entities:

- **Name Server:** Names + DNSSec
- **RA:** Direct contact, verifications
- **Key Generator:** Key generation (loses non-repudiation)
- **Certificate Directory:** Read-only certificate access

Other TTPs

Timestamp Agent (TA):

- Certifies the existence of a document/transaction at a specific time.
- Methods:
  - Associate a timestamp with the document (or $h(\text{doc})$) + sign.
  - Use an authentication tree.

Notary Agent:

- Like TA but also: validity, origin, ownership.
- Legal support for non-repudiation.

Key Escrow Agent (KEA):

- Access to session secret keys under conditions (judicial warrant).
- Requires a dedicated encryption system.

Example: Clipper/Capstone

- **Clipper chip** (1993): Symmetric encryption with KEA access.
- Controversial, flaws discovered.
- **Capstone chip:** Successor (Fortezza PCMCIA card).
- Military-level security.

Original Text

Other TTPs

- **Timestamp agent (TA):** Certifies the existence of a document or the occurrence of a transaction at a well-specified time. To do this, the TA can:
  - associate a timestamp with the document (or with $h(\text{doc})$ where h is a Collision Resistant Hash Function) and sign the whole with its private key, and
  - use an authentication tree (see page 231).
- **Notary agent:** Certifies not only the existence of a document at a given time (like the TA) but also its **validity, origin, or ownership** by a given entity. This service provides (legally necessary?) support for non-repudiation.
- **Key escrow agent (KEA):** Entity authorized to access session secret keys provided certain conditions (e.g., a court order) are met. This requires a dedicated encryption system. Example: the Clipper key escrow system:

- Announced in April 1993 by the US administration, amid great controversy, as the large-scale communication encryption solution.
- The **Clipper chip** is a symmetric encryption/decryption device that provides access to session keys when the secret keys of two KEAs (typically federal agencies) are input.
- The presence of some flaws and the need for asymmetric cryptography led to its successor: the **Capstone chip**, which can be integrated into a PCMCIA card (called **Fortezza** and used for military-level security).

Quick Review

Other TTPs:

- **TA:** Timestamp document existence
- **Notary:** TA + validity/origin (non-repudiation)
- **KEA:** Access to keys under legal conditions
- Example: Clipper/Fortezza (controversial)

Public Key Authentication

Certificates

Definition: Information associating an entity with its public key.

Generic Structure:

- **Serial Number, Version**
- **Issuer:** Identity of the signing CA (global and unique).
- **Signature Algorithm:** Algorithm used to compute the signature (e.g., MD5+ElGamal, SHA+RSA).
- **Subject:** Name of the certified entity (global and unique).
- **Subject Public Key:** Public key.
  - RSA: (n, e)
  - DH: $(p, \alpha, \alpha^x \mod p)$
- **Subject Public Key Algorithm:** RSA, DH, etc.
- **Validity:** Validity period (UTC).
- **Signature:** Covers all previous records, ensuring authenticity.

Original Text

Public Key Authenticity: Certificates
A **certificate** is a piece of information associating an entity with its public key. Generically, it consists of the following elements:

- **Serial Number, Version**.
- **Issuer:** The (global and unique) identity of the signing CA.
- **Signature Algorithm:** The algorithm used to compute the signature on the certificate. E.g.: MD5 + ElGamal or SHA + RSA.
- **Subject:** The (global and unique) name of the entity whose public key is certified.
- **Subject Public Key:** The entity's public key. For example:
  - (n, e): modulus and public exponent for RSA.
  - $(p, \alpha^x \mod p)$: modulus, generator, and public part for Diffie-Hellman.
- **Subject Public Key Algorithm:** The algorithm associated with the public key. E.g.: RSA or Diffie-Hellman.
- **Validity:** The certificate's validity period, typically expressed in UTC.
- **Signature:** Contains the signature computed using the Signature Algorithm and the CA's private key. It covers all previous records and thus guarantees the authenticity of the information they contain.

Quick Review

Certificate:

- Issuer (CA) + Subject (entity)
- Public key + algorithm
- Validity period
- CA signature on all fields

Certificate Revocation Lists (CRLs)

Definition: Lists of certificates that have become invalid.

Reasons for Revocation:

- Private key compromised.
- Algorithm modification.
- Role change (role-based certificate).

- Other factors invalidating certificate information.

