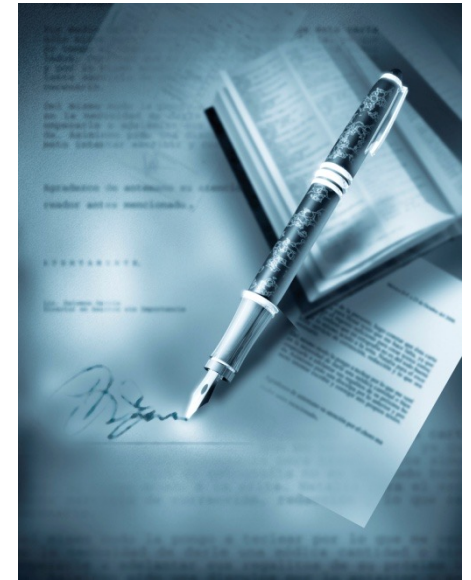


IT-Sicherheit



Kapitel 3: Prüfsummen und Digitale Signaturen

Teil 1

- ▶ Hash-Funktionen
- ▶ Message Authentication Code MAC
- ▶ Signaturverfahren
- ▶ PKI



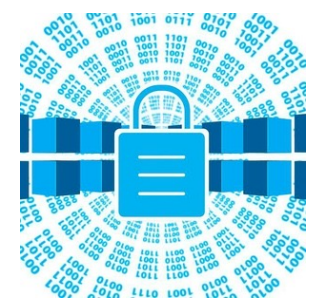


Worum geht es?



- ▶ Welche Bedeutung haben Hash Funktionen?
- ▶ Was ist eine kryptographische Hash-Funktion, wie z.B. MAC?
- ▶ Wie funktioniert eine digitale Signatur?
- ▶ Wie ist die rechtliche Bedeutung von elektronischen Signaturen?
- ▶ Was muss man bei der praktischen Umsetzung alles beachten?
- ▶ Was sind die Bestandteile einer PKI?
- ▶ Wie ist ein Zertifikat aufgebaut, wie verifiziert man es?

Wie sichere ich die Integrität meiner Daten?



- ▶ Problem:
 - ▶ Ich habe viele Daten und will eine kurze Prüfsumme, um sicherzustellen, dass die Daten nicht verändert wurden
- ▶ Mögliche Lösungen
 - ▶ Hashing
 - ▶ Kryptographisches Hashing
 - ▶ Digitale Signaturen

Nicht sicher gegen Manipulationen!



Hash Funktionen

ABER: Hash-Funktionen
sind keine Verschlüsselung!



- ▶ Einwegverschlüsselungsfunktionen:
Abbildung einer Eingabe variabler Länge auf „eindeutige“
Ausgabe fester Länge
- ▶ Dient als Prüfsumme, Fingerprint, Message Digest
- ▶ Eigenschaften
 - ▶ Kleine Änderung in Nachricht ergibt große Änderung im Hashwert
 - ▶ Nachricht aus Hashwert nicht rekonstruierbar (Funktion nicht invertierbar)
 - ▶ Hashwert möglichst eindeutig
kollisionsresistent: es ist praktisch unmöglich zwei Werte x_1 und x_2 zu finden mit $f(x_1) = f(x_2)$
 - ▶ Hashwert schnell und von jedem zu berechnen
- ▶ Algorithmen: MD5 (128 Bit), SHA-1 (160 Bit),
SHA-2 (SHA-256 (256 Bit), SHA-384, SHA-512, SHA-224)
SHA-3



Sicherheit von Hash-Funktionen

- ▶ Unsichere „klassische“ Verfahren: MD5, SHA1, RIPEMD
- ▶ Kryptographische Verfahren: Empfehlungen und Schlüssellängen (BSI)

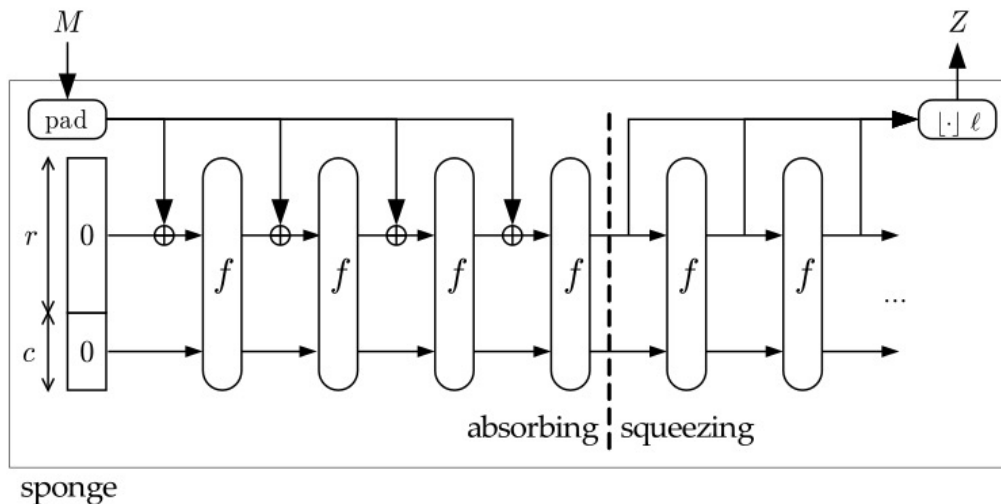
Verfahren	Empfohlene Algorithmen
Kryptographisch starke Hashverfahren	SHA-256, SHA-512/256, SHA-384 und SHA-512, SHA3-256, SHA3-384, SHA3-512
MAC-Verfahren	HMAC \geq 128, CMAC \geq 128, GMAC \geq 128
Signaturverfahren	RSA 2000, DSA 2000 , ECDSA 250, Merkle-Signaturen

