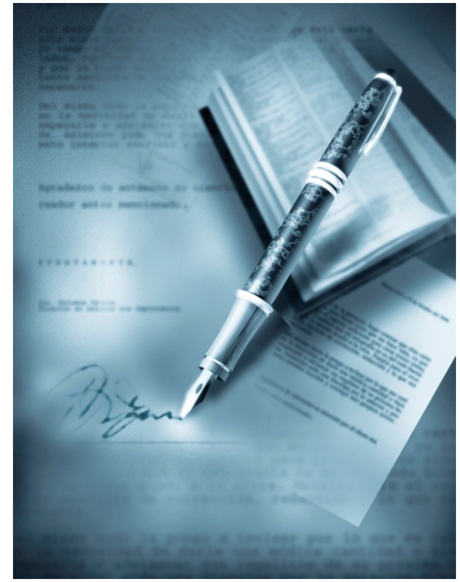


IT Security

Chapter 3: Checksums and Digital Signatures

Part 1

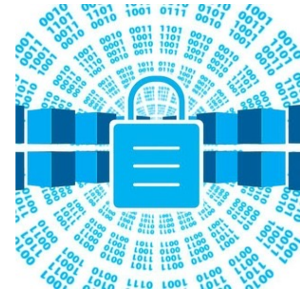
- Hash functions
- Message Authentication Code MAC
- Signature procedure
- PKI

Prof. Dr. Reiner Hüttl, TH Rosenheim, 15.03.2023

What do we want to learn?

- What is the meaning of hash functions?
- What is a cryptographic hash function, such as MAC?
- How does a digital signature work?
- What is the legal significance of electronic signatures?
- What do you have to consider in the practical implementation?
- What are the components of a PKI?
- How is a certificate structured, how do you verify it?

How do I ensure the integrity of my data?



- Problem:
  - I have a lot of data and want a short checksum to make sure the data has not been modified.
- Possible solutions
  - Hashing
  - Cryptographic hashing
  - Digital signatures

Not safe against tampering!

Hash Functions

BUT: Hash functions are not encryption!

- One-way encryption functions:
  Mapping of variable length input to "unique" output with 0 fixed length.

- Serves as checksum, fingerprint, message digest

- Properties
  - small change in message results in large change in hash value
  - message cannot be reconstructed from hash value (function cannot be inverted)
  - hash value as unique as possible
    collision resistant: it is practically impossible to find two values x1 and x2 with f(x1) = f(x2)
  - hash value can be calculated fast and by everyone

Not recomended anymore

- Algorithms: MD5 (128 bit), SHA-1 (160 bit),
  SHA-2 (SHA-256 (256 bit), SHA-384, SHA-512, SHA-224)
  SHA-3

Security of hash functions

- Insecure "classic" methods: MD5, SHA1, RIPEMD

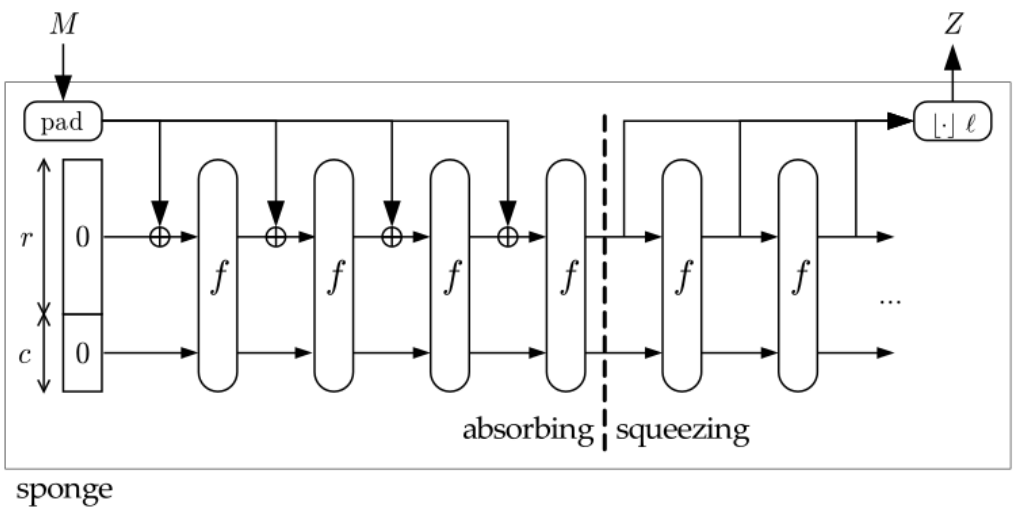
- Recommendations and key lengths for cryptographic methods by BSI

