Public Key Cryptography

Cryptography (lecture portion)

Andreas Unterweger

School of ITS
Salzburg UAS

Winter term 2023/24

Recap

- Terms
  - Plaintext (further: chosen-plaintext attack)
  - Ciphertext (further: chosen-ciphertext attack)
  - Key
  - Encryption (function)
  - Decryption (function)
- Computational security
  - Brute-force key search
  - Current key lengths
- Symmetric vs. asymmetric cryptography

Overview of public key cryptography

- Asymmetric cryptography
  - One key is public (e.g., for encryption)
  - The second key is kept private (e.g., for decryption)
  - Notion of a public-private key pair
  - There is **no shared secret key**

  $\rightarrow$ Simplified key "exchange"
- Recall: Computationally secure schemes rely on (difficulty) assumptions
- Examples:
  - The Rivest-Shamir-Adleman (RSA) cryptosystem
  - Elliptic curve cryptography (only rough outlook)
- Advanced concepts based on asymmetric cryptography:
  - Digital signatures
  - Public-key infrastructure

Recap: Modular arithmetic and multiplicative groups I

- Modular arithmetic
  - The modulo operation
  - Modulus N
  - Congruence modulo N
  - Inverses and invertibility
- Multiplicative groups
  - What constitutes a group?
  - Closure
  - Existence of an identity element
  - Invertibility of all group elements
  - Exponentiation
  - Generators
  - Subgroups
  - Discrete logarithms

Recap: Modular arithmetic and multiplicative groups II

- $\mathbb{Z}_n^* := \{1, 2, \cdots, n-1\}$ with multiplication modulo $n \in \mathbb{P}$
  - Multiplicative group
  - Previous restriction: n must be prime so that all elements are invertible
  - Generalization (without proof): If $n \notin \mathbb{P}$, exclude all elements which are not invertible $\rightarrow$ groups which allow n to be composite (non-prime)

$\rightarrow$ Full definition: $\mathbb{Z}_N^* := \{a \in \{1, 2, \cdots, N-1\} | \gcd(a, N) = 1\}$ with multiplication modulo $N \in \mathbb{N} \backslash \{0, 1\}$

- Example $\mathbb{Z}_4^* = \{1, 3\}$ with multiplication modulo 4
  - Closure (follows from the remaining conditions below)
  - Existence of an identity element: 1
  - Invertibility: 1 (trivial), $3^{-1} \equiv 3 \pmod 4$ since $3 \cdot 3 \equiv 9 \equiv 1 \pmod 4$
  - 3 generates $\mathbb{Z}_4^*$ as $3^0 := 1 \pmod 4$ and $3^1 \equiv 3 \pmod 4$
- Example $\mathbb{Z}_8^* = \{1, 3, 5, 7\}$ with multiplication modulo 8

Definition and overview I

- Definition of $\phi(N) := |\mathbb{Z}_N^*|$
  - Specifies how many elements are in $\mathbb{Z}_N^*$
  - Also called Euler's totient function
  - If $N \in \mathbb{P}$, then $\phi(N) = N - 1$ (see previous lecture)
  - If $N \notin \mathbb{P}$, i.e., $N = \prod_i p_i^{e_i}$ distinct prime factors p_i and exponents $e_i \in \mathbb{N}\backslash\{0\}$, then $\phi(N) = \prod_i p_i^{e_i - 1} \cdot (p_i - 1)$; for proof see Katz (2021)
- Example: $\phi(4)$
  - From before: $\phi(4) := |\mathbb{Z}_4^*| = |\{1, 3\}| = 2$
  - Alternatively: $\phi(4) = \phi\left(2^2\right) = 2^{2-1} \cdot (2 - 1) = 2^1 \cdot 1 = 2$
- Example: $\phi(8)$
  - From before: $\phi(8) := |\mathbb{Z}_8^*| = |\{1, 3, 5, 7\}| = 4$
  - Alternatively: $\phi(8) = \phi\left(2^3\right) = 2^{3-1} \cdot (2 - 1) = 2^2 \cdot 1 = 4$
- Example: $\phi(15) = \phi\left(3^1 \cdot 5^1\right) = \left(3^0 \cdot (3 - 1)\right) \cdot \left(5^0 \cdot (5 - 1)\right) = (1 \cdot 2) \cdot (1 \cdot 4) = 2 \cdot 4 = 8$

Definition and overview II

→ Computing $\phi(N)$ requires knowledge about the factorization of N

- Recall: Factoring a large number is assumed to be hard

→ RSA assumption

  - If $\phi(N)$ is known or the factorization of N is known, some operations in $\mathbb{Z}_N^*$ are easy (more details later)
  - If $\phi(N)$ is not known and the factorization of N is not known, these operations in $\mathbb{Z}_N^*$ are hard (assuming there are no shortcuts)

  → If only Bob knows $\phi(N)$, only he can decrypt messages

- Overview of RSA
  - Key generation
  - Encryption
  - Decryption
  - Correctness
  - Practical considerations

RSA key generation

1. Generate two distinct large primes p and q of approximately equal length (details in the literature; out of scope)
2. Compute $N := p \cdot q$
   - Later operations are performed in $\mathbb{Z}_N^*$

   $\rightarrow$ $\phi(N) = (p-1) \cdot (q-1)$
   - N is **not** secret, but its factorization and $\phi(N)$ must be **secret**
3. Choose an $e \in \mathbb{Z}_N^*$ such that $\gcd(e, \phi(N)) = 1$
   - e is **public**

   $\rightarrow$ (e, N) is the **public key**
4. Compute d as the inverse of e, i.e., $e \cdot d \equiv 1 \pmod{\phi(N)}$
   - Algorithms for computation are out of scope (details in literature)
   - d must exist as e must have an inverse by definition
   - d must remain **secret**

   $\rightarrow$ (d, N) is the **secret key** (also: private key)

RSA encryption and decryption

- Plaintext and ciphertext are numbers in $\mathbb{Z}_N^*$
- Encryption and decryption are operations in $\mathbb{Z}_N^*$
- $c := E(k_{public}, m) = E((e, N), m) = m^e \pmod N$
- $m \stackrel{!}{=} D(k_{secret}, c) = D((d, N), c) = c^d \pmod N$

- Example with small numbers:
  1. Generate $p = 11$ and $q = 13$
  2. Compute $N = p \cdot q = 143$; $\phi(N) = (p-1) \cdot (q-1) = 10 \cdot 12 = 120$
  3. Choose $e = 7$ (observe that $\gcd(7, 120) = 1$)
  4. Compute $d \equiv 103 \pmod{120}$
  5. Encrypt $m = 19$: $c := m^e \pmod N = 19^7 \pmod N \equiv 46 \pmod N$
  6. Decrypt $m \stackrel{!}{=} c^d \pmod N = 46^{103} \pmod N \equiv 19 \pmod N$