https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR02102/BSI-TR-02102.pdf?__blob=publicationFile&v=7



Neuer Hash Standard seit 2012: SHA-3 (Keccak)

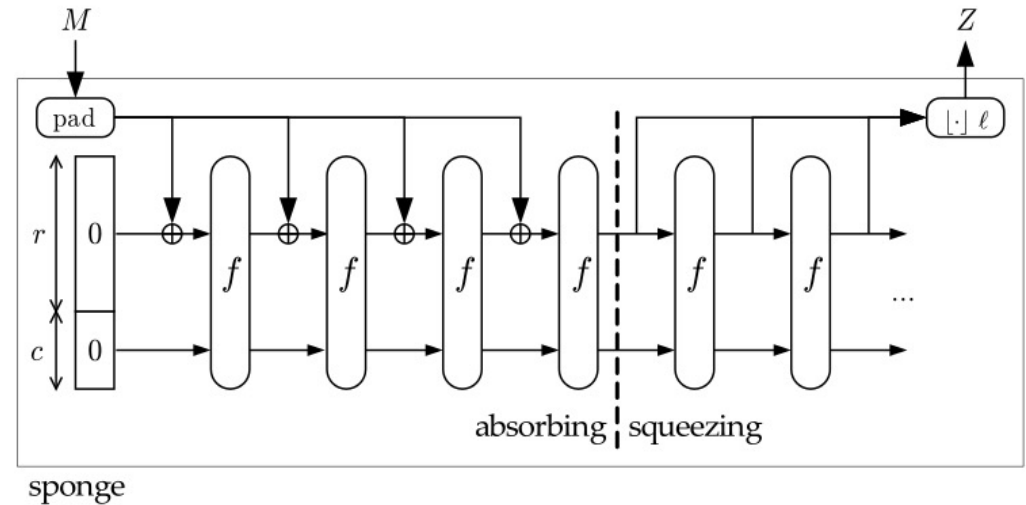
- ▶ Sieger aus einem Wettbewerb durch die NIST (National Institute of Standards and Technology)
<http://csrc.nist.gov/groups/ST/hash/sha-3/>
<https://keccak.team/index.html>
- ▶ Ausgabegröße variabel (224, 256, 384, 512 Bit)
- ▶ Sicherheit durch Kapazität beeinflussbar mittels **Sponge Function**
(mehr Sicherheit -> weniger Performance)



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



Details zu SHA-3 (Keccak)



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

- ▶ Absorbing:
 - ▶ State wird in zwei Teile geteilt: Capacity (c Bits) und Bit-Rate (r Bits)
 - ▶ Eingabeböcke M werden padded (aufgefüllt)
 - ▶ r -Bit Anteil wird mit XOR mit State verknüpft, c -Bit Anteil bleibt unverändert
 - ▶ Dann wird mit f (Keccak-Permutation) State vermischt
- ▶ Squeezing
 - ▶ Hash-Wert wird aus r -Bit Anteil des States extrahiert
 - ▶ Falls Länge nicht reicht wird State mehrfach permutiert und nach jeder Permutation weitere Ausgaben extrahiert
- ▶ Visualisierung s. CrypTool 2



Datenauthentisierung

- ▶ Unter **Datenauthentisierung** versteht man Kryptographische Verfahren, die garantieren, dass übersandte oder gespeicherte Daten nicht durch Unbefugte verändert wurden.
 - ▶ Sie Basieren auf kryptographische Schlüssel die zur Berechnung von Prüfsummen verwendet werden
 - ▶ Es gibt symmetrische und asymmetrische Verfahren
- ▶ Mit Datenauthentisierung sind zwei Sicherheitsziele erreichbar
 - ▶ Sicherung der **Integrität** der Daten
 - ▶ Sicherung der **Nichtabstreitbarkeit** einer Nachricht
→ nur bei digitalen Signaturen möglich

https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR02102/BSI-TR-02102.pdf?__blob=publicationFile&v=12



