IT-Sicherheit:

Digitale Signatur, Zertifikate, PKI, E-Mail-Sicherheit

Digitale Signaturen

- Elektronisches Pendant zur eigenhändigen Unterschrift
- Electronic Signatures in Global and National Commerce Act („E-SIGN“, US, 2000), §106(5):

  The term “electronic signature” means an electronic sound, symbol, or process, attached to or logically associated with a contract or other record and executed or adopted by a person with the intent to sign the record.

Anforderungen an digitale Signaturen

- Digitale Signatur sollte nach Bruce Schneier folgende Eigenschaften besitzen:
  1. Überprüfbar (***verifiable***): Der Empfänger kann einfach überprüfen, dass die Nachricht von dem Unterzeichner unterschrieben wurde
  2. Nicht fälschbar (***unforgeable***): Nur der Unterzeichner kann die Signatur an das Dokument anhängen
  3. Nicht wiederverwendbar (***nonreusable***): Man kann nicht einfach eine Unterschrift von einem Dokument entfernen und an ein anderes anhängen
  4. Unveränderbar (***unalterable***): Nach der Unterzeichnung ist das Dokument nicht mehr veränderbar (vgl. Datenintegrität)
  5. Nicht zurücknehmbar (***nondeniable***): Die Unterschrift kann nicht zurückgenommen werden. (vgl. Verbindlichkeit).

Digitale Signatur mit RSA

Digitale Unterschrift:

Signatur D: $D(M) = M^d \bmod n = S$

Verifikation E: $E(S) = S^e \bmod n = M^{de} \bmod n = M$

Inwieweit werden die fünf Anforderungen an eine digitale Signatur erfüllt? Zum Beispiel:

- Verifikation (s.o.)
- Unveränderbarkeit: unterschiedliche Werte von *M* liefern verschiedene Werte für $M^d \bmod n$

Digitale Signatur mit Elgamal (1)

- Keine Vertauschung von Ver- und Entschlüsselungsfunktion wie bei RSA
  - Nachricht M kann aus der Signatur nicht zurückgewonnen werden

- Schlüsselerzeugung von Alice (wie bei der Verschlüsselung):
  - Wählt zufällig eine Zahl **a** und berechnet $\boldsymbol{\alpha := g^a \bmod p}$.
  - Veröffentlicht (**α, g, p**) als öffentlichen Schlüssel.
  - **a** ist der private Schlüssel!

Digitale Signatur mit Elgamal (2)

- Digitale Signatur einer Nachricht M:
  - Alice wählt eine einmalige, geheime Zufallszahl (nonce) **k** (mit gcd(k, p–1) = 1) und bildet:

    $\mathbf{r = g^k \bmod p}$
  - Dann berechnet Alice **s,** so dass

    $\mathbf{(a \times r + k \times s) = H(M) \mod (p-1)}$

    (Erweiterten euklidischen Algorithmus anwenden und $\mathbf{k^{-1}}$ (Multiplikatives Inverses von **k**) berechnen.)
  - Digitale Signatur ist das Paar **(r, s)**.

Fermat: $g^{(p-1)} = 1$ für gcd(g,p) = 1

Digitale Signatur mit Elgamal (3)

- Verifikation der digitalen Signatur einer Nachricht M:
  - Empfänger (z.B. Bob) erhält **M** und Signatur **(r,s)**

Empfänger überprüft gegen öffentlichen Schlüssel **α**, ob:

$$g^{H(M)} \bmod p = (\alpha^r \times r^s) \bmod p$$

Denn es müsste gelten:

$$g^{H(M)} \bmod p = g^{a \times r + k \times s} \bmod p = g^{a^r} \times g^{k^s} \bmod p = (\alpha^r \times r^s) \bmod p$$

Satz von Fermat: $g^{(p-1)} = 1$ für $\gcd(g,p) = 1$

Prinzip der digitalen Signatur

- Problem bei obigem Signaturschemata mit RSA und Elgamal:
  - die Signatur ist zu lang