Methods	Recommended algorithms
cryptographic strong hash functions	SHA-256, SHA-512/256, SHA-384 und SHA-512, SHA3-256, SHA3-384, SHA3-512
MAC methods	HMAC>= 128, CMAC>=128, GMAC>=128
Signature methods	RSA 3000, DSA 3000 , ECDSA 250, Merkle signatures

https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TG02102/BSI-TR-02102-1.pdf?__blob=publicationFile&v=6

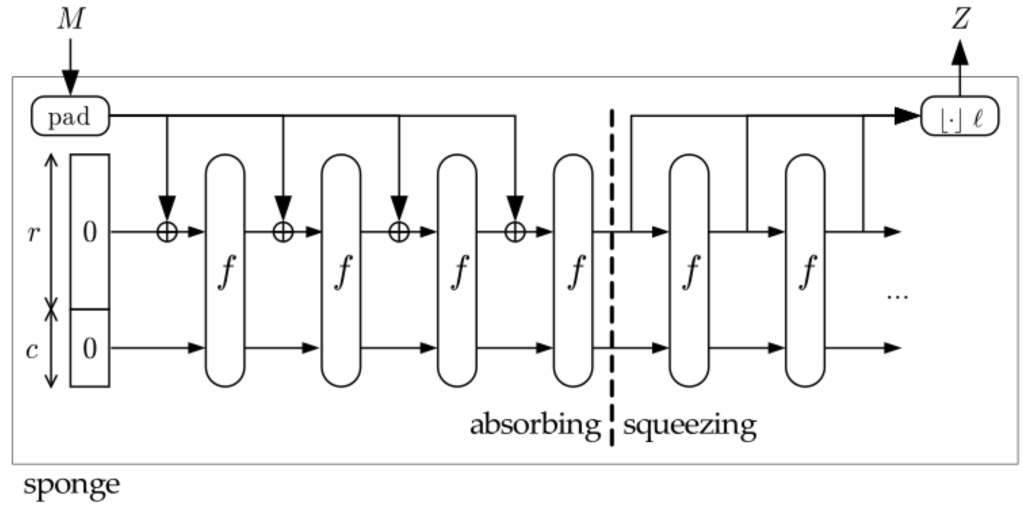
New hash standard since 2012: SHA-3 (Keccak)

- Winner from a competition by the NIST (National Institute of Standards and Technology)
  http://csrc.nist.gov/groups/ST/hash/sha-3/
  https://keccak.team/index.html
- Important requirement: hash must be quickly computable
- Output size is variable (224, 256, 384, 512 bit)
- Security can be influenced by capacity using a sponge function
  (more security -> less performance)



Quelle: http://sponge.noekeon.org/

Details of SHA-3 (Keccak)



Quelle: http://sponge.noekeon.org/

- Absorbing:
  - state is divided into two parts: Capacity (c bits) and Bit-Rate (r bits).
  - input blocks M are padded (filled up)
  - r-bit part is XORed with state, c-bit part remains unchanged
  - then state is mixed with f (Keccak permutation).
- Squeezing
  - Hash value is extracted from r-bit portion of state
  - If length is not sufficient, state is permuted several times and further outputs are extracted after each permutation.
- Visualization see CrypTool 2 (https://www.cryptool.org/de/ct2/)