Correctness of RSA

- $m \stackrel{!}{=} D(k_{secret}, E(k_{public}, m)) = D((d, N), E((e, N), m)) = D((d, N), m^e \pmod N) = (m^e)^d \pmod N = m^{e \cdot d} \pmod N$

$\rightarrow$ $m \stackrel{!}{\equiv} m^{e \cdot d} \pmod N$

$\rightarrow$ $m^{e \cdot d} \stackrel{!}{\equiv} m^1 \pmod N$

- m cyclically generates a subgroup of $\mathbb{Z}_N^*$ (repeats after subgroup size)
- $\mathbb{Z}_N^*$ has $\phi(N)$ elements
- All subgroups of $\mathbb{Z}_N^*$ have a size which is an integer divisor of $\phi(N)$ (details in Katz (2021)), e.g., 60 elements for $m = 19$ in $\mathbb{Z}_{143}^*$

$\rightarrow$ $m^{\phi(N)} \equiv m^1 \pmod N$

- Since $e \cdot d \equiv 1 \pmod{\phi(N)} \rightarrow m^{e \cdot d} \equiv m^1 \pmod N$

Practical concerns I

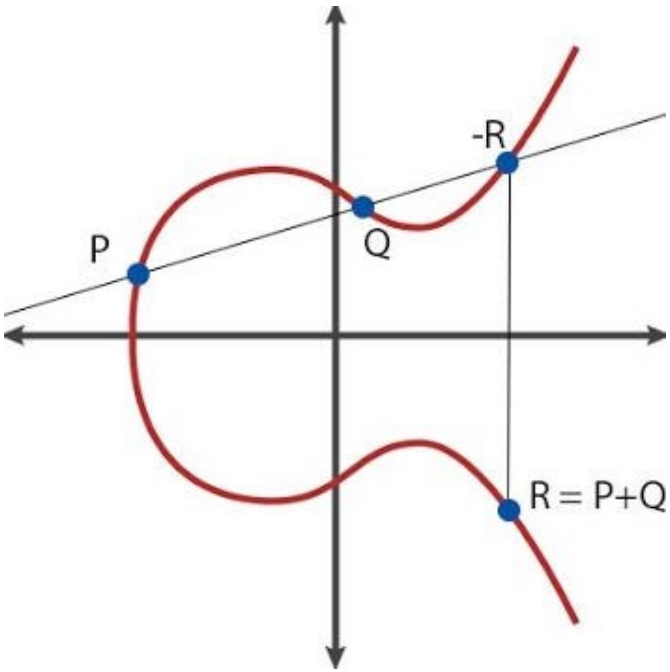
- "Textbook RSA" is **not** secure
  - Small messages can make attacks easy
  - $\rightarrow$ Use padding (bits) to remove structure from message
  - RSA with commonly used padding is secure against chosen plaintext, but not against chosen ciphertext attacks (without details)
- RSA cannot encrypt numbers larger than $N-1$ in $\mathbb{Z}_N^*$
  - Possibility 1: Split number into parts (similar to block ciphers) and use a random key for the actual encryption of all separate (padded!) parts
  - Possibility 2: Hybrid encryption (later) – encrypt or derive (padded!) symmetric key and continue communication with a symmetric cipher
- RSA cannot encrypt numbers outside of $\mathbb{Z}_N^*$
  - Convert binary string (message) into number (with padding!)
  - Theoretical corner case: $m \notin \mathbb{Z}_N^*$, e.g., $11 \notin \mathbb{Z}_{143}^*$ (correctness still holds and attacks are still hard; without details)

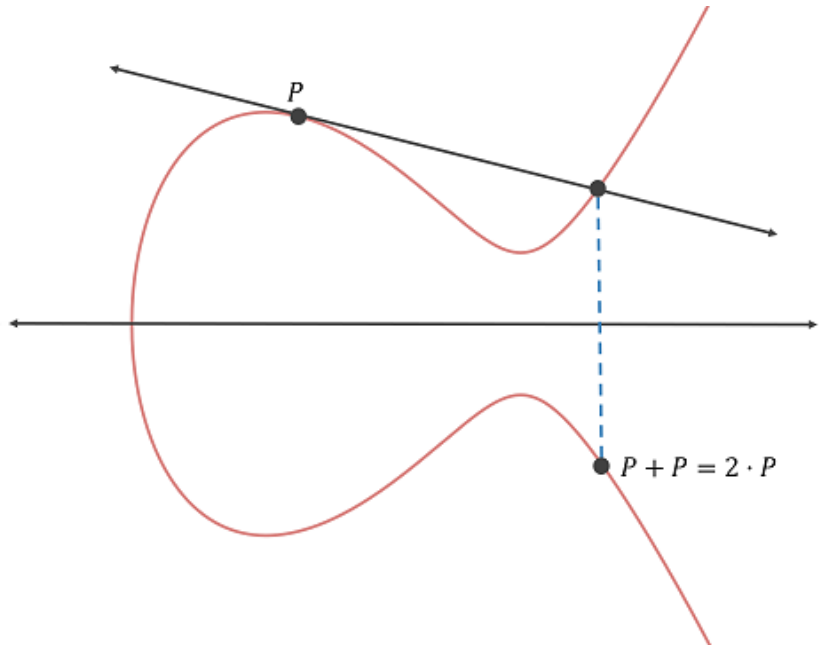
Practical concerns II

- RSA is relatively expensive to compute
  - Use RSA only for short messages, e.g., key exchange
  - Use symmetric ciphers for longer messages
- How to choose a safe size for the factoring modulus N
  - Asymmetric key sizes **cannot** be compared to symmetric key sizes
  - Asymmetric key sizes are comparable to (sub)group sizes
  - Recommendations at `https://www.keylength.com/en/compare/`
- Choices of values
  - p and q: Must be similar in size (otherwise advantage for an attacker)
  - e: Can be small, e.g., commonly 3 or 65537
  - d: Discouraged even though it could be chosen and e computed as its inverse; must not be too small (otherwise advantage for an attacker)

Outlook: Elliptic curve cryptography I

- Elliptic curves
  - Special curves (without details) whose points form a group
  - Special definition of addition: Line intersections of points with curve with special cases involving a point at infinity (without details)