CRL Structure:

- **Issuer, Signature Algorithm**
- **Date of Issue, Date of Next Issue**
- For each revoked certificate:
  - **Serial Number**
  - **Revocation Date**
- **Signature:** On the entire list.

Requirements:

- CAs must publish CRLs **frequently**.
- **Wide-audience** distribution channels.
- Minimize fraud risk.

Problem: Revocation is the Achilles' heel of public key systems.

Alternative Solutions:

- Certificates with very short validity (a few minutes).
- Periodic re-confirmation by CAs.
- Return to on-line mode → high availability required.

Original Text

Certificate Revocation Lists (CRLs)
These are lists containing certificates that have become **invalid** due to a compromised private key or any other factor affecting the validity of the information contained in a certificate (algorithm change, role change for a role-based certificate, etc.).
A generic CRL has the following elements:

- **Issuer, Signature Algorithm:** As for certificates.
- **Date of Issue, Date of Next Issue:** Issue date and next issue date.
- For each revoked certificate, the following records:
  - **Serial Number** of the revoked certificate.
  - **Revocation Date**.
- **Signature:** Signature covering the entire list.

A CA must publish CRLs with a **very high frequency** and use **wide-audience** distribution channels to reduce the risk of fraud.
Revocation is the Achilles' heel of any public key system...

A solution: certificates with very short validity periods (a few minutes) requiring periodic re-confirmation by CAs…
…but this brings us back to on-line mode and thus requires high CA availability.

Quick Review

CRLs:

- Lists of invalid certificates (compromise, etc.)
- Structure: issuer, dates, serial numbers, signature
- Frequent publication required
- Achilles' heel of PKI
- Alternative: short-lived certificates ($\rightarrow$ on-line)

