

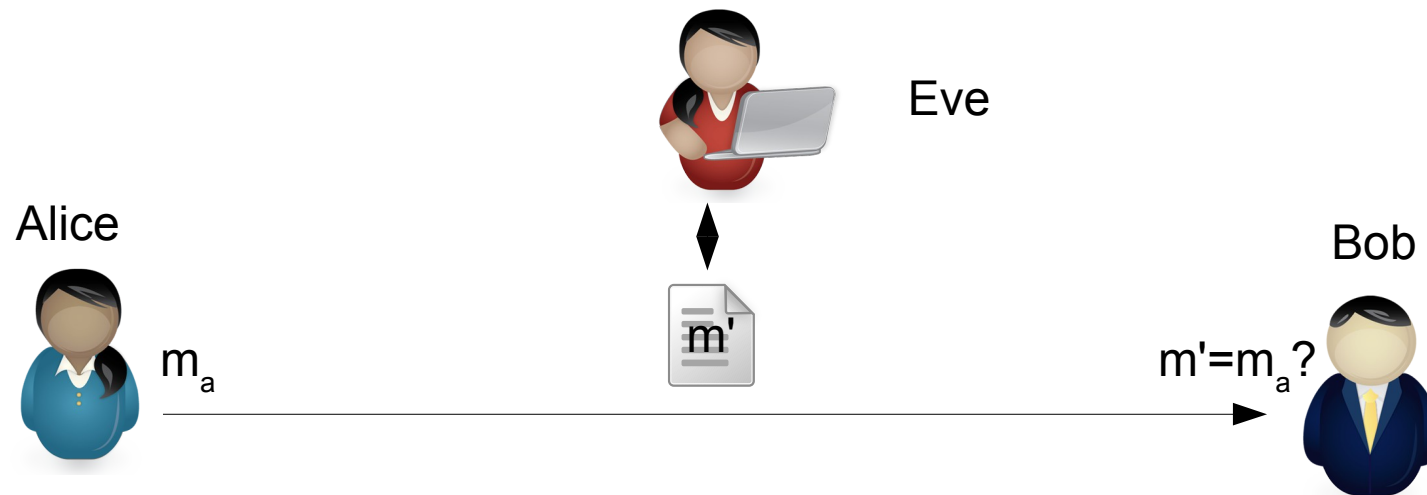
Themen zur Computersicherheit

Authentische Nachrichten

PD Dr. Reinhard Bündgen
buendgen@de.ibm.com

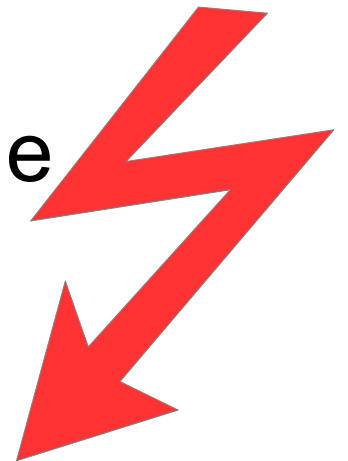
Nachrichten Authentizität

- Bob empfängt eine Nachricht von Alice
- Probleme:
 - Kommt die Nachricht wirklich von Alice?
 - Wenn ja, ist sie unverändert?