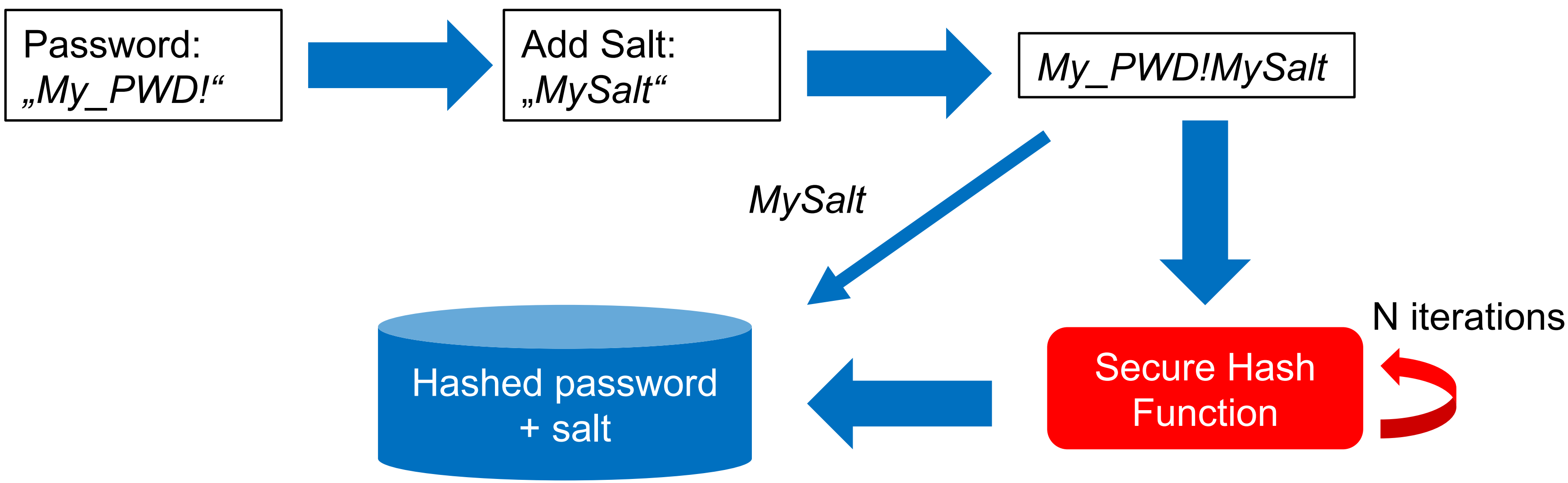
Hash functions for password hashing

- Problems
  - normal Hash functions are very fast and support brute force attacks
  - same password results in same hash value

- Criteria for Secure Hash functions https://www.password-hashing.net/cfh.html
  - add cost parameter to tune time and/or space usage
  - provide cryptographic security (no speed up for hackers possible)
  - combine it with a **Salt** to produce different hash values for same password
    Salt: random data as additional input for the hash function,
    generate new salt for each password

- Secure Password Hashing Algorithms
  - Bcrypt, Scrypt, Argon2, PBKDF2

Process for secure password hashing



Data authentication

- Data authentication means cryptographic procedures that guarantee that transmitted or stored data has not been modified by unauthorized persons.
  - it is based on cryptographic keys that are used to calculate checksums.
  - there are symmetric and asymmetric procedures
- Two security goals can be achieved with data authentication
  - ensuring the **integrity** of data
  - securing the **non-repudiation** of a message
    → this is only possible with digital signatures

https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TG02102/BSI-TR-02102-1.pdf?__blob=publicationFile&v=6

Message Authentication Code (MAC)

- A hash function that additionally uses a secret key
- The key is known only to the two communication partners
- This enables the detection of unauthorized modifications (**integrity**)
- This enables the verification if data have an authentic origin (**authenticity**)

First attempt for a MAC

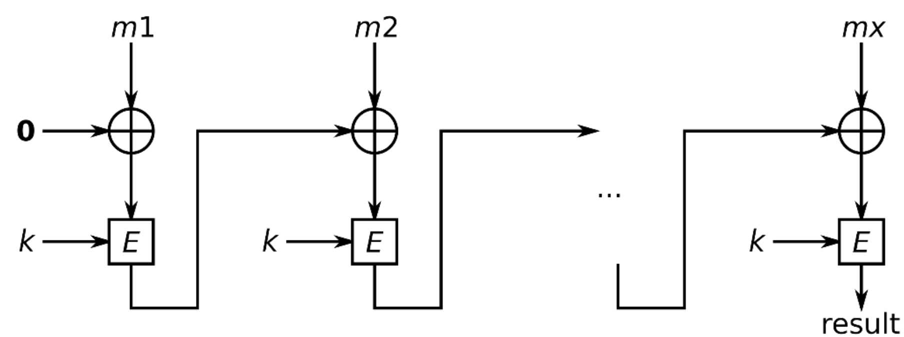
- Simple concatenation *hash(Secret || Daten)* = MAC
- Problem:
  - if the attacker knows *length(secret || data)*
  - then he can compute for some hash functions *hash(secret || data || more data)* **without** knowing the secret!
- This is called: **Length extension attack**
  https://en.wikipedia.org/wiki/Length_extension_attack
- Vulnerable are e.g. SHA1, SHA2, not vulnerable is SHA3
- Problem: you must create a hash, which can only be calculated with a secret.
  - one possible solution: HMAC