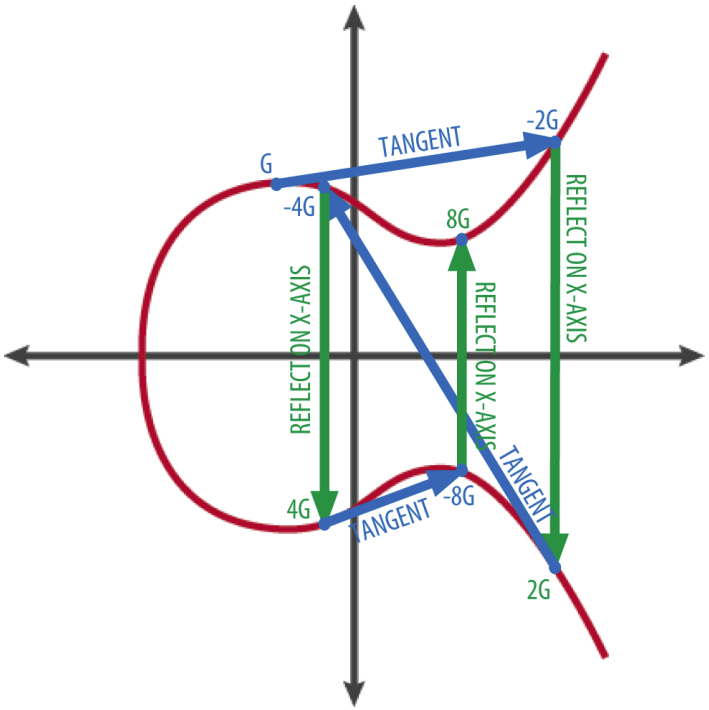


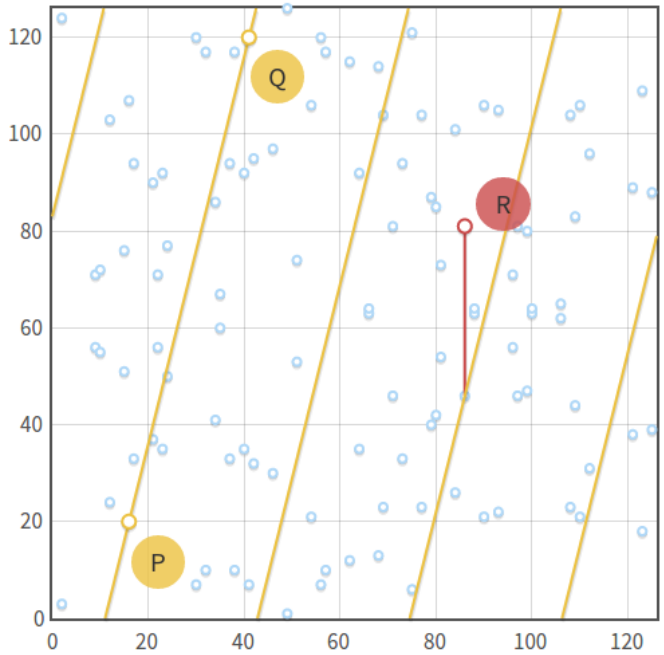


Sources: Reid, P.: Part 2 – Is Elliptic Curve Cryptography (ECC) a step towards something more – Understanding ECC. https://medium.com/@peterreid_12788/part-2-is-elliptic-curve-cryptography-ecc-a-step-towards-something-more-understanding-ecc-3c933d3922e (accessed on September 27, 2022), 2016; Yang, H.: Same Point Addition on an Elliptic Curve. http://www.herongyang.com/EC-Cryptography/Introduction-Same-Point-Addition-on-Elliptic-Curve.html (accessed on September 27, 2022), 2022.

Outlook: Elliptic curve cryptography II

- Multiplication is defined as repeated addition of a starting point
- Curve equation and point coordinates are actually in a prime modulus





Sources: Uszak, P.: How does ECC go from decimals to integers? https://crypto.stackexchange.com/q/48657 (accessed on September 27, 2022), 2017; Corbellini, A.: Elliptic Curve Cryptography: finite fields and discrete logarithms. https://andrea.corbellini.name/2015/05/23/elliptic-curve-cryptography-finite-fields-and-discrete-logarithms/ (accessed on September 13, 2022), 2015.

[4] Corbellini, A.: Elliptic Curve Cryptography: finite fields and discrete logarithms. https://andrea.corbellini.name/2015/05/23/elliptic-curve-cryptography-finite-fields-and-discrete-logarithms/ (accessed on September 13, 2022), 2015.

Outlook: Elliptic curve cryptography III

- Generators (forming sub-groups) can be defined based on multiplication analogous to multiplicative groups (without details)
- The discrete logarithm (based on multiplication) is assumed to be especially hard to compute in elliptic curve groups
  - $\rightarrow$ Use as a base for asymmetric cryptosystem (without details)
  - $\rightarrow$ Smaller key lengths required compared to cryptosystems relying on the discrete logarithm problem based on exponentiation (see recommendations at `https://www.keylength.com/en/compare/`)

[4] Corbellini, A.: Elliptic Curve Cryptography: finite fields and discrete logarithms. `https://andrea.corbellini.name/2015/05/23/elliptic-curve-cryptography-finite-fields-and-discrete-logarithms/` (accessed on September 13, 2022), 2015.

Overview of digital signatures

- Generated and added to a message by the signer/sender
- Can be checked by the receiver (and everybody else)
- Preserve message integrity instead of message privacy
- Guarantees
  - The signed message has not been modified
  - The signed message originates from the signer

  $\rightarrow$ Non-repudiation: A signer cannot claim **not** to have signed the message
- Use case example: Software update/patch verification
  - Alice (provider) publishes the update with a signature
  - Bob (user) can verify whether the signature is correct
  - If the signature is incorrect, e.g., due to a malicious modification, Bob can reject the update
- Signatures rely on public key cryptography and hashes

Recap: Asymmetric cryptosystems and hashes

- Asymmetric cryptography
  - One key is public (e.g., for encryption)
  - The second key is kept private (e.g., for decryption)
  - Notation: $c = E(k_{secret}, m)$ and $m \stackrel{!}{=} D(k_{public}, c)$
- Cryptographic hashes
  - create a fingerprint/digest/hash of a given message
  - map a variable-sized input to a fixed-sized output
  - are "one-way", i.e., computationally infeasible to invert
  - protect integrity (can be used to detect tampering)
  - Notation: $h = H(m)$