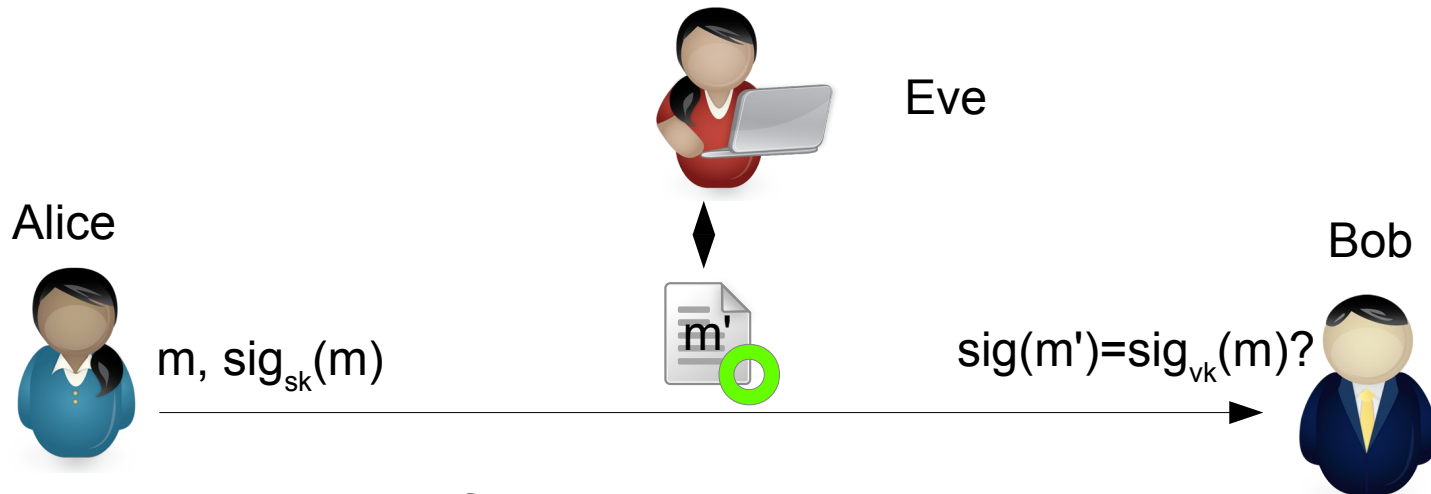
Idee: Nutze kryptographische Hashes

- sei H kryptografische Hashfunktion
- Alice verschickt m und $H(m)$
- Bob empfängt m' und h' und kann feststellen, ob $h' = H(m')$
- Aber, Eve kann sowohl m durch m' ersetzen als auch $H(m)$ durch $H(m')$
- Ferner gibt es keinen Hinweis, dass die Nachricht überhaupt von Alice kommt.



Verfahren zur Nachrichtenauthentifizierung

- Nachrichten werden „unterschrieben“



- Symmetrische Verfahren:
 - Sender & Empfänger nutzen gemeinsames Geheimnis (Schlüssel)
 - Verfahren basierend auf Blockchiffren
 - Verfahren basierend auf kryptografischen Hashes
- Asymmetrische Verfahren
 - Sender unterschreibt mit geheimen Schlüssel
 - Empfänger prüft Unterschrift mit öffentlichem Schlüssel

Symmetrische Nachrichtenauthentifizierung

- message authentication code (MAC)
- $\text{mac}: \Sigma^n \times \Sigma^* \rightarrow \Sigma^t$ berechnet zu einem Schlüssel k der Länge n und einer Nachricht m den MAC (Tag) $\text{mac}_k(m)$ der Länge t
- muss stark kollisionsresistente Einwegfunktion sein
- Sender
 - verschickt $m, \text{mac}_k(m)$
- Empfänger
 - empfängt m', s und
 - überprüft, ob $s = \text{mac}_k(m')$, dann folgt $m' = m$

MACs die auf Blockchiffren basieren

- CBC-MAC
- XCBC-MAC
- CMAC

CBC-MAC & CMAC

- CBC-MAC:
 - definiert in FIPS 113
 - MSBs des letzten Geheimtextblock einer CBC-Verschlüsselung mit
 - mit $IV = 0^b$
 - bei Bedarf mit Padding durch Nullen
 - Probleme
 - $t < b$ (i.A. < 128 bit)
 - Falls $t = b$ dann gilt (z.B wenn fehlende Bits geraten werden):
 - $CBC\text{-}mac_k(m_1) = CBC\text{-}mac_k(m_2) \Rightarrow CBC\text{-}mac_k(m_1 || m_3) = CBC\text{-}mac_k(m_2 || m_3)$
 - $|m_1| = |m_2| = b \Rightarrow CBC\text{-}mac_k(m_1 || m_2) = CBC\text{-}mac_k(m_2 || CBC\text{-}mac_k(m_1) \uplus CBC\text{-}mac_k(m_2) \uplus m_2)$
- CMAC
 - definiert in NIST SP 800-38B
 - CBC-MAC für alle Blöcke bis auf letzten Block
 - letzter Block: wenn nötig Padding mit 10^*
 - Maskierung mit schlüsselabhängiger Maske