HMAC (Hashed MAC)

- HMAC is a popular MAC
- $HMAC\ (K,m) = H\ ((K \oplus opad) \parallel H\ ((K \oplus ipad) \parallel m))$
  - K is the secret key
  - block size is 64 bytes: either pad K with zeros or reduce to 64 bytes with a hash function.
  - opad, ipad are constants in block size
- A video illustrating HMAC
  https://www.youtube.com/watch?v=BjInMA-b8ZE

CBC-MAC (Cipher Block Chaining MAC)

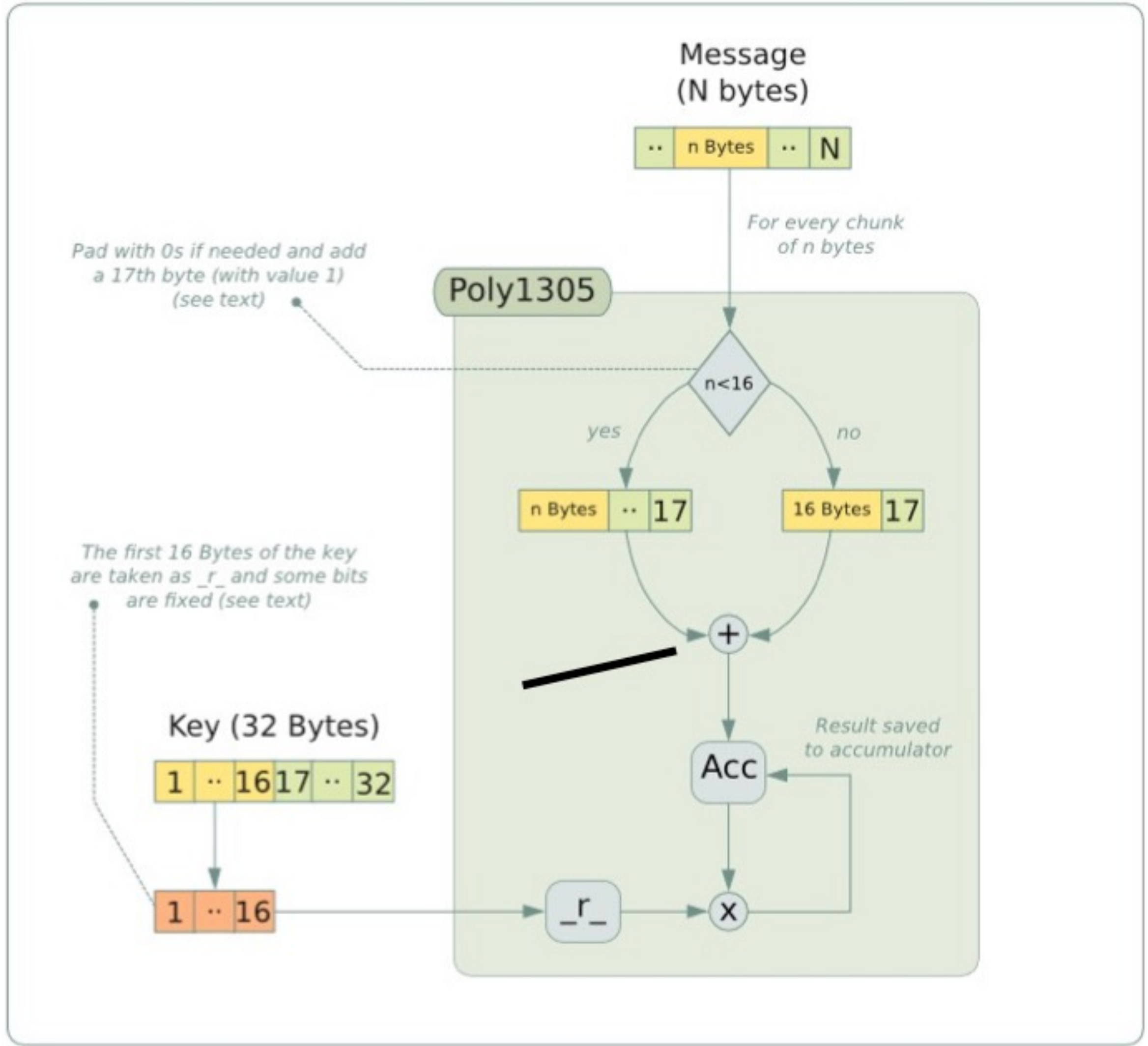
- CBC-MAC is a technique to calculate a MAC using a block cipher method.
- It uses block cipher method in CBC mode
- The IV is set to 0, the MAC is the last encrypted block
- $c_1 = (m_1 \oplus 0) \oplus K \qquad c_x = (m_x \oplus c_{x-1}) \oplus K$
- CBC-MAC is only safe for messages of fixed/known length, otherwise a length extension attack is possible.