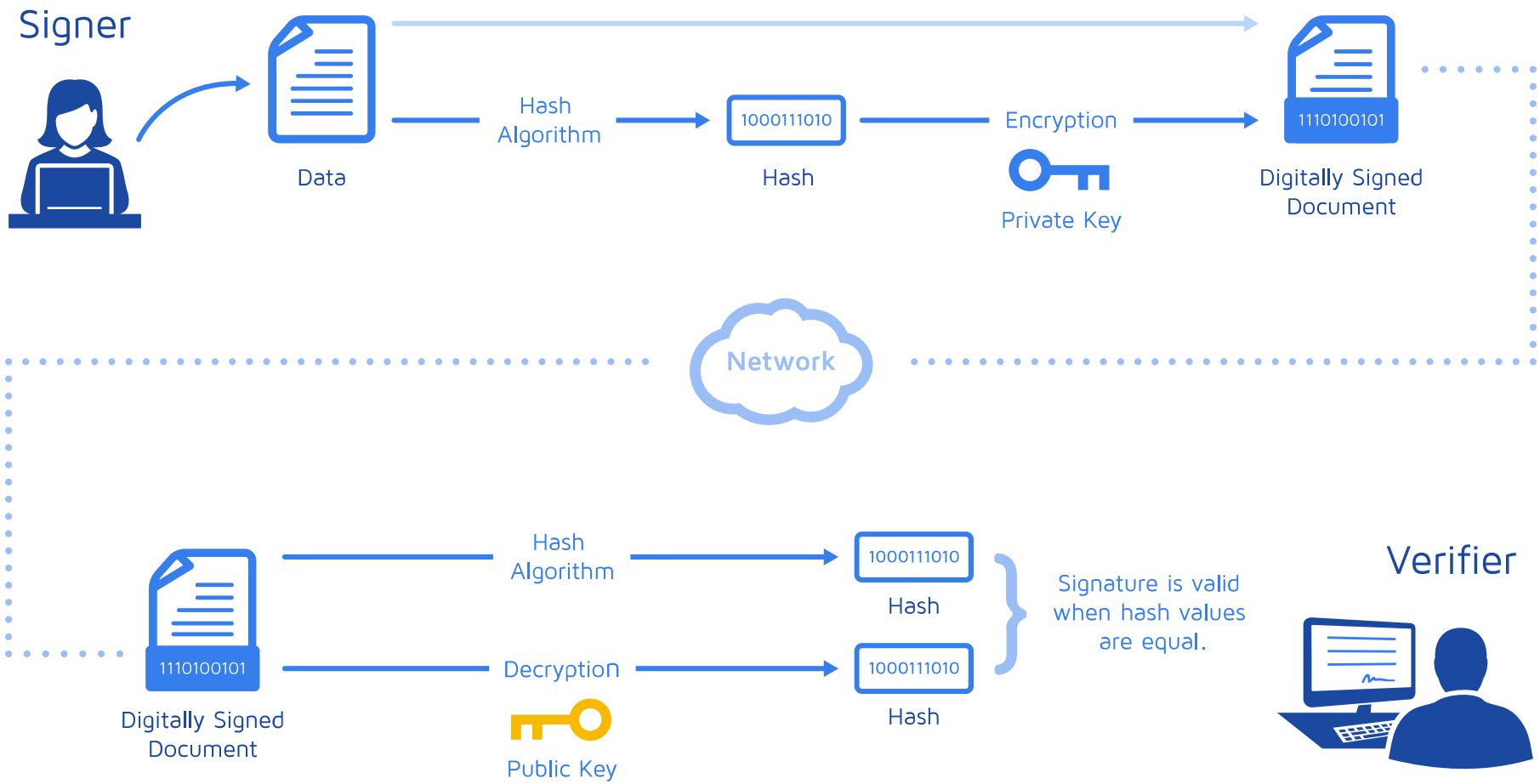
Signing

- Works with hashes of the original message
  - Less computational effort required for long messages
  - Signature is of constant size
  - Security relies on the security of the asymmetric cryptosystem **and** the security of the hash function
- Notation: $s = S(k_{secret}, m) := E(k_{secret}, H(m))$
- Signing steps
  1. Alice generates a public-private key pair
  2. Alice computes the hash of the message to be sent
  3. Alice encrypts the hash to obtain the signature
  4. Alice sends the signature, together with the message, to Bob

Signature verification I

- Verification steps (assuming Bob knows Alice's public key)
  1. Bob receives the message m' and its signature s'
  2. Bob computes the hash $h' = H(m)$ of the message m'
  3. Bob decrypts the hash from the signature: $h = D(k_{public}, s)$
  4. Bob compares the hash h' he computed with the hash h from the signature
- A signature is **only** valid if $h = h'$
- If $h = h'$, the message has not been changed **and** is from Alice
- If $h \neq h'$, **either**
  - the message has been tampered with (changing its hash)
  - the signature is not from Alice (but another key pair)

Signature verification II



Source: DocuSign, Inc.: Understanding digital signatures. `https://www.docusign.com/how-it-works/electronic-signature/digital-signature/digital-signature-faq` (accessed on September 13, 2022), 2022.

Practical considerations

- Recall: The security of a signature relies on the security of the asymmetric cryptosystem **and** the security of the hash function
  - → Use secure cryptosystems and secure hash functions
    - Use sufficient key lengths and output sizes, respectively
- Levels of signatures, e.g., European eSignatures
  - Simple: "Regular" digital signature
  - Advanced: Additionally linked to and identifying the signer (and more)
  - Qualified: Additionally created by a special signature device (and more)
- Identity issue: How to link Alice to her public key?

→ Public-key infrastructure

[1] European Commission: eSignature FAQ. https://ec.europa.eu/digital-building-blocks/wikis/display/DIGITAL/eSignature+FAQ (accessed on September 13, 2022), 2022.

Overview of public-key infrastructure

- Public-key infrastructure (PKI)
  - Architecture which allows associating public keys with their "owners"
  - Solves key management and distribution
  - Requires trust in a party or multiple parties
- Different models for PKI; focus on most widely used one(s)

- Focus topics:
  - Digital certificates
  - Certificate authorities and authority hierarchies
  - Alternative infrastructure concepts

Digital certificates I

- Recall: Digital signatures only allow verifying signers by their public key (they do not verify that the key belongs to any person or entity)

$\rightarrow$ Link between a person/entity and their public key is required

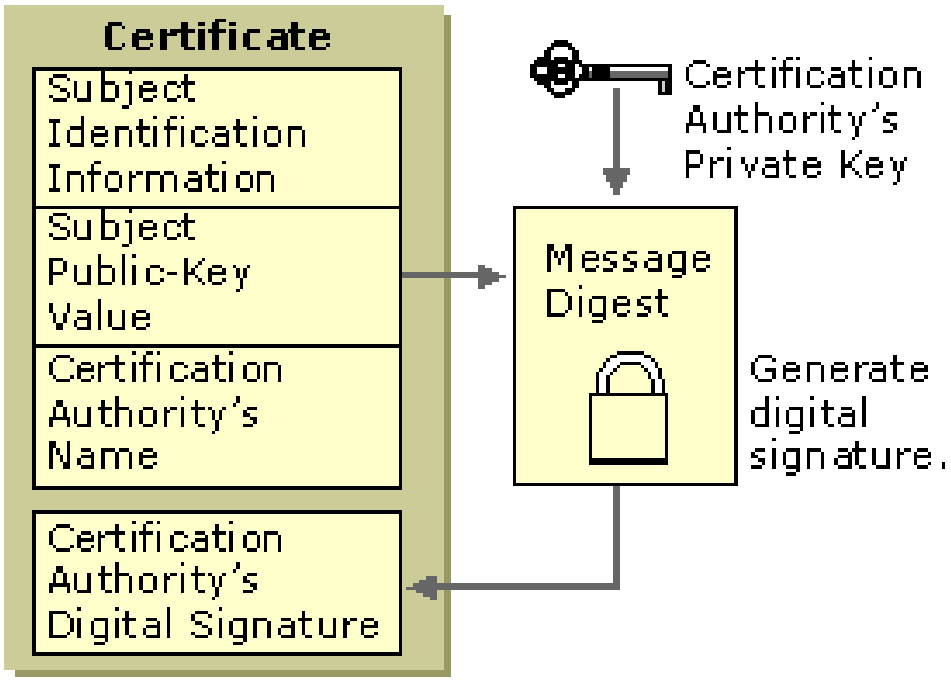
$\rightarrow$ Digital certificate: Associates public key with a person/entity