Hashbasierte MACs

- HMAC

- sei h eine Hashfunktion mit Blocklänge b
- k der Schlüssel dann ist
- $k' = k$ falls $|k| \leq b$ und $k' = h(k)$ sonst
- $\text{opad} = (0x5C)^{b/8}$, $\text{ipad} = (0x36)^{b/8}$
- $h\text{-hmac}_k(m) = h(k' \uplus \text{opad} \parallel h((k' \uplus \text{ipad}) \parallel m))$
- Bemerkung: PBKDF2 nutzt HMAC in seiner PRF

Kombinierte Authentisierungs- und Verschlüsselungsverfahren

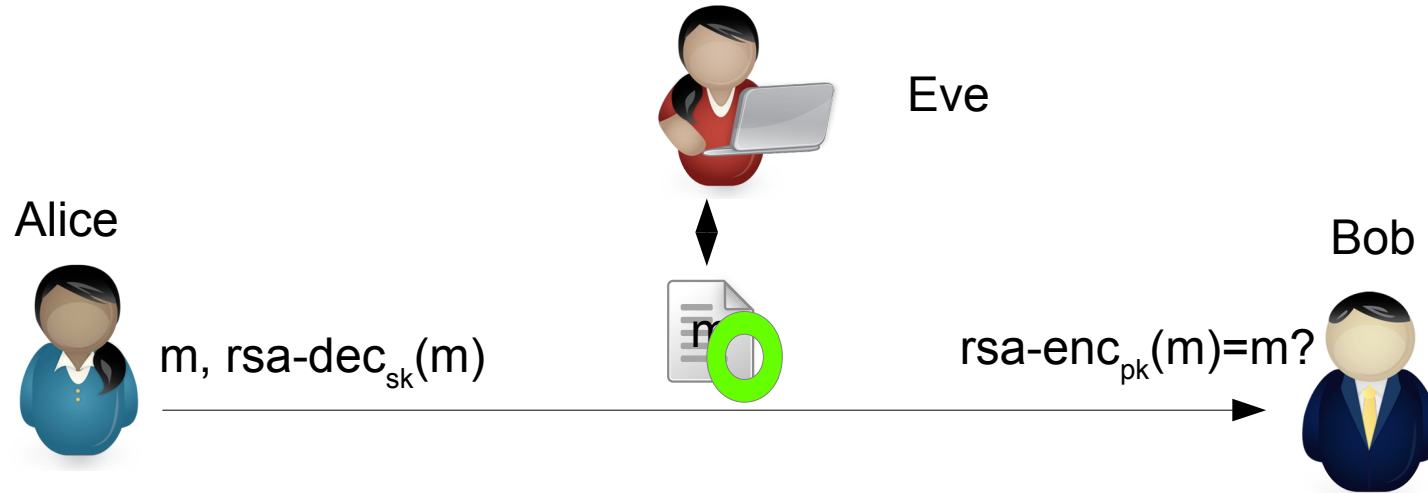
- Authenticated Encryption with Associated Data (AEAD)
- Galois/Counter Mode (GCM)
 - NIST SP 800-38D
 - basiert auf zugelassener Blockchiffre BC mit 128 Bit Blocklänge und ≤ 128 Bit Schlüssellänge (z.B. BC = AES128)
 - Eingaben (Verschlüsselung)
 - Initialisierungswert iv, Schlüssel k, zusätzliche authentifizierte Daten a, Klartext m, Taglänge t
 - Ausgabe (Verschlüsselung):
 - $BC\text{-}GCTR_k(m)$
 - $MSB(t, BC\text{-}GCTR_k(GHASH(\text{format}(a, BC\text{-}GCTR_k(m)))))$
- GMAC: GCM mit leerem Klartext
- Counter with CBC-MAC (CCM)
 - NIST SP 800-38C
 - basiert auf zugelassener Blockchiffre BC mit 128 Bit Blocklänge
 - Eingaben (Verschlüsselung):
 - Schlüssel k, Nonce n, assoziierte Daten a, Klartext m, Taglänge t
 - Ausgabe (Verschlüsselung):
 - $BC\text{-}CTR_k(m)$
 - $MSB(t, BC\text{-}CBC\text{-}MAC_k(\text{format}(n, a, m) \parallel BC_k(ctr_0))$