Quelle: By Benjamin D. Esham (bdesham) - Own work based on: Cbcmac.png by en:User:PeterPearson.Own work by bdesham using: Inkscape., Public Domain, https://commons.wikimedia.org/w/index.php?curid=2277179

Poly1305 MAC

- Poly1305 is a MAC which uses a 32 Byte secret key and generates a 16 Byte authenticator
- Poly1305 is faster than HMAC



Quelle: https://www.adalabs.com/adalabs-chacha20poly1305.pdf

Applications and limitations of MAC

- Modern ciphering methods combine encryption with MAC calculation
  - Counter Mode **CTR with CBC-MAC**
  - **GCM** (see chapter 2: GCM is a cipher with MAC calculation)
  - **ChaCha20 with Poly1305**

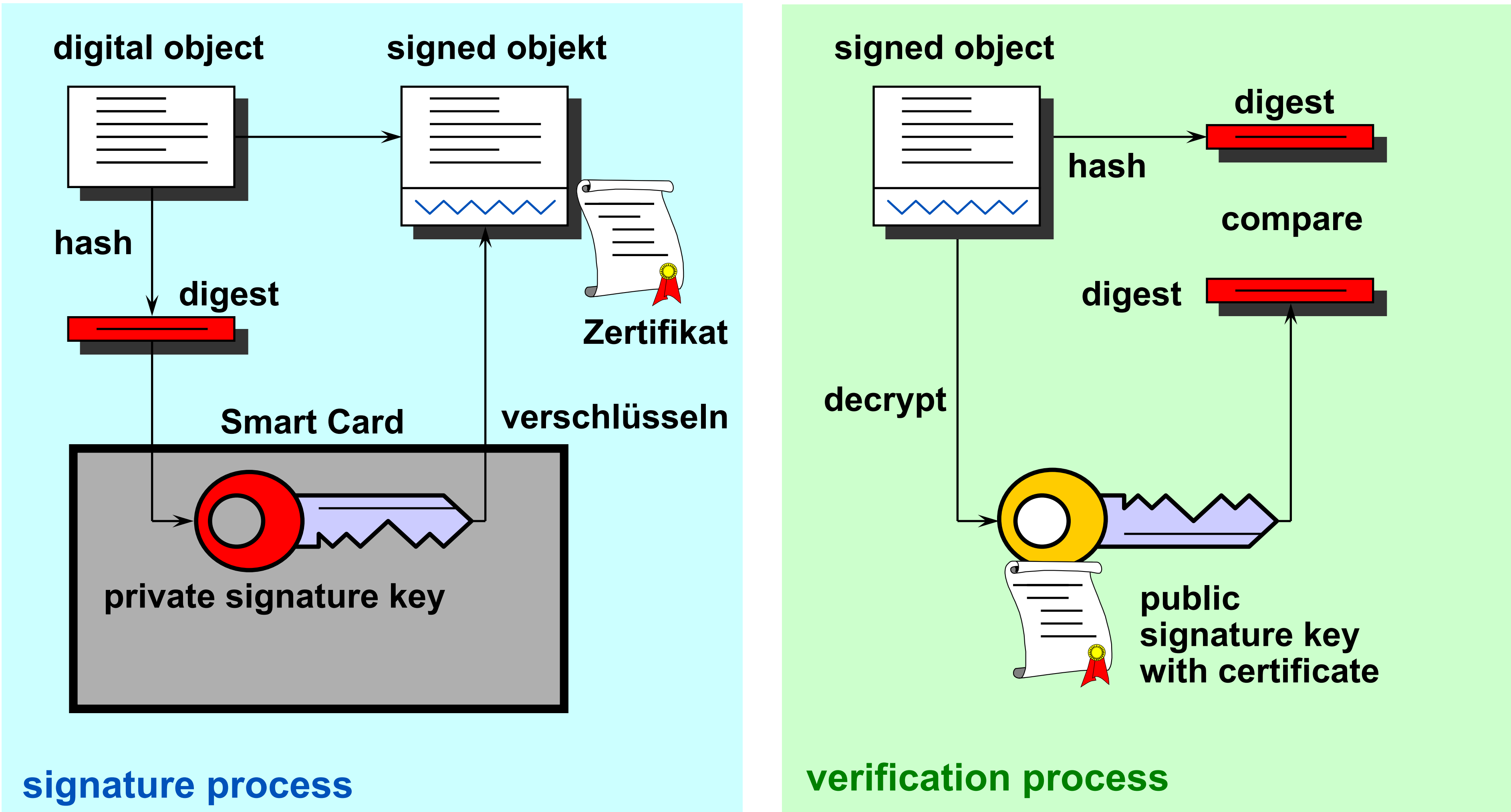
- Application of MAC
  - attack detection for file systems (Filesystem Intrusion Detection)
  - securing software packages (patch, update)
  - securing of communication protocols (e.g. TLS)