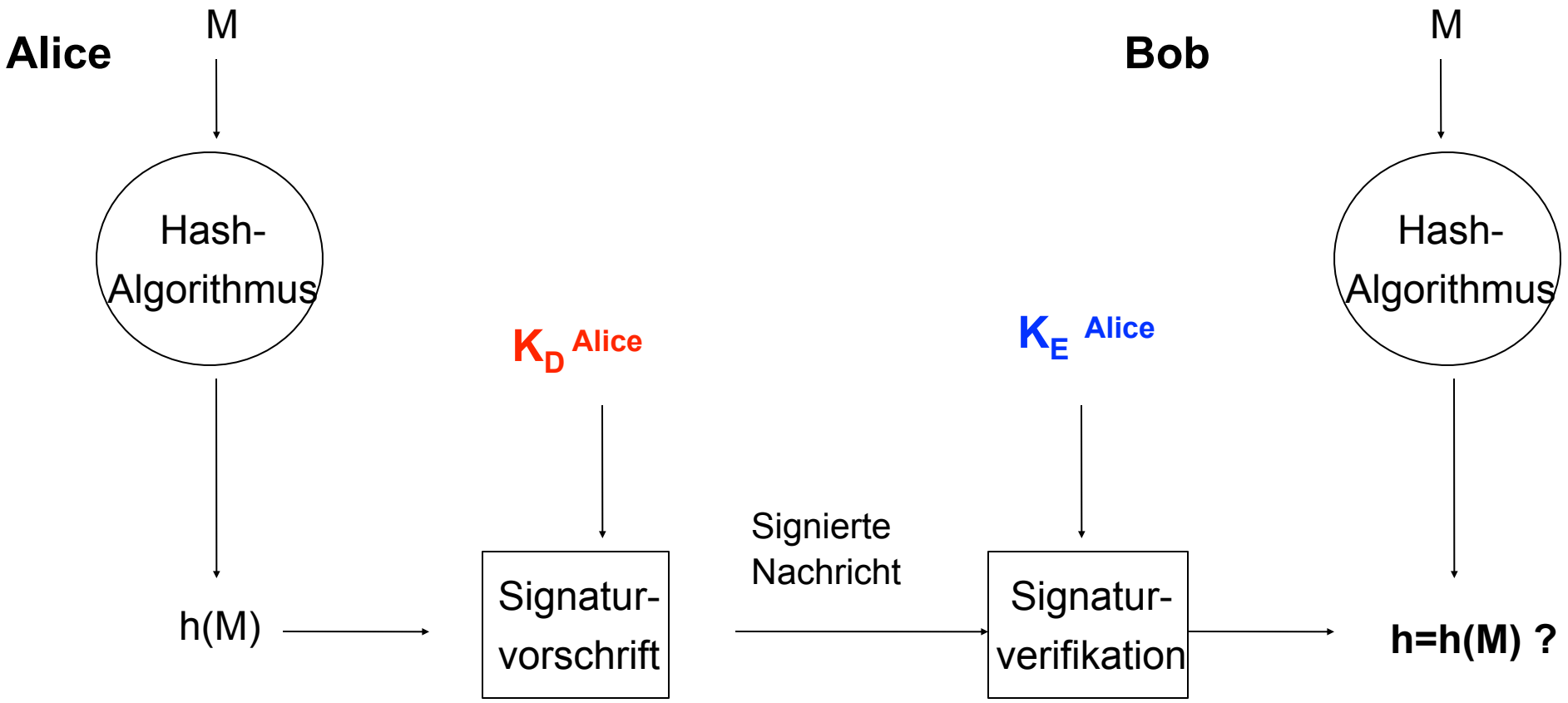
- Lösungsidee:
  - Alice bildet zunächst den Hashwert *H(M)* (z.B. mit SHA1; MD5) von *M* und signiert diesen mit ihrem privaten Schlüssel.
  - Alice sendet die Nachricht und die Signatur an Bob.
  - Bob bildet ebenfalls den Hashwert *H(M)* von *M* .
  - Bob entschlüsselt die „digitale Signatur“ mit dem öffentlichen Schlüssel von Alice und erhält den Wert **h**.
  - Gilt **h=H(M)**, so akzeptiert Bob die Signatur.

Wiederholung: Kryptographischer Hash

- Merkle-Damgård: Hash fester Größe aus variablen Strings
  - ~~MD5~~: 128 bit
  - ~~SHA-1~~: 160 bit $H_i = C(H_{i-1}, M_i)$, nach Padding (Block: 512 bit)
  - RIPEMD-160 (von der EU entwickelt)
- Heute: SHA-2 (Merkle-Damgård), SHA-3 (Keccak — sponge function)

- Ziele:
  - Kollisionsresistent: schwer, $x \neq y$ zu finden mit $h(x) = h(y)$
  - Urbildresistent: schwer, zu einem a ein y zu finden mit $a = h(y)$
  - (Urbild-2: schwer, zu einem x ein $y \neq x$ zu finden mit $h(x) = h(y)$)

Funktionsweise: Bildung und Verifikation der digitalen Signatur



Digital Signature Algorithm (DSA)