  - Assumption: Trusted party T whose public key is known by all parties
  - T signs a statement like "Alice's public key is k^A_{public}"
  - Bob can verify the statement given the public key of T
  - Notation: $C_{A \to T}$: T certifies A's key (with its own secret key)

- Notation: $C_{A \to T}$: A's key is certified by T
- $C_{A \to T} := S(k^T_{private}, \text{"Alices's public key is } k^A_{public}\text{"})$
- Certificate can be verified by everybody through its signature

Digital certificates II

- Practical signed message in a digital certificate: Separate fields for key(s) and identifying information with combined hash
- Example certificate standard: X.509

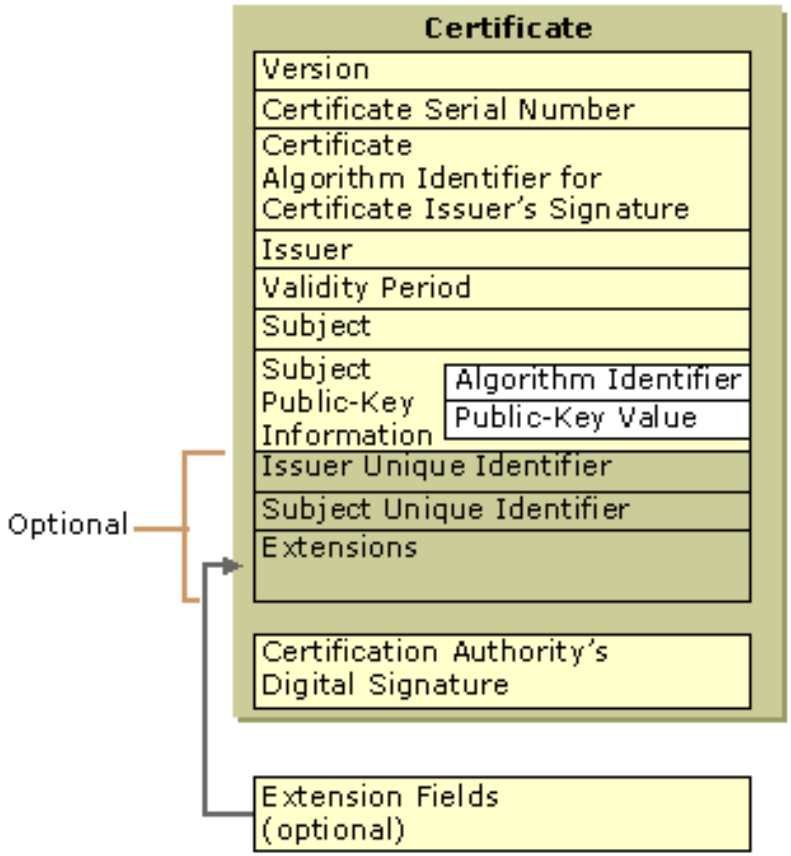


Source: Microsoft: Digital Certificates. https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-2000-server/cc962029(v=technet.10)?redirectedfrom=MSDN (accessed on September 13, 2022), 2012.

[2] Microsoft: Digital Certificates. https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-2000-server/cc962029(v=technet.10)?redirectedfrom=MSDN (accessed on September 13, 2022), 2012.

Digital certificates III

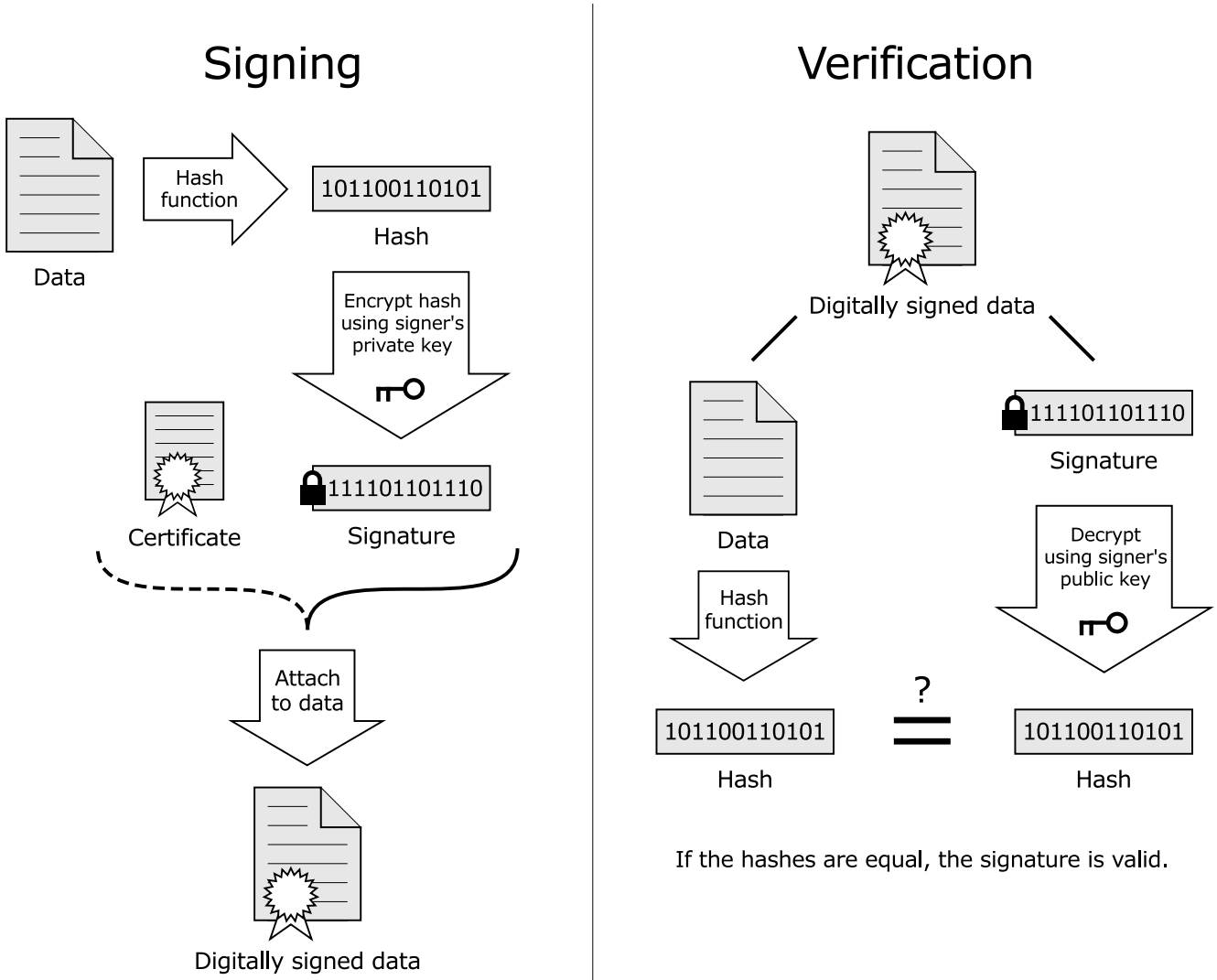
X.509 fields (simplified):



Source: Microsoft: Digital Certificates. `https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-2000-server/cc962029(v=technet.10)?redirectedfrom=MSDN` (accessed on September 13, 2022), 2012.