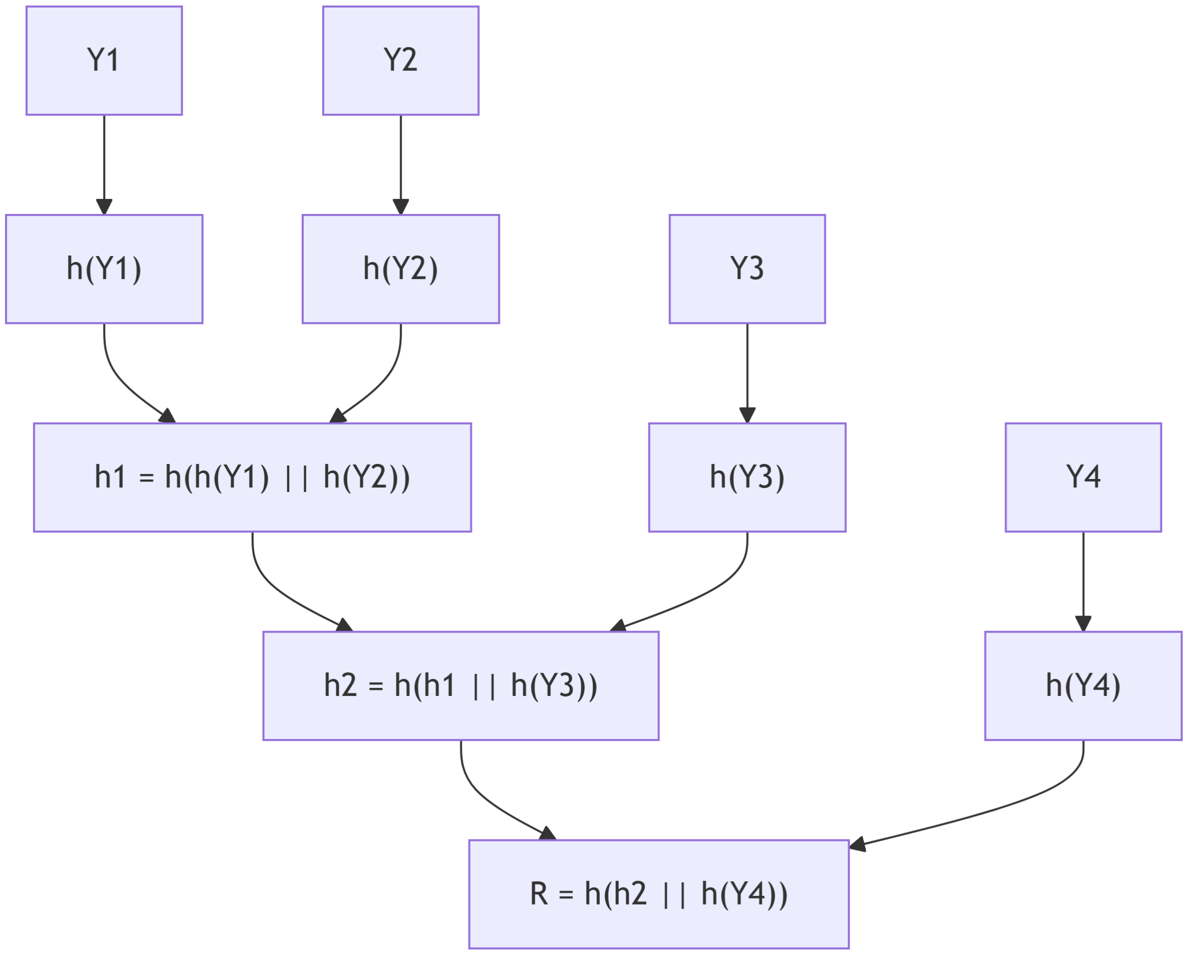
Authentication Trees

Principle: Alternative to certification for authenticating public information.

Construction: Tree (binary) with hash function + root authentication.

For n values $Y_1, Y_2, \ldots, Y_n$:

1. Values placed in leaves.
2. Edges from leaves labeled $h(Y_i)$.
3. Non-terminal nodes labeled $h(h_1||h_2)$ ($||$ = concatenation).
4. Root R authenticated (digital signature).



Verification of Y_1: Provide $h(Y_2), h(Y_3), h(Y_4)$, compute $h(Y_1), h_1, h_2$, and verify $h(h_2||h(Y_4)) = R$.

Advantages:

- Only R requires cryptographic protection.
- Balanced trees: Intermediate data $\sim \log_2 n$.

Disadvantages:

- Modified node $\rightarrow$ recalculate path to root.
- Adding nodes $\rightarrow$ unbalanced trees recommended.

Main Application: Timestamping

TA:

- Builds tree.
- Provides signed timestamp + verification path.
- Publishes R daily (newspaper).

Original Text

Authentication Trees
Authentication trees are an **alternative to certification** for authenticating public information.
They exploit the advantages of a tree structure (typically binary) with the use of hash functions and root authentication.
Given a tree A with n leaves. Let h be a collision-resistant hash function (CRHF). Tree A can be used to authenticate n public values $Y_1, Y_2, ..., Y_n$ by constructing an authentication tree as follows:

1) The values $Y_1, Y_2, ..., Y_n$ are placed in the leaves of the tree.
2) Each edge from a leaf Y_i is labeled $h(Y_i)$ (h being a CRHF).
3) Each non-terminal node with underlying edges labeled h_1 and h_2 is labeled $h(h_1||h_2)$ ($||$ denotes concatenation).

[Tree diagram]
To verify the authenticity of Y_1, it is necessary to provide the values $h(Y_2), h(Y_3), h(Y_4)$. Then, it suffices to compute $h(Y_1), h_1$, and h_2 (according to the figure) and accept the authenticity of Y_1 if $h(h_2||h(Y_4)) = R$. An illicit modification in Y_1 would result (by the characteristics of the CRHF) in a different value for $h(h_2||h(Y_4)) \neq R$.
Note that only the value R needs to be authenticated (e.g., using a digital signature). The other values are protected by the irreversibility of the CRHF.
Advantage: Only R requires cryptographic protection for authentication!
Disadvantages:

- To verify the value Y_1, the values $h(Y_{2,3,4})$ and the value R are needed. To minimize this effect, balanced trees (trees whose paths differ by at most one edge) can be used to reduce the number of intermediate data to $\sim \log_2 n$.
- When a node is modified, the entire path to the root must be recalculated.
- When new nodes are added, it is advisable to build unbalanced trees (like the one in the figure) and add nodes via the root.

Main application: timestamping: The timestamping agent (TA) builds such a tree and provides the requester with the timestamp signed with its private key along with the verification path. The TA publishes R daily in a newspaper, preventing it from cheating!