Message Authentication Code (MAC)

- ▶ Eine Hashfunktion die zusätzlich geheimen Schlüssel verwendet
- ▶ Der Schlüssel ist nur den beiden Kommunikationspartnern bekannt
- ▶ Das ermöglicht die Aufdeckung von unautorisierten Modifikationen (Integrität)
- ▶ Das ermöglicht die Überprüfung ob Daten einen authentischen Ursprung besitzen (Authentizität)




Erster Versuch für einen MAC

Konkatenieren

- ▶ Einfache Konkatenation $\boxed{\text{hash}(\text{Secret} \parallel \text{Daten})} = \text{MAC}$
KLEIN
- ▶ Dummerweise:
 - ▶ Wenn der Angreifer $\text{length}(\text{secret} \parallel \text{daten})$ kennt
 - ▶ Dann kann er $\text{hash}(\text{secret} \parallel \text{daten} \parallel \text{mehr-daten})$ berechnen **OHNE** secret zu kennen!
- ▶ Das nennt man: **Length extension attack**
https://en.wikipedia.org/wiki/Length_extension_attack
- ▶ Anfällig dafür sind z.B. SHA1, SHA2, nicht anfällig ist SHA3
- ▶ Problem: man muss einen Hash herstellen, den man nur mit einem Secret errechnen kann
 - ▶ Eine Lösung: HMAC

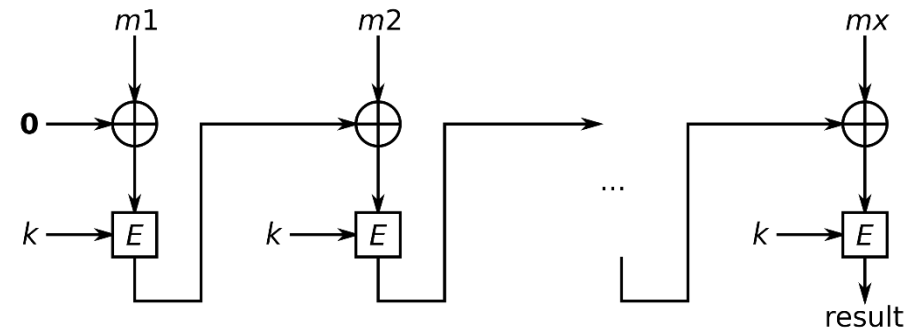


HMAC (Hashed MAC)

- ▶ HMAC ist ein beliebter MAC
- ▶
$$HMAC(K, m) = H((K \oplus opad) \parallel H((K \oplus ipad) \parallel m))$$
 - ▶ K ist secret key
 - ▶ Blockgröße ist 64 Bytes: entweder K mit Nullen auffüllen oder mit einer Hashfunktion auf 64 Byte reduzieren.
 - ▶ opad, ipad sind Konstanten in Blockgröße
- ▶ Ein Video zur Veranschaulichung von HMAC
 - ▶  <https://www.youtube.com/watch?v=BjlnMA-b8ZE>

▶ CBC-MAC (Cipher Block Chaining MAC)