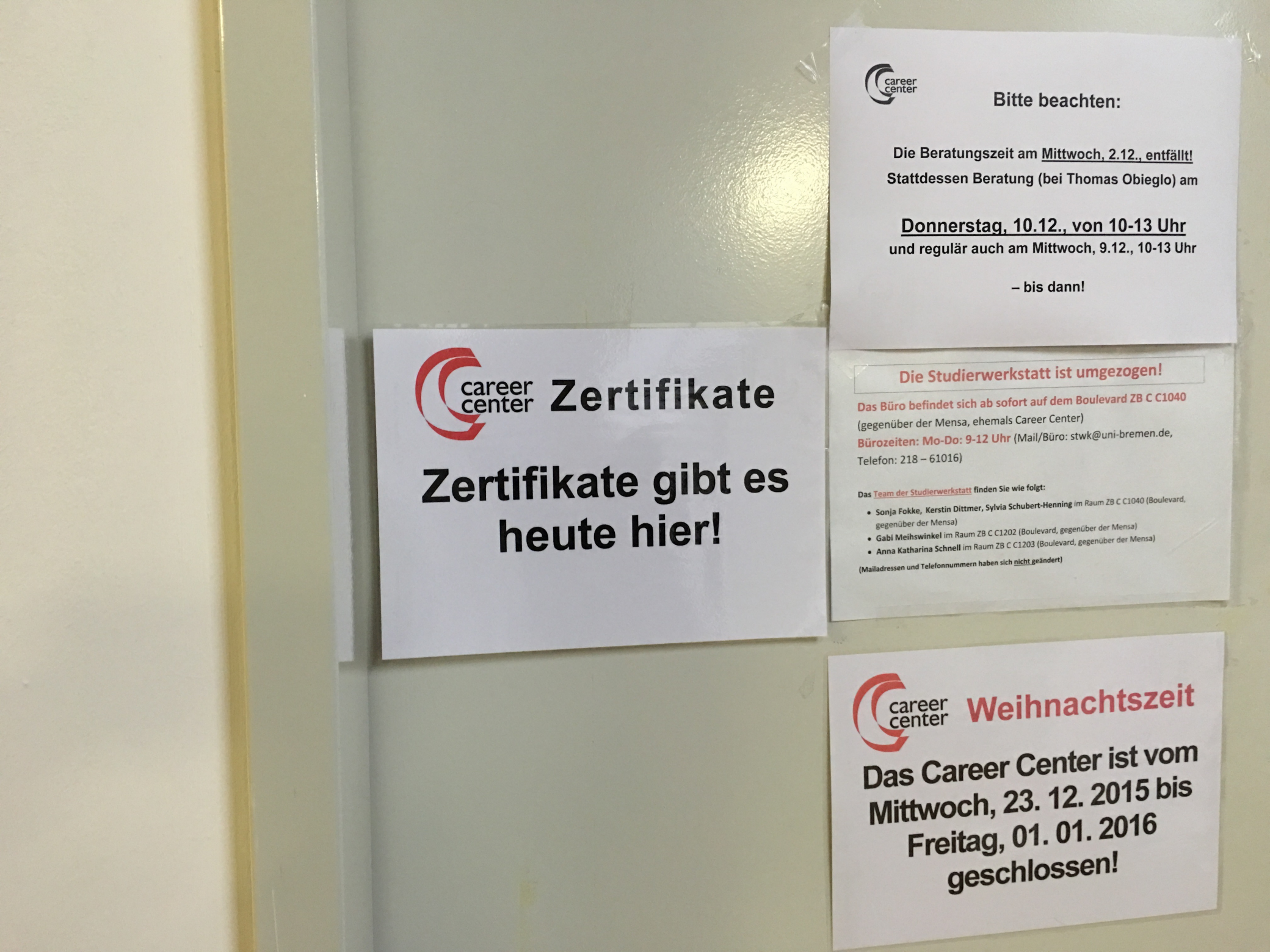
- United States Federal Government Standard für digitale Signaturen
- 1991 vom National Institute for Standards and Technology (NIST) als ein Algorithmus für den Einsatz im Digital Signature Standard (DSS) vorgeschlagen
- Basiert auf Algorithmus für Digitale Signaturen gemäß ElGamal
- Verwendung der Einweg-Hashfunktion SHA-1 (oder -2)
- Details bitte nachlesen (z.B. Wikipedia)

DSA: Status

- Erfordert guten RNG
- Abhilfe: RFC 6979 (Deterministic Usage)
  - „Zufallszahl“ aus dem Hash bestimmen
- OpenSSH: DSA war eine Weile auf 1024 bit beschränkt
  - Das ist zu wenig gegen nationale Angriffe
- DSA ist jetzt in OpenSSH ganz deprecated
  - Man will eh ECDSA (oder ed25519) verwenden
- DSA ist aber weiterhin sicher (mit genug Bits, und mit gutem RNG oder RFC 6979)

Zertifikate

- Problem: Woher weiß Bob, dass K_E^{Alice} zu Alice gehört?
  - Persönlicher Austausch des öffentlichen Schlüssels ist in der Regel nicht praktikabel.
  - Gefahr eines Man-in-the-Middle-Angriffs
- Wie kann man einen öffentlichen Schlüssel mit der Identität des Besitzers (z.B. Alice) verknüpfen?
- Lösung: Ausstellung von Zertifikaten durch eine vertrauenswürdige Partei bzw. Zertifizierungsstelle (***Certification Authority***, CA)