Asymmetrische Signaturen

- Unterschreiben mit privatem Schlüssel
- Verifizieren mit öffentlichem Schlüssel

- RSA basierte Verfahren
- Digital Signature Algorithm (DSA)

RSA basierte Unterschrift

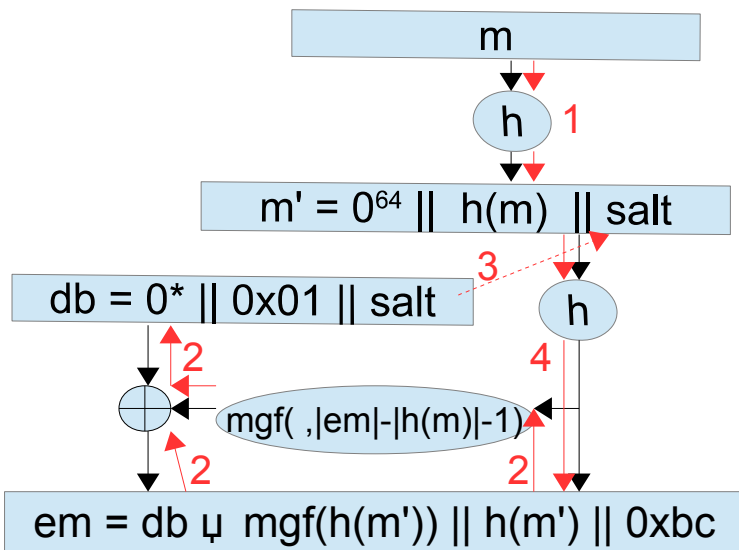


- RSA Entschlüsselung dient als Signatur
 - $\text{RSA-sig}_{sk} = \text{RSA-dec}_{sk}$
- RSA Verschlüsselung dient als Verifikationsverfahren
 - $\text{RSA-ver}_{pk} = \text{RSA-enc}_{pk}$
- Wenn $m < \text{RSA Modulus}$, dann kann die Nachricht aus der Signatur gewonnen werden
- bei größeren Nachrichten wird ein Hash der Nachricht signiert und die Nachricht muss zusammen mit der Signatur versendet werden
 - Verifikation: $\text{RSA-ver}_{pk}(\text{RSA-sig}_{sk}(h(m))) = h(m)?$
- Achtung!
 - Eine Signatur, darf nicht mit gleichem Schlüssel verschlüsselt werden
 - Ein RSA Geheimtext darf nicht mit gleichem Schlüssel signiert werden