- A MAC secures the integrity and authenticity of a message
  - but it lacks the evidential value for non-repudiation
  - it cannot be verified by a third-party authority
  - it is based on symmetric cryptography
  - it lacks certificates

Digital signatures ensure the integrity of data and prove authenticity

- A Digital Signature contains
  - time from a timestamp service
  - person or place (server name)
  - signed digest of the document
- A Digital Signature proves
  - integrity of the document (what)
  - who signed and when

The procedure of the Digital Signature

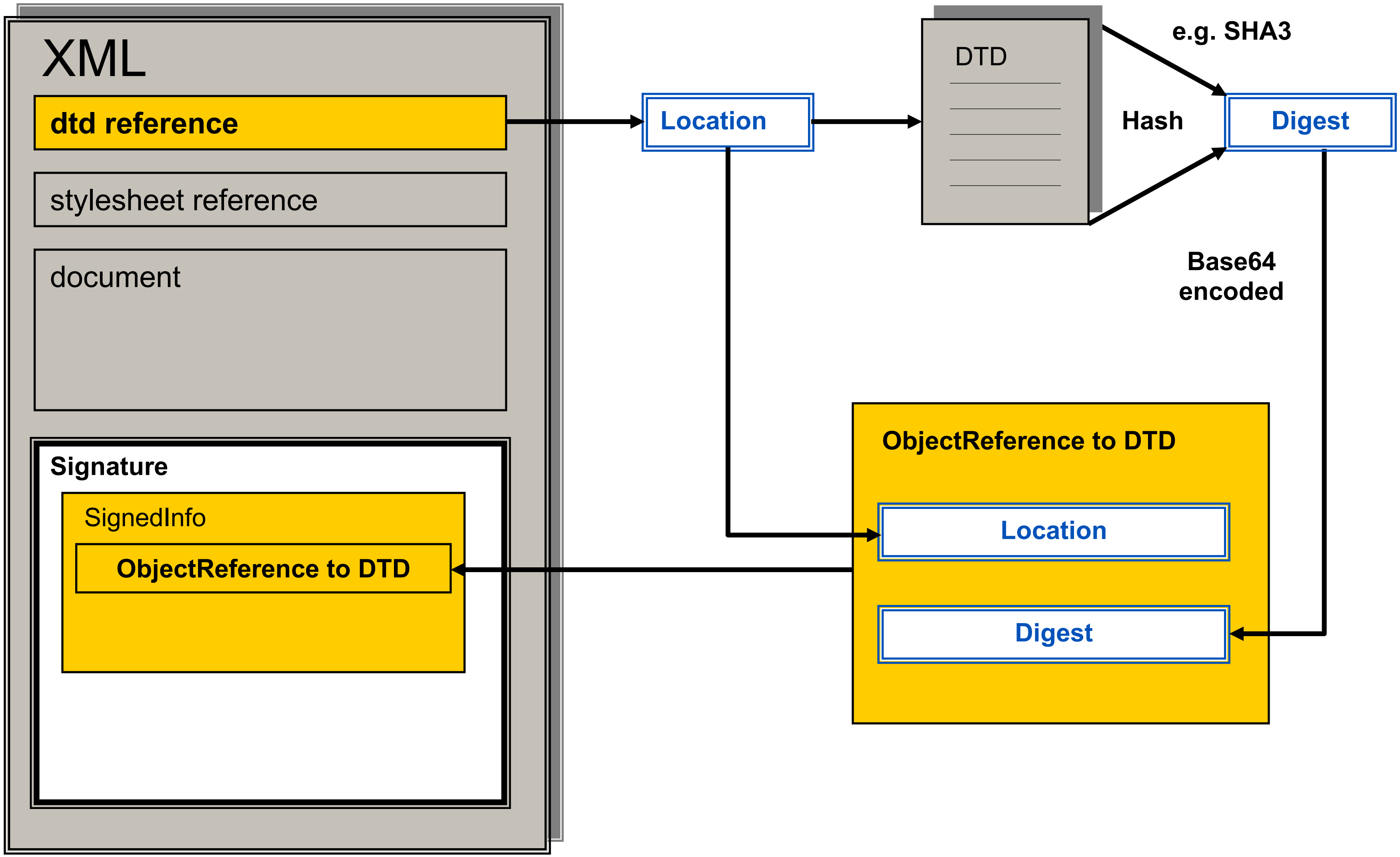


Standard for XML signatures

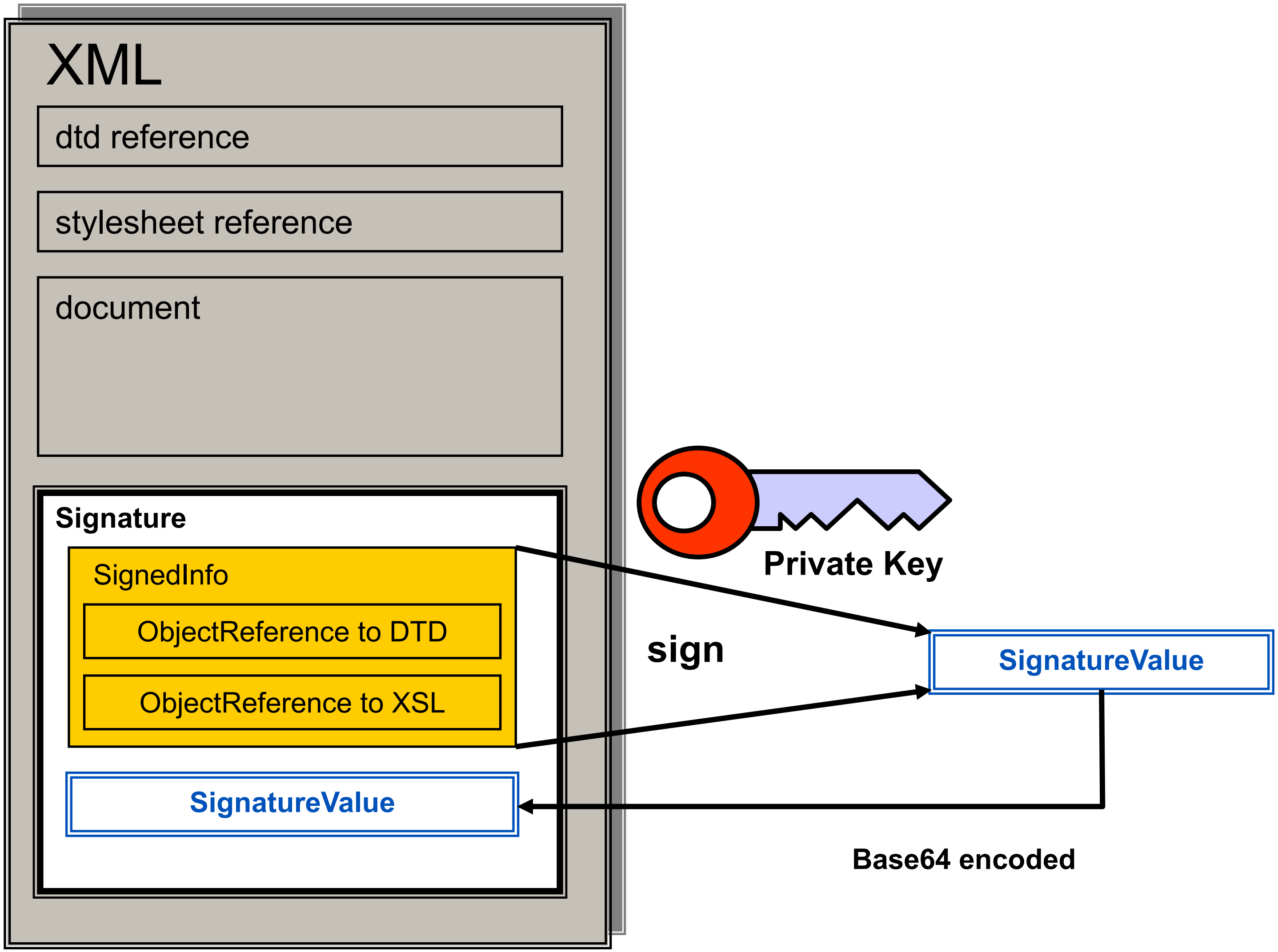
- Defines rules and syntax for signature
  - of whole XML documents
  - of parts of XML documents
  - any other files
  - literature: http://www.w3.org/Signature/