Quick Review

Authentication Trees:

- Certification alternative via hash + tree
- Only root R signed
- Verification: path $\sim \log_2 n$ values
- Application: timestamping
- TA publishes R daily

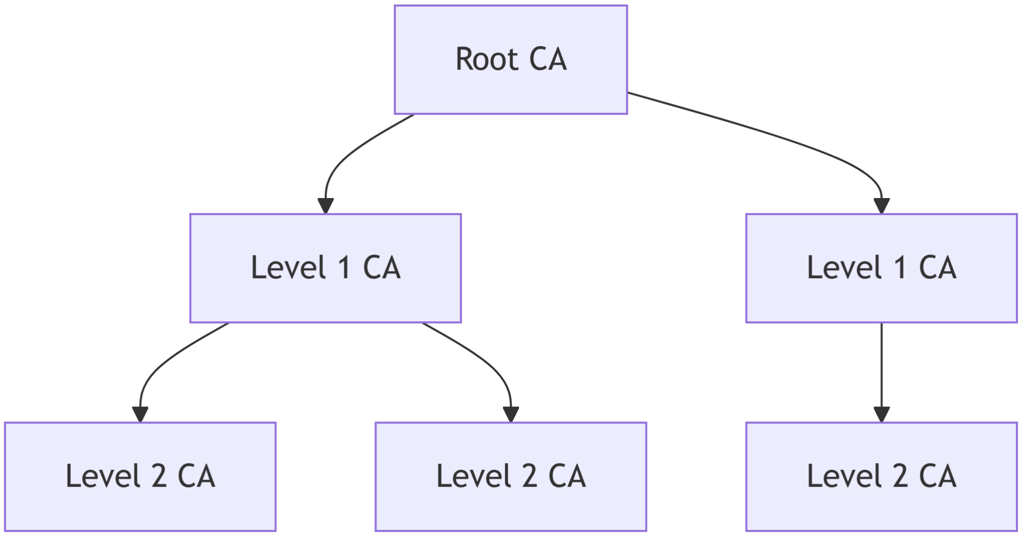
Certification Topologies

Problem: Communication between users of different CAs → trust question.

Cross-Certification:

- CA_A certifies the public key pub_{CA_B} of CA_B.
- Cross-certificate: $CA_A\{CA_B\}$.
- Certification chain: $CA_A\{CA_B\} \rightarrow CA_B\{B\}$.
- A verifies B's key with an authentic copy of pub_{CA_A}.

Strict Hierarchical Model (PEM/X.509):



- Any non-local chain starts at the root node.

- Root public key assumed to be globally known.
- **Problem:** Centralization, single point of failure.

Graph Model (PGP):

- Users act as CAs for correspondents.
- Decentralized graph structure.
- Suitable for closed groups.
- **Limitations:** Unconnected populations.

Hybrid Models:

- Hierarchy + bidirectional cross-certification.

Golden Rule: Short chains (weakest link!).

Original Text

Certification Topologies
When two users belonging to different CAs wish to communicate, a trust problem arises: should one trust a certificate issued by another CA?
The **cross-certification** process allows CA_A to certify the public key pub_{CA_B} of CA_B. The resulting certificate is called a cross-certificate, denoted: $CA_A\{CA_B\}$.
If A wants to verify the authenticity of B's public key and there exists a cross-certificate $CA_A\{CA_B\}$, A will ask B to provide its certificate signed by CA_B, i.e., $CA_B\{B\}$. The resulting **certification chain**: $CA_A\{CA_B\} \rightarrow CA_B\{B\}$ allows A to verify B's public key using an authentic copy of pub_{CA_A}.
The trust relationship necessary for cross-certification is not always easy to establish in competitive environments, which is why **hierarchical models** among CAs have been proposed. Example: the strict hierarchical model of PEM/X.509:
[Hierarchy diagram]
In the PEM environment, any non-local certification chain starts at the **root node**, whose public key is assumed to be known worldwide…
Other models, such as the one proposed by **PGP**, are based on a graph structure where the nodes are users who act as CAs to certify the public keys of correspondents. Although well-suited for closed user groups, this model has its limitations when applied to unconnected populations.
Other proposed schemes combine the hierarchical structure with bidirectional cross-certification.
Keep certification chains as short as possible (a chain is always as vulnerable as its **weakest link**!).

Quick Review

Certification Topologies:

- **Cross-certification:** $CA_A\{CA_B\} \rightarrow CA_B\{B\}$
- **Hierarchical** (PEM/X.509): Global root
- **Graph** (PGP): Users = CAs
- **Hybrid:** Hierarchy + cross-cert
- Rule: Short chains!

Public Key Infrastructure (PKI)

Definition: Integrated infrastructure providing security services based on public-key cryptography.

Main Functional Entities

Certification Authority (CA):

- Creation and maintenance of certificates.