career center
Zertifikate
Zertifikate gibt es heute hier!
career center
Bitte beachten:
Die Beratungszeit am Mittwoch, 2.12., entfällt!
Stattdessen Beratung (bei Thomas Obieglo) am
Donnerstag, 10.12., von 10-13 Uhr
und regulär auch am Mittwoch, 9.12., 10-13 Uhr
– bis dann!
Die Studierwerkstatt ist umgezogen!
Das Büro befindet sich ab sofort auf dem Boulevard ZB C C1040
(gegenüber der Mensa, ehemals Career Center)
Bürozeiten: Mo-Do: 9-12 Uhr (Mail/Büro: stwk@uni-bremen.de,
Telefon: 218 – 61016)
Das Team der Studierwerkstatt finden Sie wie folgt:
Sonja Fokke, Kerstin Dittmer, Sylvia Schubert-Henning im Raum ZB C C1040 (Boulevard, gegenüber der Mensa)
Gabi Meihswinkel im Raum ZB C C1202 (Boulevard, gegenüber der Mensa)
Anna Katharina Schnell im Raum ZB C C1203 (Boulevard, gegenüber der Mensa)
(Mailadressen und Telefonnummern haben sich nicht geändert)
career center
Weihnachtszeit
Das Career Center ist vom Mittwoch, 23. 12. 2015 bis Freitag, 01. 01. 2016 geschlossen!

Zertifikate vs. Schlüsselpaare

- Server authentisiert sich, indem er beweist, im Besitz des **privaten Schlüssels** zu sein
- Schlüsselpaar: Privater Schlüssel + Öffentlicher Schlüssel
  - wird im allgemeinen direkt auf dem **Server** generiert
- Zertifikat: Öffentlicher Schlüssel, Digitale Signatur der CA
  - wird von der **CA** ausgestellt
  - die braucht dafür nur den Öffentlichen Schlüssel + Metadaten: **Certificate Signing Request (CSR)**
- Zertifikate sind **öffentlich** und werden jedermann gegeben, damit diese den Server authentisieren können
  - für letzteres benutzt Server dann den geheimen **privaten** Schlüssel

X.509 v3-Zertifikate

- Festgelegt in RFC 5280 und RFC 6818 (war 3280, 2459); Standard für digitale Zertifikate
- **SubjectName:** C=DE, O=Uni-Bremen, OU=FB3, OU=People, CN=Emil Mustermann