RSA Signatur Schemata

Signature with Appendix Probabilistic Signature Scheme RSASSA-PSS

- PSS Kodierung
 - salt: Zufallsstring
 - h: Hashfunktion
 - $|em| = |n|-1$



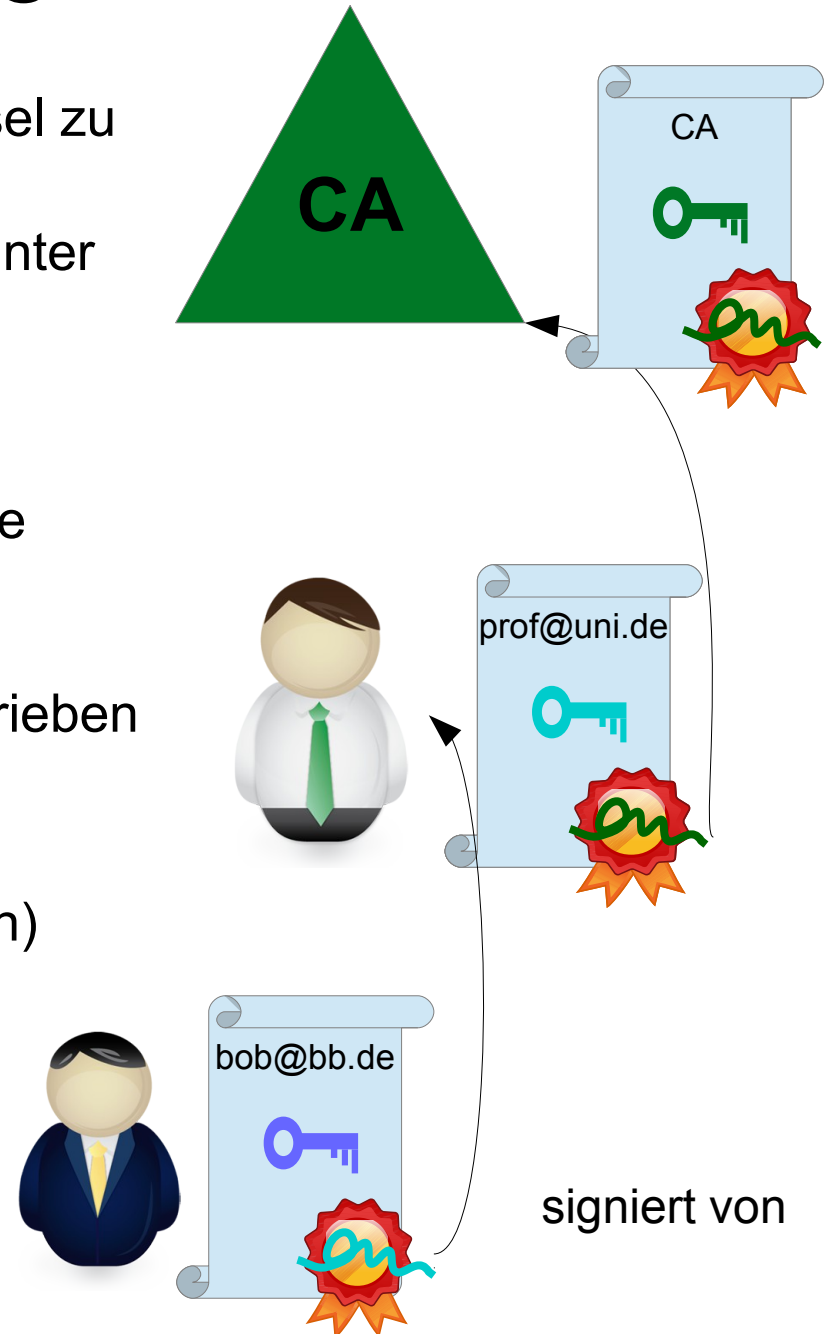
Signature with Appendix RSASSA-PKCS1-v1_5

- $t = \text{DER}(\text{DigestInfo}(h, h(m)))$
- $em = 0x00 || 0x01 || 0xff^{|\text{em}|-|t|-3} || 0x00 || t$
- DER ist eine Distinguished Encoding Rules Kodierung der Struktur DigestInfo mit ASN.1 Syntax

```
DigestInfo ::= SEQUENCE {
    digestAlgorithm AlgorithmIdentifier,
    digest OCTET STRING
}
```

Zertifikate

- Wie stellt Alice fest, dass öffentlicher Schlüssel zu Bob gehört?
 - Eve könnte ihren öffentlichen Schlüssel unter Bobs Namen in öffentlichem Verzeichnis ablegen
- Zertifikat
 - Datenstruktur, die Attribute eines Subjekte (z.B. IP-Adresse) zusammen mit seinem öffentlichen Schlüssel enthält und
 - von vertrauenswürdiger Instanz unterschrieben ist
- Zertifizierung
 - rekursives Problem (Kette von Zertifikaten)
 - am Ende muss eine wohlbekannte Zertifizierungsautorität (CA) stehen
 - selbstsignierte Zertifikate
 - für CAs
 - zum Testen