Certificate Repository:

- Accessible certificate directory (X.500, LDAP, WWW, DNS).

Certificate Revocation:

- Management of compromised/obsolete certificates (CRLs).

Key Backup and Recovery:

- Backup/restoration of lost keys.
- Media destruction, password loss, employee departure.
- Decryption private key (not signature).

Automatic Key Update:

- Key update after validity expiration.

Key and Certificate History:

- Retrieval of obsolete keys to decrypt old documents.

Cross-Certification:

- Validation of certificates from other PKIs (clients, suppliers, partners).

Non-Repudiation Support:

Demonstrates the proper execution of an authenticated transaction:

- Data origin authentication.
- Time-stamped data signature.
- Signed receipt of delivery.

Secure Time Stamping:

- Time reference accepted by all.

Client Software:

- PKI operations on the client side.
- Certificate management, signatures, decryption.
- Peripherals (smart cards, biometrics).

Original Text

Public Key Infrastructure (PKI): Definitions
Definition: A PKI is an integrated infrastructure enabling the provision of a set of security services based on public-key cryptography.
Functional Entities:

- **Certification Authority (CA):** Entity responsible for creating and maintaining certificates.
- **Certificate Repository:** Directory making certificates available to users and applications. Technologies used: X.500, LDAP, WWW servers, DNS, etc.
- **Certificate Revocation:** Compromised or obsolete certificates (notably CRL management).
- **Centralized Key Backup and Recovery:** Entity allowing key loss management due to various events: destruction of the material medium, password loss, employee departure, etc. Note that this procedure mainly applies to the decryption private key (as opposed to the signature private key).
- **Automatic Key Update:** After their validity expires.
- **Key and Certificate History.** This entity allows the retrieval of obsolete keys that were used to encrypt a document in the past.

- **Cross-Certification** with other PKIs (clients, suppliers, partners, etc.). This functionality allows (under certain constraints) the validation of certificates issued by other PKIs.
- **Non-Repudiation Support:** Value-added service providing the necessary evidence to demonstrate the execution of an authenticated transaction (data origin authentication, time-stamped data signature, signed receipt of delivery, etc.).
- **Secure Time Stamping:** Entity capable of providing a reference time accepted by all PKI participants. Main applications: non-repudiation, arbitration in case of conflicts, etc.
- **Client Software:** This functional entity allows all PKI operations on the client side. Examples: user certificate management, document signing, information decryption, management of specific peripherals (smart card readers, biometric devices, etc.).

Quick Review

PKI – Main Entities:

- CA: Certificate creation/maintenance
- Repository: Certificate storage
- Revocation: CRLs
- Backup/Recovery: Lost keys (decryption)
- Cross-cert: Validation of other PKIs
- Time stamping: Time reference
- Client: User operations

Advantages and Disadvantages

Advantages:

Security:

- Integrated environment without weak links.

All-in-One:

- Integration of multiple services: strong authentication, signatures, single sign-on, VPNs, B2C/B2B.
- Cost savings vs. “case-by-case” solutions.

Interoperability:

- Widespread standards (X.509, PKCS, OCSP).
- Compatible applications and devices.
- Possible inter-enterprise interoperability.

Disadvantages:

Implementation Cost:

- Expensive products.
- Rare skills.

Complexity:

- Complex implementation and management.
- **Alternative:** Outsourcing PKI service.

Original Text

PKI: Main Advantages and Disadvantages
Advantages

- **Security:** The integrated nature of a PKI allows the creation of a security environment without weak links.

- **All-in-One:** A PKI enables the integration and management of all security parameters for a wide range of services: strong entity authentication, document signing for non-repudiation, single sign-on, virtual private networks (VPNs), secure communications with clients/partners/suppliers (B2C, B2B), etc. The PKI represents a significant cost saving compared to “case-by-case” solutions.

- **Intra- and Inter-Enterprise Interoperability:** The main PKI products comply with widely adopted standardization norms (X.509, PKCS, OCSP, etc.). A large number of applications and hardware devices now conform to these standards. The possible compatibility between different PKI providers also allows (with some reservations) inter-enterprise interoperability.

Disadvantages

- **Implementation Cost:** Expensive products, rare skills.
- **Complexity**...but:
  - Outsourcing the “PKI service” is an alternative.

Quick Review

PKI:

Advantages:

- Integrated security
- All-in-one: multi-services
- Interoperability (standards)

Disadvantages:

- High cost
- Complexity
- Solution: Outsourcing