  IssuerName: C=US, O=Verisign, CN=Public CA

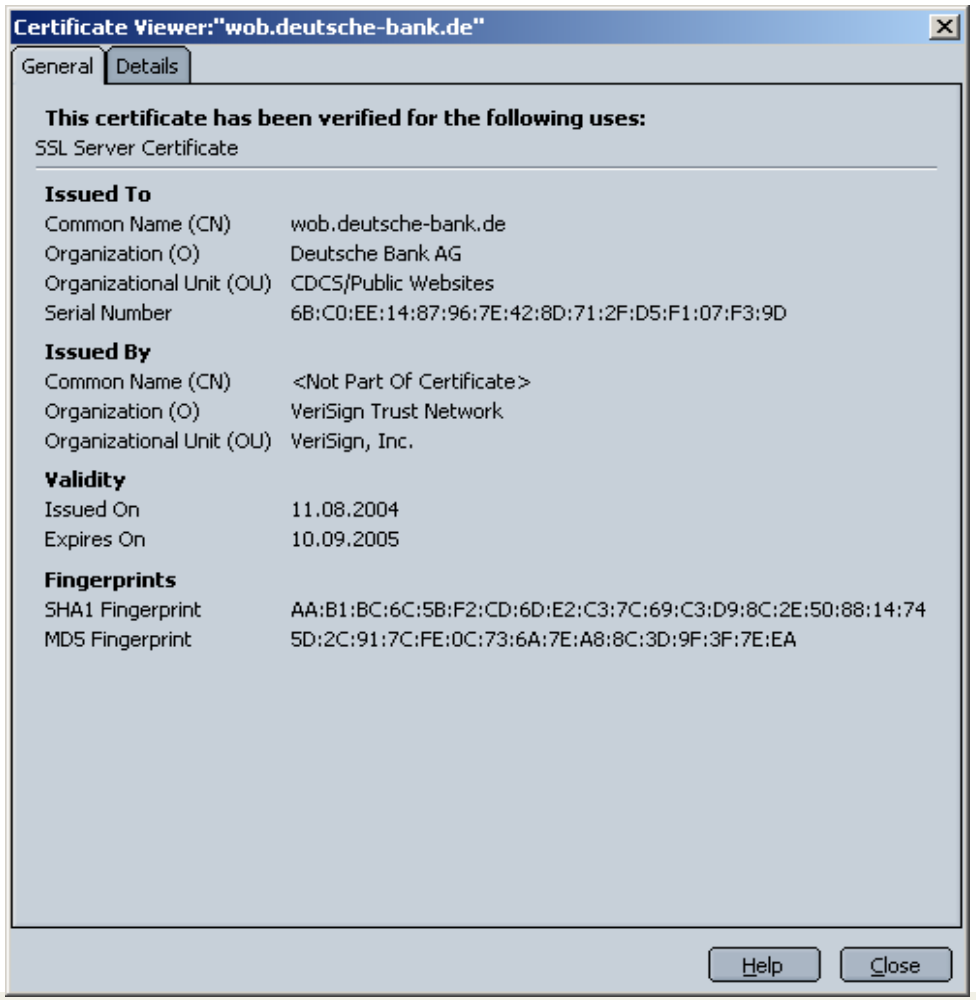
  SerialNumber: 1B

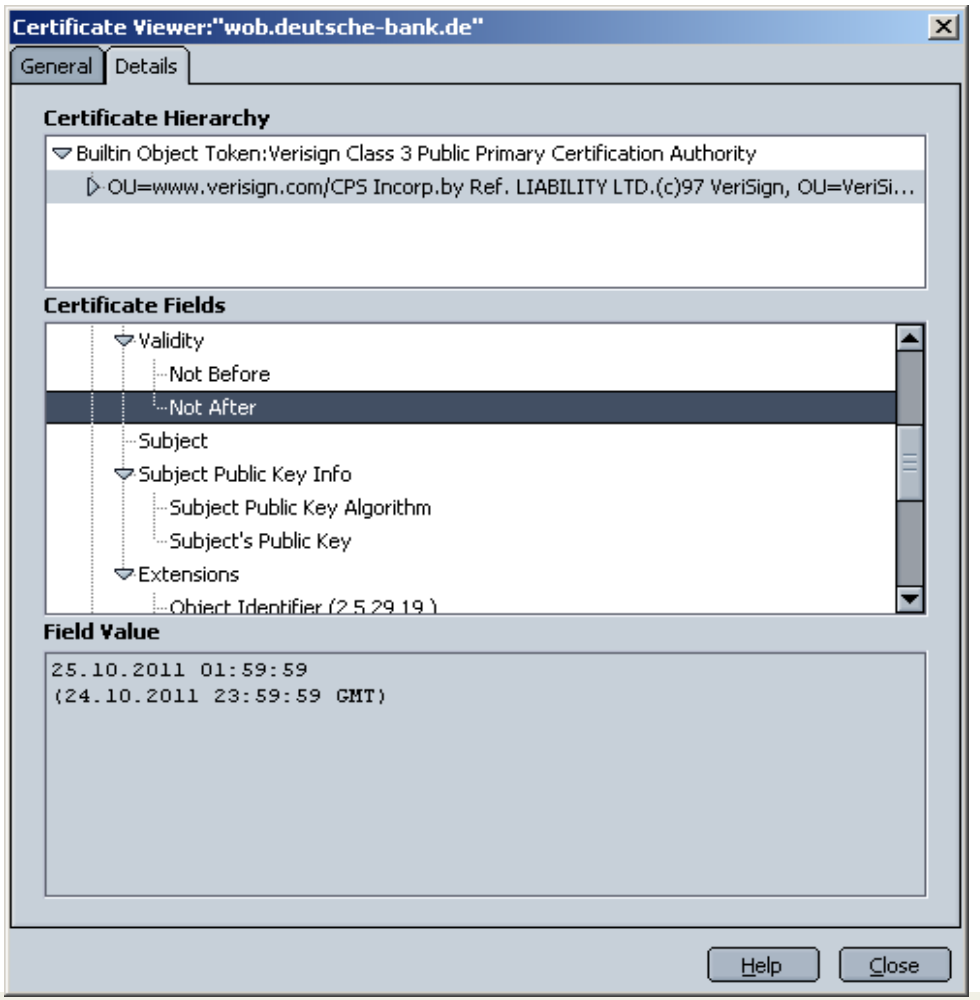
  Validity - NotBefore: Wed Sep 18 09:12:25 2002 (020918071225Z)

  NotAfter: Mon Jan 1 00:00:00 2007 (061231230000Z)

  SubjectKey: Algorithm RSA (OID 1.2.840.113549.1.1.1), NULL
  Public modulus (no. of bits = 1024):
  0 CACAF080 64F76D97 EA2662FA 81FF10FF .....
  Public exponent (no. of bits = 17):
  0 010001

Bsp.: X.509 v3-Zertifikat & seine CA





X.509 v3-Zertifikate: Details

- Im Wesentlichen entspricht ein X.509-Zertifikat der folgenden kryptographischen Formulierung:

  $Cert_{CA}(A) = \{A, K_E^A, T, L\}_{K_D^{CA}}$, wobei

  A – SubjectName
  T – Gültigkeitsbeginn des Zertifikates
  L – Gültigkeitsdauer
- CA bestätigt also durch eine digitale Signatur, dass K_E^A zu A gehört und dass K_E^A aktuell ist.
- Woher aber kennt B nun den öffentlichen Schlüssel K_E^{CA} der CA?
- Bildung von Zertifikatsketten

Ketten von Zertifikaten

- **Kette von Zertifikaten** der Certification Authorities CA_1, ..., CA_n
- CA_i bestätigt öffentlichen Schlüssel von CA_{i+1}
- CA_n bestätigt öffentlichen Schlüssel des Benutzers
- Wer bestätigt das Wurzelzertifikat?
  - In einem Web-Browser geschieht dies durch Einbau der Wurzelzertifikate in den Browser.
  - „Soziale“ Argumentation: Der Benutzer vertraut seiner Software (Browser) und somit auch den eingebauten Wurzelzertifikaten.