[2] Microsoft: Digital Certificates. `https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-2000-server/cc962029(v=technet.10)?redirectedfrom=MSDN` (accessed on September 13, 2022), 2012.

Digital signatures with certificates



Source: Acdx: Diagram illustrating how a simple digital signature is applied and verified. `https://commons.wikimedia.org/wiki/File:Digital_Signature_diagram.svg` (accessed on September 13, 2022), 2012.

Certificate authorities

- Certificate authorities (CAs)
  - are trusted to issue certificates
  - check the identity of the owner or entity
  - have their own private-public key pair (every actor is assumed to know the CA's public key)
- Limitations of having only one CA
  - Unlikely that **everybody** trusts the CA
  - Single point of failure: What to do on compromise?
  - How to *securely* get the CA's key?
- Solution: Use multiple CAs (multiple possibilities; example: CA hierarchies)

Certificate authority hierarchies I

- Certificate authority hierarchies (CA hierarchies)
  - Issuing of certificates is delegated to sub-CA(s)
  - Multi-level certificates from CA to sub-CA(s) to entity

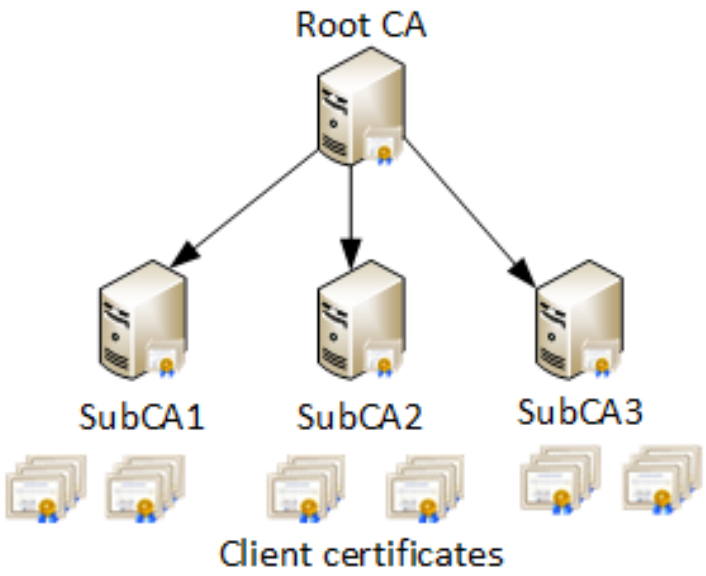
  $\rightarrow$ Hierarchical trust model
- Two-level example (more levels possible):
  - CA issues certificate for sub-CA (SCA): $C_{SCA \rightarrow CA}$

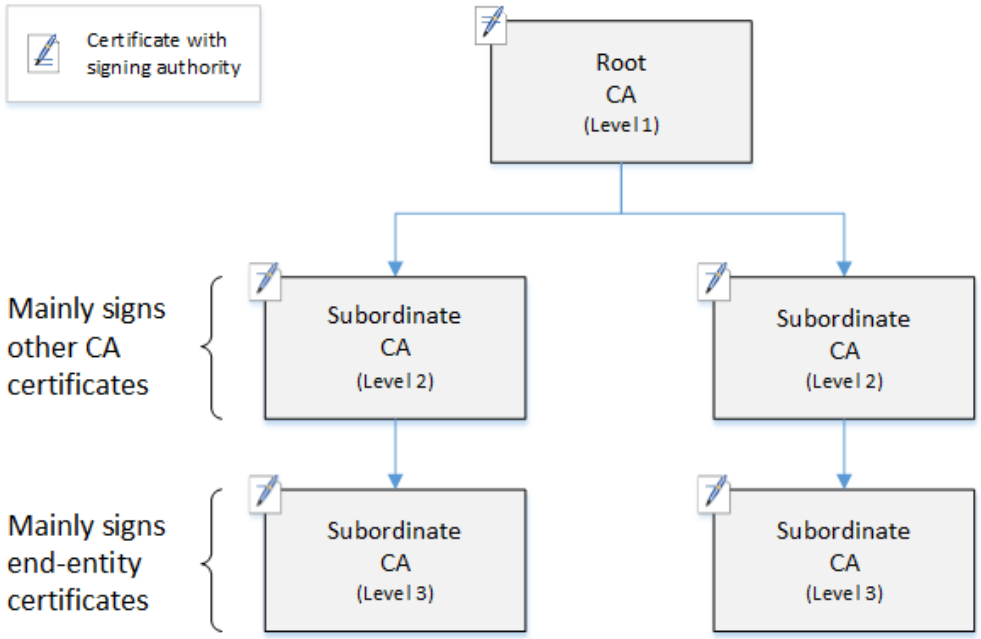
  $\rightarrow$ Sub-CA and its public key are signed by CA
  - Sub-CA issues certificate for Alice: $C_{A \rightarrow SCA}$

  $\rightarrow$ Bob can verify Alice's certificate through the certificate chain $(C_{A \rightarrow SCA}, C_{SCA \rightarrow CA})$

  $\rightarrow$ Bob must trust both, the sub-CA and the CA!