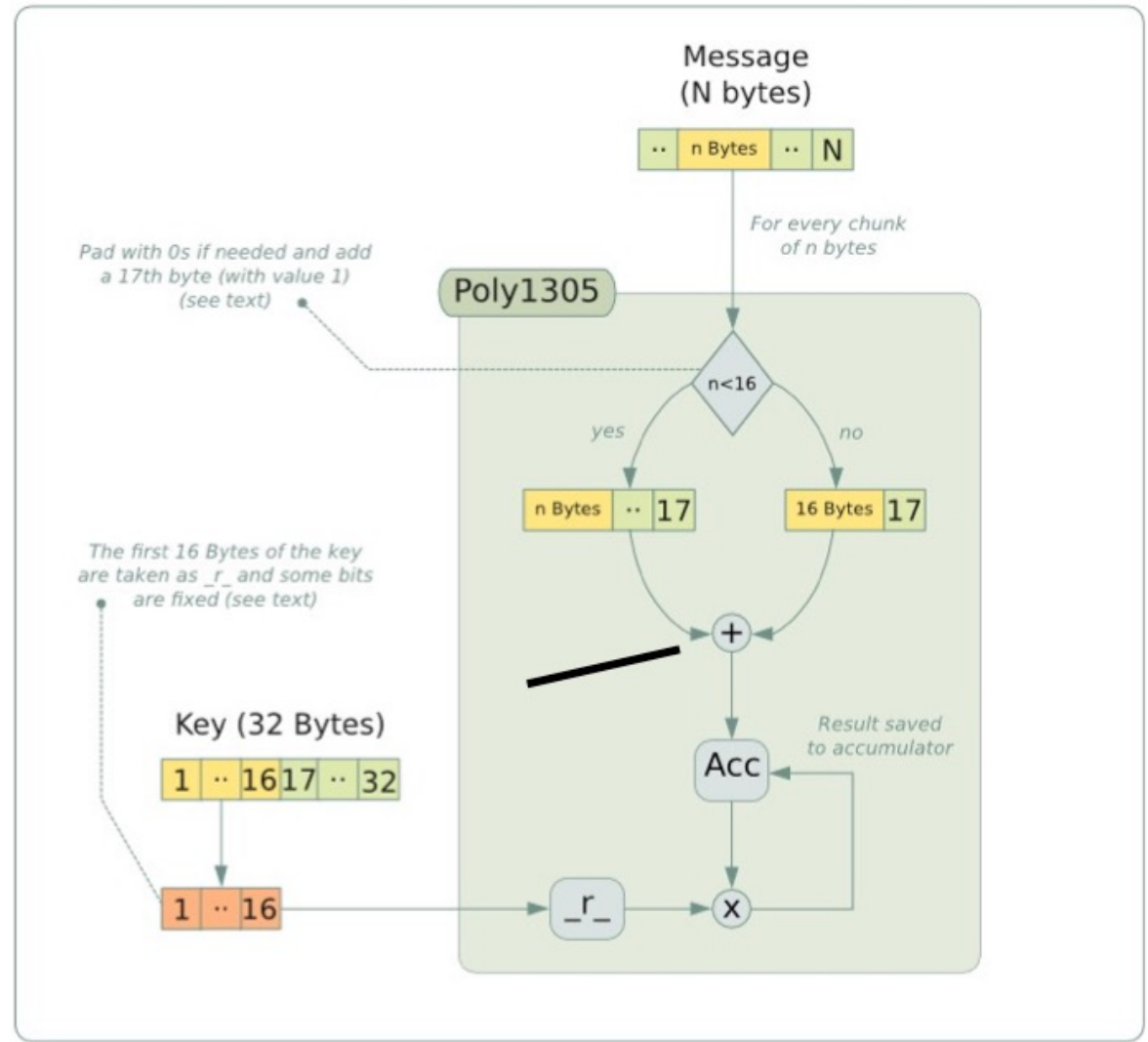
- ▶ CBC-MAC ist eine Technik um mit einem Block-Cipher Verfahren einen MAC zu berechnen.
- ▶ Er verwendet Blockchiffre im CBC-Modus
- ▶ Der IV wird auf 0 gesetzt, der MAC ist der letzte verschlüsselte Block
- ▶ $c_1 = (m_1 \oplus 0) \oplus K$ $c_x = (m_x \oplus c_{x-1}) \oplus K$
- ▶ CBC-MAC ist nur für Nachrichten fester/bekannter Länge sicher
sonst ist eine Length extension attack möglich



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



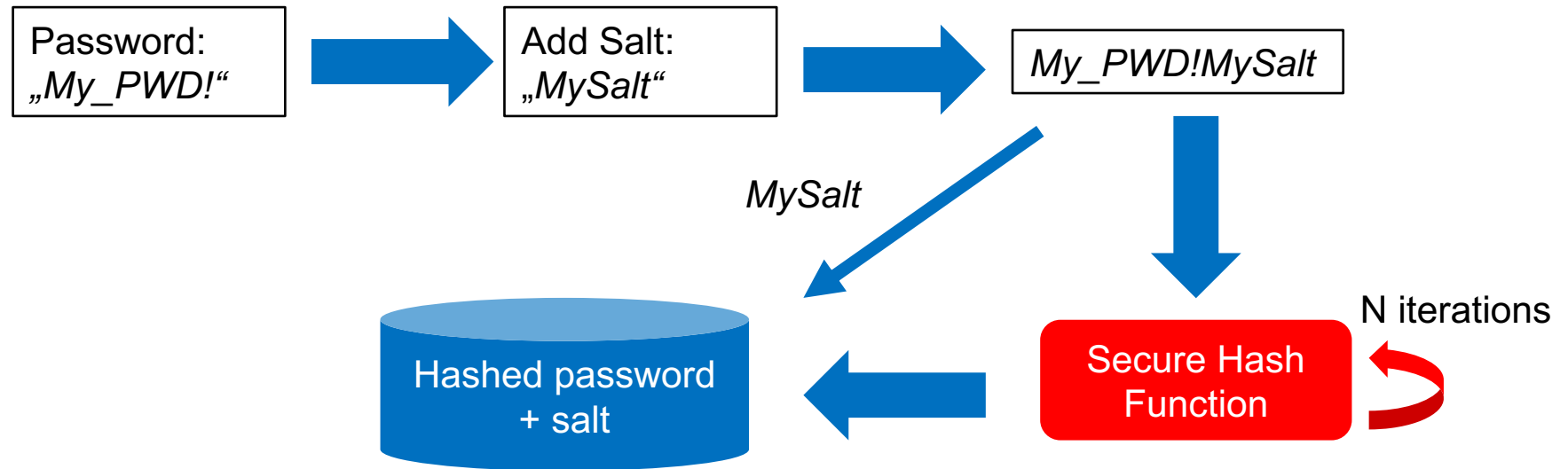
Quelle: <https://www.adalabs.com/adalabs-chacha20poly1305.pdf>