Beispiel: X.509v3 Zertifikat

- Daten eines X.509v3 Zertifikats: Certificate:
 - Version Data: Version: 3 (0x2)
 - Seriennummer Serial Number: 1 (0x1)
 - Algorithmen-ID Signature Algorithm: md5WithRSAEncryption
 - Aussteller Issuer: C=AT, ST=Steiermark, L=Graz, O=TrustMe Ltd, OU=Certificate Authority, CN=CA/Email=ca@trustme.dom
 - (Land/Bundesland/Ort/Organisation ...) Validity
 - Gültigkeit (von/bis) Not Before: Oct 29 17:39:10 2000 GMT
 - Zertifikatinhaber Not After : Oct 29 17:39:10 2001 GMT
 - Zertifikatinhaber-Schlüsselinformationen Subject: C=AT, ST=Vienna, L=Vienna, O=Home, OU=Web Lab, CN=anywhere.com/Email=xyz@anywhere.com
 - (asymmetrisches Verfahren/öffentlicher Schlüssel) Subject Public Key Info:
 - optional: eindeutige ID des Ausstellers Public Key Algorithm: rsaEncryption
 - optional: eindeutige ID des Inhabers RSA Public Key: (1024 bit)
 - Erweiterungen Modulus (1024 bit):
 - 00:c4:40:4c:6e:14:1b:61:36:84:24:b2:61:c0:b5:
d7:e4:7a:a5:4b:94:ef:d9:5e:43:7f:c1:64:80:fd:
9f:50:41:6b:70:73:80:48:90:f3:58:bf:f0:4c:b9:
90:32:81:59:18:16:3f:19:f4:5f:11:68:36:85:f6:
1c:a9:af:fa:a9:a8:7b:44:85:79:b5:f1:20:d3:25:
7d:1c:de:68:15:0c:b6:bc:59:46:0a:d8:99:4e:07:
50:0a:5d:83:61:d4:db:c9:7d:c3:2e:eb:0a:8f:62:
8f:7e:00:e1:37:67:3f:36:d5:04:38:44:44:77:e9:
f0:b4:95:f5:f9:34:9f:f8:43
 - Exponent: 65537 (0x10001)
 - X509v3 extensions:
 - X509v3 Subject Alternative Name: email:xyz@anywhere.com
 - Netscape Comment: mod_ssl generated test server certificate
 - Netscape Cert Type: SSL Server
 - Signature Algorithm: md5WithRSAEncryption
 - 12:ed:f7:b3:5e:a0:93:3f:a0:1d:60:cb:47:19:7d:15:59:9b:
3b:2c:a8:a3:6a:03:43:d0:85:d3:86:86:2f:e3:aa:79:39:e7:
82:20:ed:f4:11:85:a3:41:5e:5c:8d:36:a2:71:b6:6a:08:f9:
cc:1e:da:c4:78:05:75:8f:9b:10:f0:15:f0:9e:67:a0:4e:a1:
4d:3f:16:4c:9b:19:56:6a:f2:af:89:54:52:4a:06:34:42:0d:
d5:40:25:6b:b0:c0:a2:03:18:cd:d1:07:20:b6:e5:c5:1e:21:
44:e7:c5:09:d2:d5:94:9d:6c:13:07:2f:3b:7c:4c:64:90:bf:
ff:8e

Sperren von Zertifikation

- Was passiert wenn die Unterschrift eines Zertifikats nicht mehr sicher ist?
 - Aussteller ist nicht länger vertrauenswürdig
 - privater Schlüssel des Ausstellers geknackt
- Zertifikatssperrlisten (certificate revocation lists, CRL)
 - Überprüfung, ob Zertifikat in CRL enthalten
 - regelmäßiges aktualisieren der CRLs