Certificate authority hierarchies II



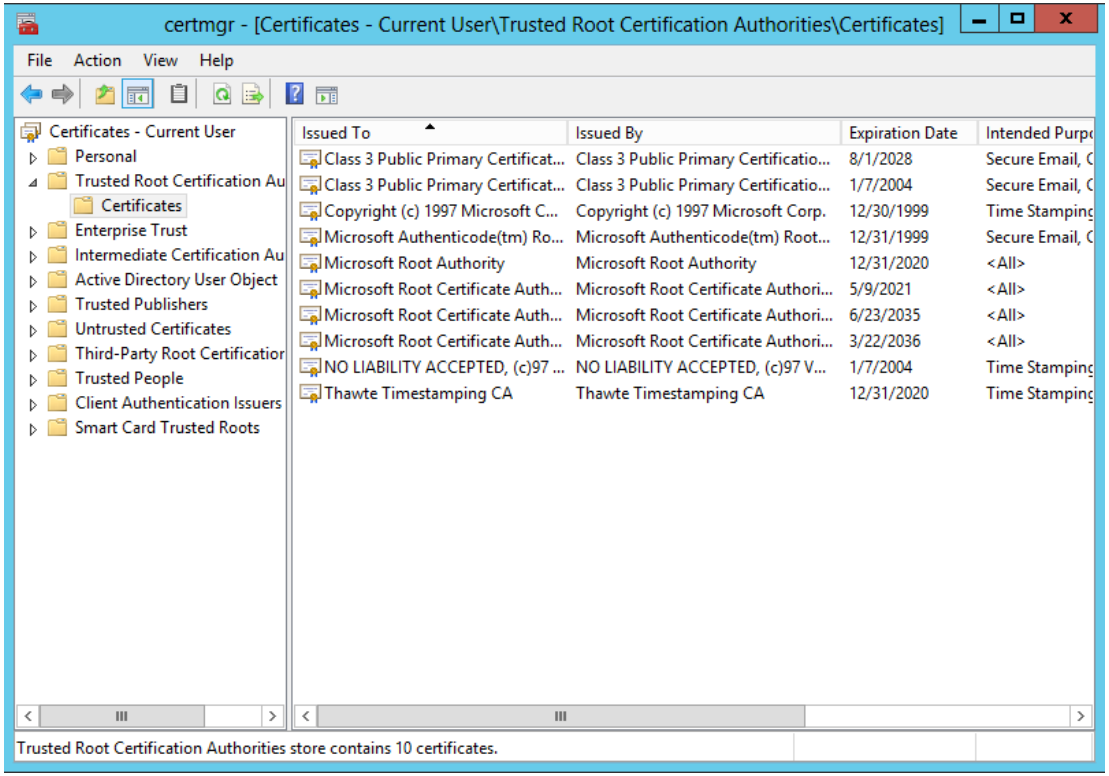


Sources: Podāns, V.: Certificate Chaining Engine — how it works. `https://www.sysadmins.lv/blog-en/certificate-chaining-engine-how-this-works.aspx` (accessed on September 13, 2022), 2011; Amazon Web Services, Inc.: Designing a CA hierarchy. `https://docs.aws.amazon.com/acm-pca/latest/userguide/ca-hierarchy.html` (accessed on September 13, 2022), 2022.

Certificate authority hierarchies III

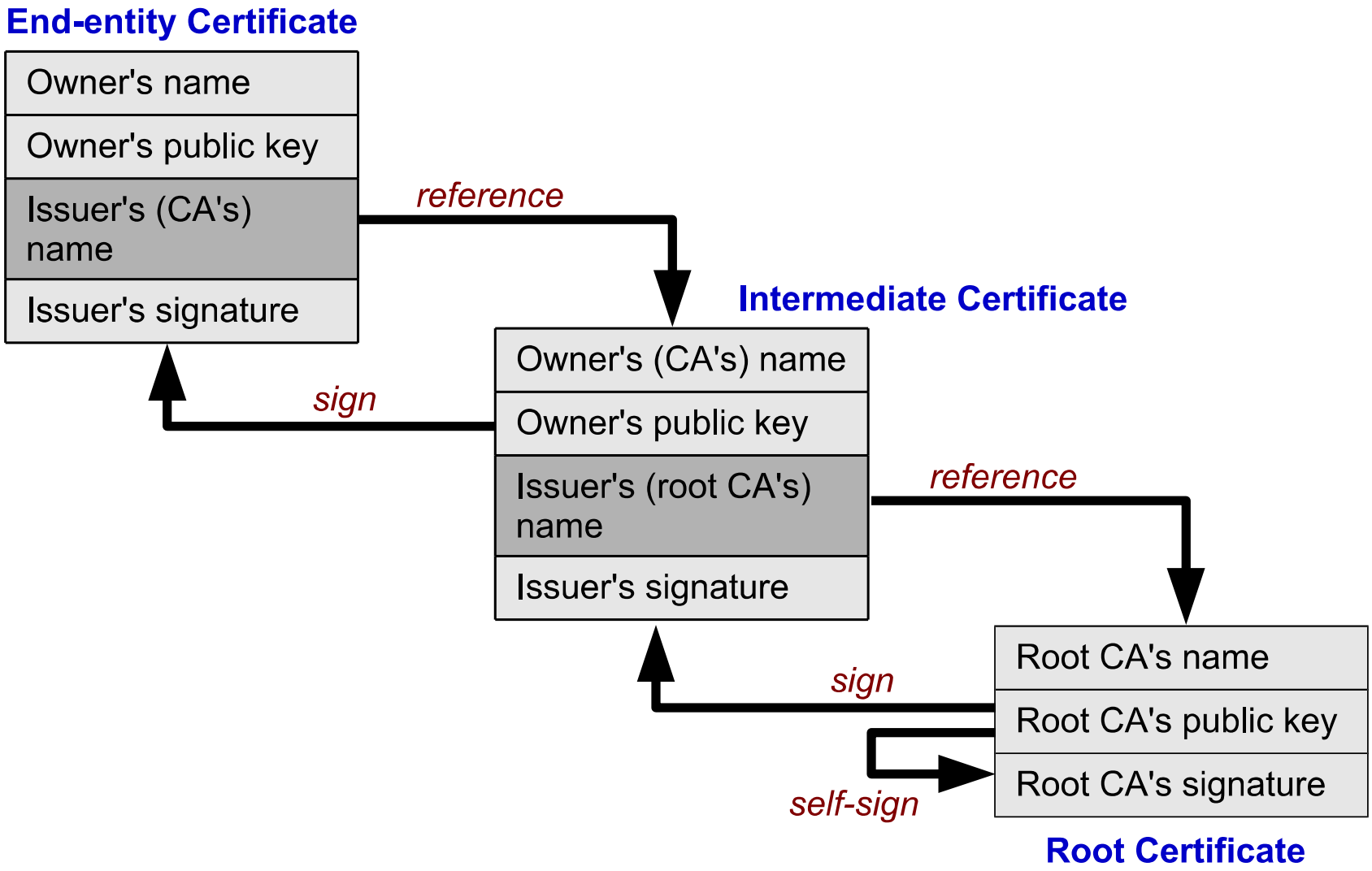
- Root CA(s) (top-level CA(s) of the hierarchy)
  - self-sign their own certificate(s)
  - come pre-installed with Web browsers, operating systems etc.

  → High level of trust required



Source: pdubs: Where to get root CA certificates for Windows Server now that Microsoft no longer updates them? `https://serverfault.com/q/541922` (accessed on September 13, 2022), 2013.