Hash Functions for Password Hashing

- ▶ Problems
 - ▶ Normal Hash functions are very fast and support brute force attacks
 - ▶ Same password results in same hash value
- ▶ Criterias 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)
 - ▶ Combination with **Salt**
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





Anwendungen und Grenzen von MAC

- ▶ Moderne Chiffrierverfahren kombinieren die Verschlüsselung mit einer MAC Berechnung
 - ▶ Counter Mode **CTR** mit **CBC-MAC**
 - ▶ **GCM** (s. Kapitel 2: GCM ist ein Chiffre mit MAC Berechnung)
 - ▶ **ChaCha20** mit **Poly1305**
- ▶ Anwendung von MAC
 - ▶ Angriffserkennung für Dateisysteme (Filesystem Intrusion Detection)
 - ▶ Absicherung von Softwarepakete (Patch, Update)
 - ▶ Sicherung von Kommunikationsprotokollen (z.B. TLS)
- ▶ Ein MAC sichert die Integrität und die Authentizität einer Nachricht
 - ▶ Aber es fehlt die Beweiskraft für Nichtabstreitbarkeit
 - ▶ Er kann nicht durch eine dritte Instanz verifiziert werden
 - ▶ Er basiert auf symmetrischer Kryptografie
 - ▶ Es fehlen Zertifikate



Digitale Signaturen sichern die Integrität von Daten und beweisen die Authentizität