Public Key Infrastructure (1)

- **Public Key Infrastruktur (PKI)**

  "Public Key Infrastructure (PKI) provides the means to bind public keys to their owners and helps in the distribution of reliable public keys in large heterogeneous networks. " NIST

  The set of hardware, software, people, policies and procedures needed to create, manage, store, distribute, and revoke Public Key Certificates based on public-key cryptography. IETF (PKIX WG)

Public Key Infrastructure (2)

- **Public Key Infrastruktur (PKI)**

  “A system of CAs (and, optionally, RAs and other supporting servers and agents) that perform some set of certificate management, archive management, key management, and token management functions for a community of users in an application of asymmetric cryptography” IETF (RFC 2828, 4949)

Public Key Infrastructure (3)

- **Komponenten einer PKI**
  - Certification Authority (CA):
    - Stellt Zertifikate aus und signiert sie
    - Veröffentlicht aktuelle Zertifikate
    - Erstellt und veröffentlicht Listen von ungültigen Zertifikaten (***Certificate Revocation List***, CRLs),
    - bietet Online Certificate Status Protocol (OCSP) für Clients
  - Registration Authority (RA):
    - Arbeitet CA zu, bürgt für die Verbindung zwischen öffentlichem Schlüssel und Identitäten/Attributen der Zertifikatsinhaber
  - Verzeichnisdienst: i.d.R. LDAP, Verteilung der Zertifikate und CRLs
  - Zeitstempeldienst: signierte Zeitstempel (Gültigkeitsdauern, …)

Fazit zu Signaturen, PKI

- Wissenschaftliche Fragestellungen sind im Prinzip schon lange gelöst
- Dem PKI-Hype Anfang 2000er folgte die Ernüchterung:
  - Das Etablieren einer Unternehmens-PKI ist aufwendig
  - Das Ausrollen von Zertifikaten und Signaturkarten ist aufwendig, Nutzen war gering (Anwendungen?)
  - Sperrlistenverwaltung, Trust-Center-Aufbau etc. schwierig
  - Unternehmensübergreifende PKI-Strukturen gibt es kaum (**Haftung**?)
- **Einige Standards zur Thematik:**
  - X.509: Standard für digitale Zertifikate
  - PKCS: De-Facto Standards, definiert von RSA (vgl. RFC 3447 (und RFC 8017) = PKCS#1)
  - PKIX: PKI-Gesamtstandard für das Internet
  - ISIS-MTT: deutscher PKI-Gesamtstandard, angelehnt an PKIX

S/MIME – Secure Multipurpose Internet Mail Extension

- Erweitert den MIME-Standard um Sicherheitsdienste
- Ursprünglich von der Firma RSA entwickelt
- Nähere Informationen zur aktuellen Version (3.1):
  - http://www.ietf.org/html.charters/smime-charter.html
  - Relevante RFCs: RFC 8550 (5750, 3850) (Certificate Handling), RFC 8551 (5751, 3851) (Message Specification), RFC 5652 (3852) (CMS = Cryptographic Message Syntax)
- Automatische Unterstützung in verschiedenen E-Mail-Programmen wie z.B.:
  - Outlook, Outlook Express, Lotus Notes, Netscape

Sicherheitsdienste von S/MIME

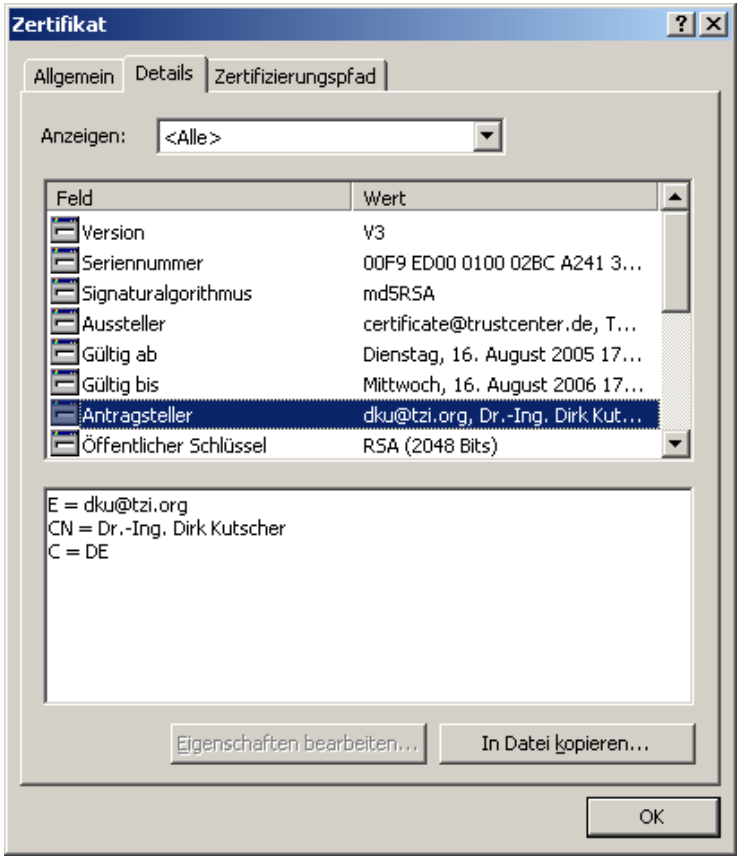
- Ähnliche Sicherheitsdienste wie bei PGP:
  - Verschlüsselung von Nachrichteninhalten (*envelop*):
    - Hybrides Verfahren
  - Signieren der Daten (inkl. E-Mail-Anhängen)
  - Signieren der Daten (inkl. E-Mail-Anhängen), aber Signatur von Nachricht getrennt (*clear signing*)
    - Nicht S/MIME-fähige Empfänger können die Mail auch verarbeiten
  - Verschlüsselung und Signatur
    - Zuerst Signatur, dann Nachricht verschlüsseln
  - Signierte Bestätigungen (*signed receipts*)
    - Stellen Verbindlichkeit sicher
- Verwendete kryptographische Verfahren (~~obsolet~~): RSA, ~~3-DES~~, AES, ~~40-Bit RC2~~, ~~SHA-1, MD5~~, DSA, ECDSA, EdDSA, Diffie-Hellman, ECDH