Certificate authority hierarchies IV



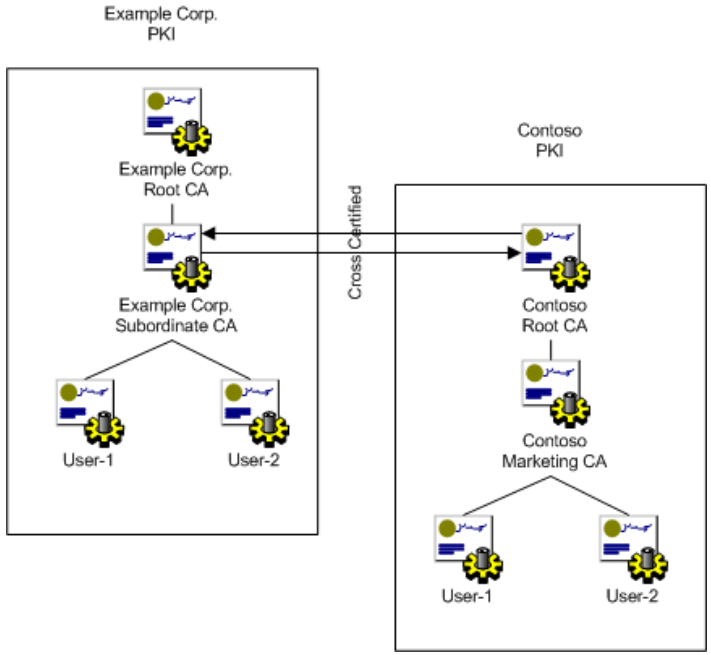
Source: Yanpas: Chain of trust SSL certificate. `https://commons.wikimedia.org/wiki/File:Chain_of_trust.svg` (accessed on September 13, 2022), 2017.

Practical concerns I

- There is no universal PKI
  - Different use cases and organizations have their own PKIs
  - → Users/Entities have different keys (and certificates) for each case
  - → Users/Entities use multiple PKIs at the same time
- Certificate revocation
  - Secret keys get stolen, lost, compromised
  - → Possibility to revoke corresponding certificate as fast as possible
  - Mitigation: Certificate revocation lists (CRLs) and/or online certificate verification through the CA or another trusted party
- Certificate expiration
  - Certificate has a validity date (range)
  - → Certificate becomes invalid outside of this date range
  - → Certificates need to be renewed (periodically)
  - For very short validity ranges, no CRL is required

Practical concerns II

- Cross-signing (CAs signing each others' certificates)
  - → Multiple valid certificate chains/paths exist
  - → More robust if certain certificates expire or are revoked

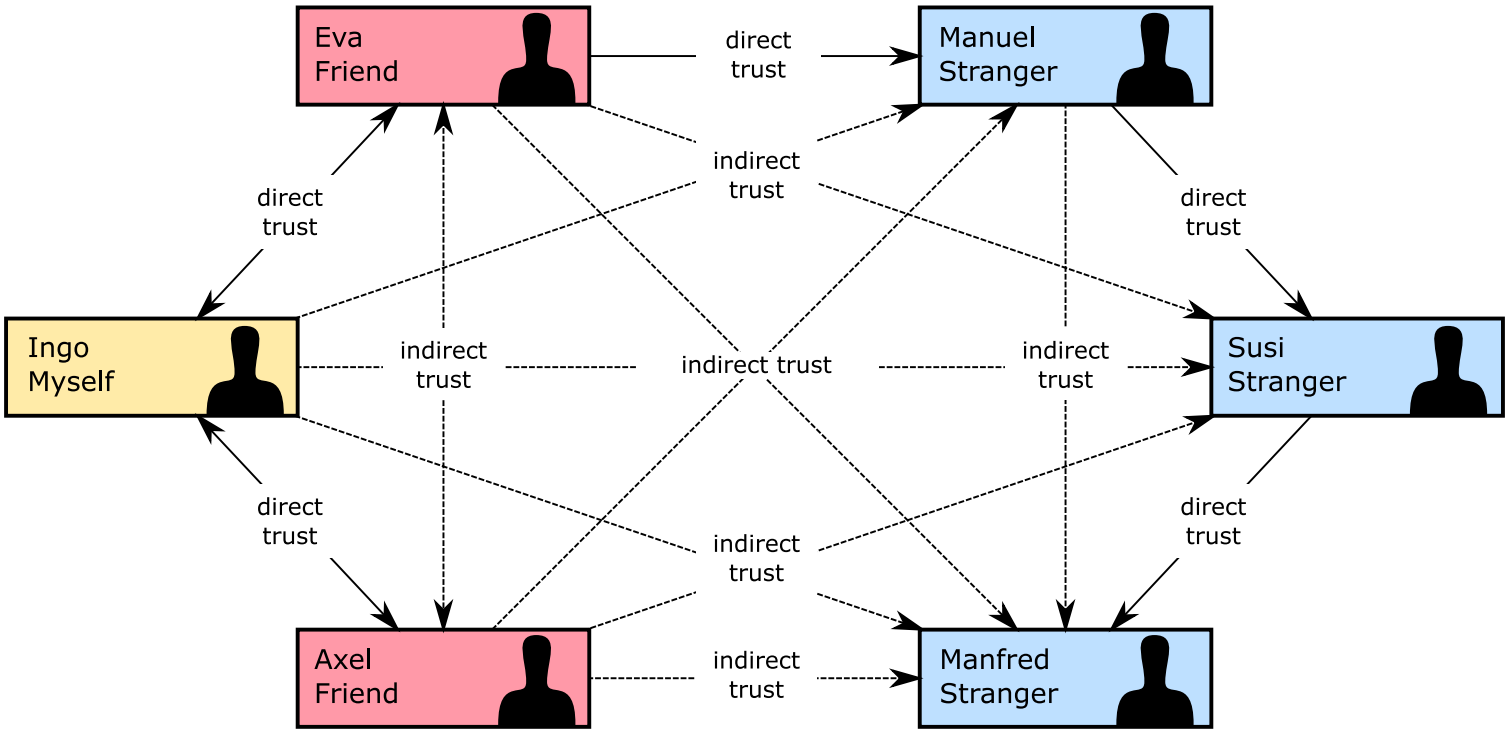


Source: Microsoft, Inc.: Cross Certification. `https://docs.microsoft.com/en-us/windows/win32/seccertenroll/about-cross-certification` (accessed on September 13, 2022), 2021.

[3] SSLTrust: Understanding Certificate Cross-Signing. `https://www.ssltrust.com.au/blog/understanding-certificate-cross-signing` (accessed on September 13, 2022), 2022.

Alternative infrastructure concepts

- Web of Trust
  - Everybody can issue certificates (there are no central authorities)
  - Each person or entity verifying a certificate must decide whether they trust any individual issuer (directly or indirectly)



Source: Kku: Web of Trust network with levels of trust. `https://commons.wikimedia.org/wiki/File:Web_of_Trust-en.svg` (accessed on September 13, 2022), 2019.

Thank you for your attention!

Questions?