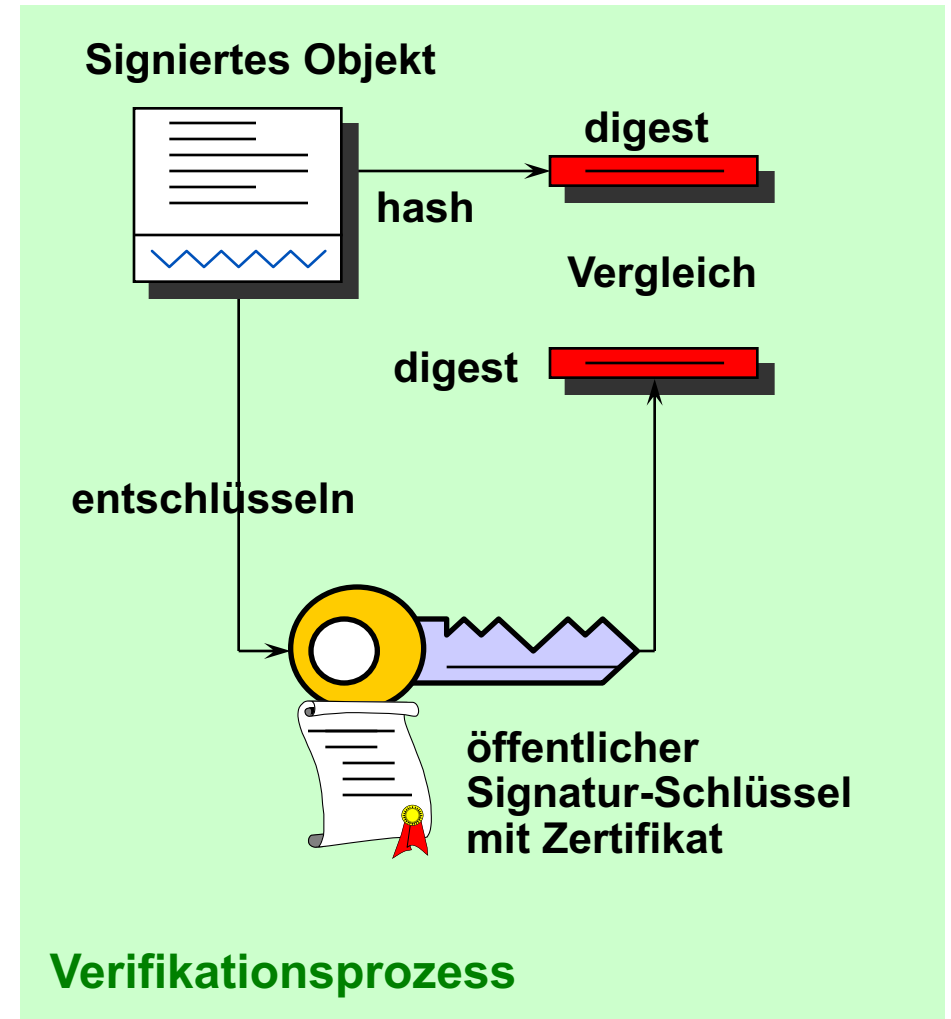
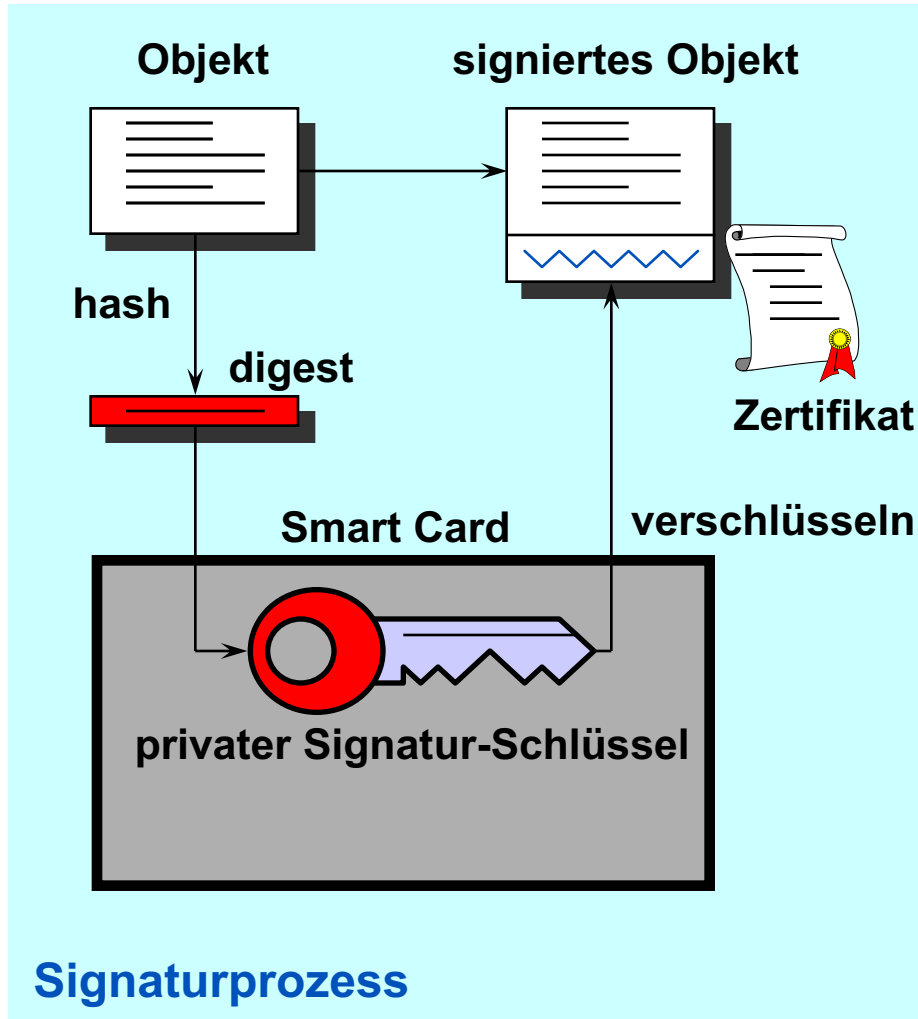
- ▶ Eine Digitale Signatur enthält
 - ▶ Zeit von einem Zeitstempeldienst
 - ▶ Person oder Ort (Server Name)
 - ▶ Signierten Digest des Dokuments

- ▶ Eine Digitale Signatur beweist
 - ▶ Integrität des Dokuments (was)
 - ▶ wer und wann signiert hat