- Three ways to integrate XML signatures
  - **Detached Signature**: The signature is detached from the document and not embedded in the document.
  - **Enveloped Signature**: The signature is embedded in the document.
  - **Enveloping Signature**: The signature has the function of an envelope. It encloses the entire XML document.

XML-Signature: Digest calculation, reference creation



XML-Signature: Signature of the objekt references



Components of an XML signature

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Envelope xmlns="urn:envelope">
  <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
    <SignedInfo>
      <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n- 20010315#WithComments"/>
      <SignatureMethod Algorithm="http://www.w3.org/2000/09/ xmldsig#rsa-sha1"/>
      <Reference URI="">
        <Transforms>
          <Transform Algorithm="http://www.w3.org/2000/09/ xmldsig#enveloped-signature"/>
        </Transforms>
        <DigestMethod Algorithm="http://www.w3.org/2000/09/ xmldsig#sha1"/>
        <DigestValue>uooqbWYa5VCqcJCbuymBKqm17vY=</DigestValue>
      </Reference>
    </SignedInfo>
    <SignatureValue> KedJuTob5gtvYx9qM3k3gm7kbLBwVbEQRl26S2tmXjqNND7MRGtoew== </SignatureValue>
    <KeyInfo>
      <KeyValue>
        <RSAKeyValue>
          <Modulus>
              4IlzOY3Y9fXoh3Y5f06wBbtTg94Pt6vcfcd1KQ0FLm0S36aGJtTSb6pYKfyX7PqCUQ8wgL6xUJ5GRPEsu9gyz8
              ZobwfZsGCsvu40CWoT9fcFBZPfXro1Vtlh/xl/yYHm+Gzqh0Bw76xtLHSfLfpVOrmZdwKmSFKMTvNXOFd0V18=
          </Modulus>
          <Exponent>AQAB</Exponent>
        </RSAKeyValue>
      </KeyValue>
    </KeyInfo>
  </Signature>
</Envelope>
```

Document to sign

Processing information

Referenced data with processing information and digest

Base64 encoded signature value over the SignedInfo element

Information about the key to verify the signature

XML tags for XML signature

Canonicalization

- XML documents with semantically the same content can be represented in different ways:
  <myelement attr="123"/>
  <myelement attr="123"></myelement>
- This gives the signature a completely different value
- Therefore, a standardized representation is necessary: Canonization
- Literature: https://www.w3.org/TR/xml-c14n/