Authentisierung in S/MIME

- **Bekanntes Problem**: Woher weiß ich, dass es sich wirklich um den öffentlichen Schlüssel des Kommunikationspartners handelt?
- Anstelle Web of Trust: X.509v3-Zertifikate
- X.509v3-Zertifikate werden mit der verschlüsselten bzw. signierten Nachricht mitgeschickt.
- Das S/MIME-fähige E-Mail-Programm übernimmt dies für den Absender.
- Ausstellung von X.509v3-Zertifikaten durch eine Certification Authority (CA), z.B.:
  - VeriSign (DigitalID), Telekom, TC-Trustcenter in Hamburg, organisationsinterne CA

Beispiel für ein S/MIME-Zertifikat

- X.509 v3 Zertifikat
- Bindung des öffentlichen Schlüssels an den Namen und die E-Mail-Adresse des Antragstellers
- Aussteller des Zertifikates ist TC Trustcenter
- Später in diesem Kurs: Serverzertifikate



Vergleich PGP – S/MIME (1)

Merkmal	PGP	S/MIME
Verschlüsselung	+ (hybrid)	+ (hybrid)
Digitale Signaturen	+	+
Verschlüsselung und Signatur von Anhängen Mgl.	+ (sogar einzelne Dateien)	+
Stärke der kryptographischen Verfahren	Hoch (ursprünglich Einsatz von patentierten Verfahren wie IDEA)	Hoch, aber auch schwache Verfahren möglich wie RC2/40bit (Exportbeschränkungen)
Authentisierung von öffentlichen Schlüsseln	Web of Trust	X.509v3-Zertifikate

Vergleich PGP – S/MIME (2)

Merkmal	PGP	S/MIME
Rücknahme von Schlüsselpaaren	+ (Benutzer selbst verantwortlich für Rücknahme)	+ (Benutzer selbst verantwortlich für Rücknahme; CRL)
Kosten	Freie Software, früher kommerzielle Fassung 50$	Kosten für X.509-Zertifikate: z.B. ca. $20 pro Jahr, freemium Kostenlos in der Uni
Signed Receipts	-	+
Unterstützung in E-Mail-Programmen	Plug-Ins	Direkter Einbau in gängige E-Mail Programme wie Outlook Express
Einsatzgebiet	Vor allem Privatnutzer	Auch kommerzieller Einsatz

Server
Netzwerk
Security
Desktop
Administration
Entwicklung
Hardware
Szene
Special
IT-Profimarkt
Fachbücher

Heftarchiv
Service
Bestellen
Kontakt

Grüne Abgeordnete setzen PGP/GnuPG im Bundestag durch

19.12.2011 **Grüne Bundestagsabgeordnete haben einer weiteren Open-Source-Software den Weg in den deutschen Bundestag geebnet. Bundestagsmitarbeiter können nun neben S/MIME auch GnuPG zur Verschlüsselung von E-Mails verwenden, melden die Grünen.**

"Gerade für den Kontakt von Parteien und Politikern mit Whistleblowern und JournalistInnen" sei sichere Kommunikation eminent wichtig, erklärt Konstantin von Notz in einer Presseerklärung. Der Bundestagsabgeordnete aus Schleswig-Holstein setzt sich nach eigener Aussage für Open-Source-Software ein und erklärt:"Wir sind uns unserer Verantwortung bewusst und haben uns daher bei der Bundestagsverwaltung und im Ältestenrat dafür eingesetzt, dass der IT-Service des Bundestags die entsprechende Software anbietet und neben der bisher für Abgeordnete verfügbaren S/MIME-Zertifikat-Verschlüsselung nun auch eine Verschlüsselungssoftware auf Basis freier Software,offener Standards und dezentraler Schlüsselverwaltung über die offizielle Bundestags-IT ermöglicht."