Das Verfahren der Digitalen Signatur



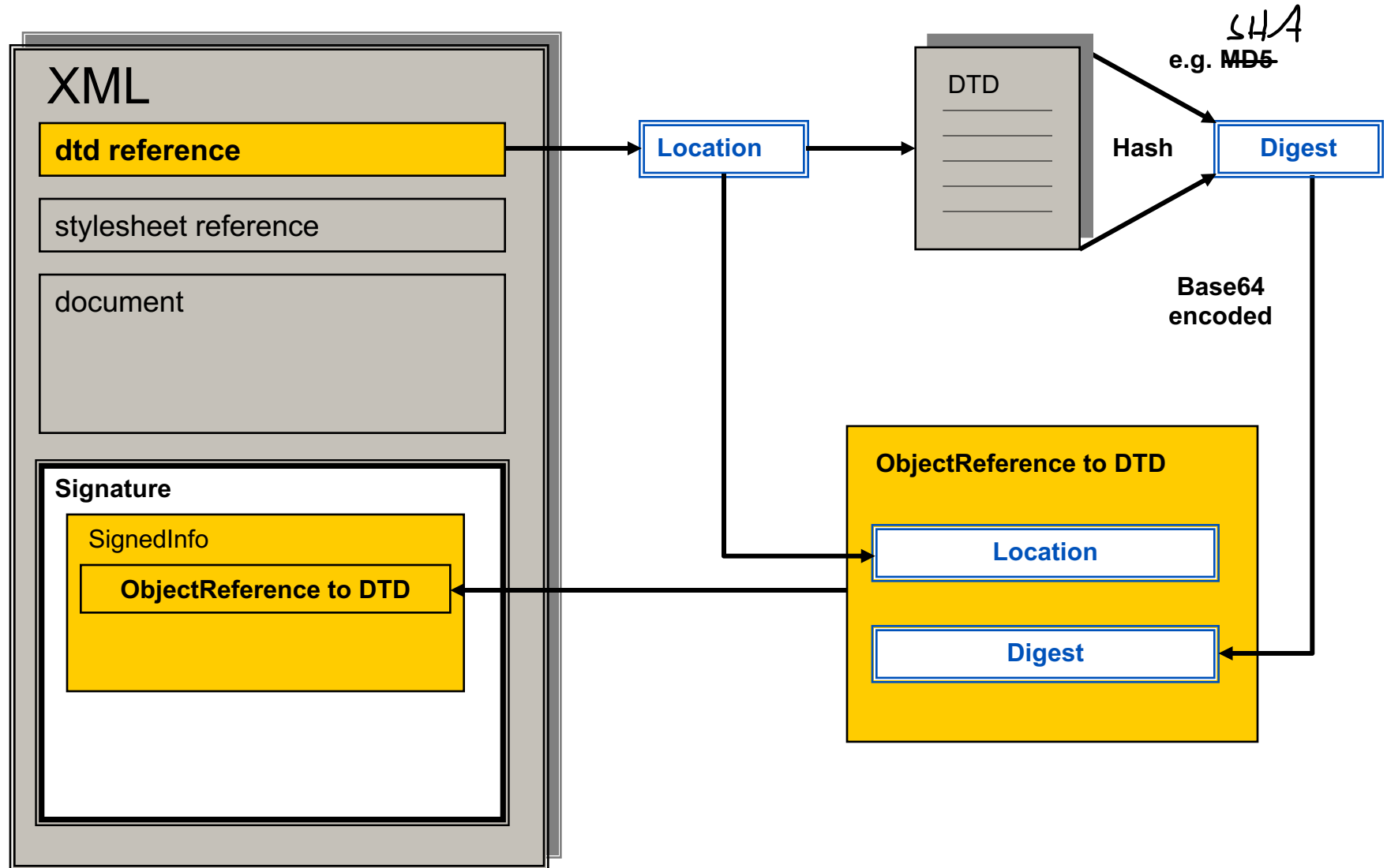


Standard für XML Signaturen

- ▶ Definiert Regeln und Syntax für die Signatur
 - ▶ Ganzer XML Dokumente
 - ▶ Von Teilen von XML Dokumenten
 - ▶ Beliebiger anderer Dateien
 - ▶ Literatur: <http://www.w3.org/Signature/>
- ▶ Drei Möglichkeiten zur Integration von XML Signaturen
 - ▶ **Detached Signature:** Die Signatur ist vom Dokument losgelöst und nicht in das Dokument eingebettet (getrennt)
 - ▶ **Enveloped Signature:** Die Signatur ist in das Dokument eingebettet (ummantelt)
 - ▶ **Enveloping Signature:** Die Signatur hat die Aufgabe eines Umschlags. Sie umschließt das ganze XML Dokument (ummantelnd)

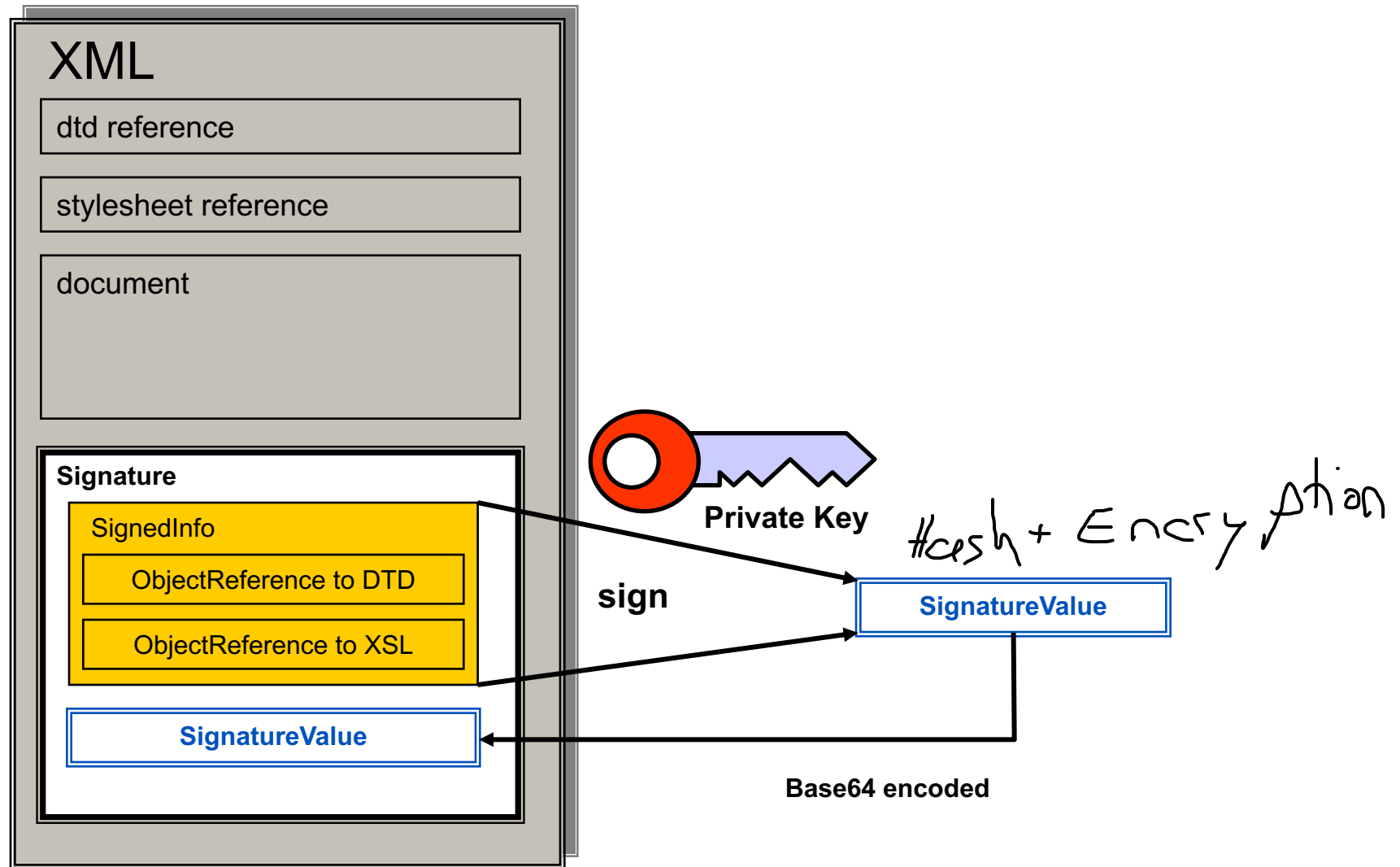


XML-Signature: Digest Berechnung, Referenzerstellung





XML-Signature: Signatur der Objektreferenzen





Bestandteile einer XML Signatur

```
...  
<Signature>  
  <SignedInfo>  
    <CanonicalizationMethod Algorithm="c14n"/>  
    <SignatureMethod Algorithm="rsa-sha1"/>  
    <Reference URI="http://foo.org/picture.jpg.zip">  
      <Transforms>  
        <Transform Algorithm="UnZip">  
        </Transform>  
      </Transforms>  
      <DigestMethod Algorithm="sha1"/>  
      <DigestValue>345x3mUrks563X</DigestValue>  
    </Reference>  
  </SignedInfo>  
  <SignatureValue>MC0affe34lkV</SignatureValue>  
  <KeyInfo>  
    <X509Data>  
      <X509SubjectName>DN=John Doe</X509SubjectName>  
    </X509Data>  
  </KeyInfo>  
</Signature>  
...
```

Verarbeitungsinformationen:
verwendete Algorithmen
Kanonisierung

Verarbeitungsinformationen:
Datenquellen für Signatur
Transformationen

Signaturwert

Key-Management:
PKI-Anbindung



Kanonisierung

- ▶ XML Dokumente **semantisch** gleichen Inhalts können auf verschiedene Weise repräsentiert werden:
`<myelement attr=„123“/>`
`<myelement attr=„123“></myelement>`
- ▶ Die Signatur erhält dadurch einen völlig anderen Wert
- ▶ Eine standardisierte Darstellung ist notwendig: Kanonisierung
- ▶ Literatur: <https://www.w3.org/TR/xml-c14n/>