Auf seiner Webseite und der der Grünen steht dementsprechend auch der benötigte PGP-Key bereit, auf der Seite der Partei erklärt zudem ein Video und eine Pressemitteilung, wie Anwender die freie Verschlüsselung verwenden können, um sichere Kommunikation zu betreiben.

IT-Sicherheit:

Authentisierung, Sicherheitsprotokolle

Authentisierung

- **Ziel**
  - Eindeutige Identifikation und Nachweis der Identität
  - Abwehr von Spoofing-Angriffen
- **Problem:**
  - Nicht nur Mensch-zu-Gerät Interaktion, sondern auch
  - Gerät-zu-Gerät bzw. Gerät/Dienst-zu-Gerät/Dienst
- **Bemerkung:** zunehmende Vernetzung u. Miniaturisierung:
  Geräte-zu-Geräte-Kommunikation steigt rapide an!
- **Notwendig:** Konzepte und Verfahren, um sowohl
  - menschliche Individuen
  - als auch Geräte (Web-Server, Laptop, Mobiltelefon, ...) und
  - Dienste (Dateisystem, Amazon, Bankportal, ...)

  eindeutig zu identifizieren

Beispiele für die Authentisierung

- Viele Beispiele in der Praxis:
  - Benutzer gegenüber PC (Mensch-zu-Gerät)
  - Mobiltelefon gegenüber Netzbetreiber (Gerät-zu Gerät)
  - Bankkunde gegenüber Bankautomat (Mensch-zu-Gerät)
  - WWW-Server gegenüber Nutzer (Dienst-zu-Mensch)
  - Laptop gegenüber Access Point im WLAN (Gerät-zu-Gerät)

Arten der Authentisierung

- **Authentisierung durch**
  - Wissen: z.B. Passworte, PINs, kryptogr. Schlüssel („Was ich weiß“)
  - Besitz: z.B. Smartcard, USB-Token, SIM-Karte („Was ich habe“)
  - biometrische Merkmale: z.B. Fingerabdruck („Was ich bin“)
  - Mehrfaktor-Authentisierung: Kombination von Konzepten:
    - z.B. SIM-Karte + PIN, Biometrie + Zugangskarte, …

- **Beispiel**: 2-Faktor-Authentisierung beim Mobiltelefon:
  (1) Authentisierung über PIN (Wissen) gegenüber SIM-Karte
  (2) und Besitz der SIM-Karte (enthält geheimen Schlüssel K_{SIM})
  SIM-Karte authentisiert sich gegenüber dem Netz mit K_{SIM}

Authentisierung durch Passwörter

- Vereinbartes Geheimnis zwischen dem Benutzer und dem System
  - Meist speichert das System nur einen kryptographischen Hashwert des Passwortes
    - Zum Beispiel in Unix: `/etc/passwd`
- Ablauf der Authentisierung durch Passwörter:
  - Angabe der Identität (Benutzerkennung, Login)
  - System: „Geben Sie Ihr Passwort ein!“
  - Benutzer: Eingabe des Passwortes
  - Überprüfung durch das System:
    - Berechne den kryptographischen Hashwert des Passwortes
    - Vergleiche das Ergebnis dieser Berechnung mit dem Eintrag in der Passwort-Datei für die Benutzerkennung

Angriffe auf Passwort-Systeme (1)

- Angriffe auf die Passwort-Eingabe:
  - „Über die Schulter schauen“
  - Mitschneiden („Sniffen“) von Passwörtern in einem LAN bei unverschlüsselter Kommunikation); Keylogger im Tastaturkabel
  - Nicht vertrauenswürdige Rechner: Keylogger im System
    - Zum Beispiel in einem öffentlichen Terminalraum
  - Passwort auf Post-it-Zettel
- Social Engineering-Angriffe
  - Beispiel: Telefonanruf an einen Administrator:

    „Hallo, hier ist die Sekretärin der Geschäftsführung. Mein Chef braucht dringend das Administrator-Passwort für die Anwendung XYZ!“

Angriffe auf Passwort-Systeme (2)

- Angriffe auf den Passwort-Speicher
  - Besonders leicht bei unverschlüsselten Passwörtern:
    Nie Passwörter im Klartext speichern
    (besser kryptographische Hashwerte)!
  - Nutzen von Logging-Informationen:

    Ein nicht-existenter Login-Name wie z.B. **`gzu%8yp`** in einer Log-Datei ist ein lohnendes Ziel.
  - Password-Cracking (folgende Folie)

Password-Cracking

- Password-Cracker: **Offline-Angriff**
  - Enthalten Wörterbücher (***Dictionaries***): u.U. in verschiedenen Sprachen
  - Unterstützen Heuristiken, um Kombinationen aus Sonderzeichen und Wörtern durchzuprobieren:
    Das Passwort `Cars%ten` kann leicht ermittelt werden
  - Brute-Force-Angriff (vor allem bei zu kurzen Passwörtern erfolgreich)
  - Üblicherweise können mit einem Password-Cracker ungefähr 21-25% der Passwörter eines Systems ermittelt werden (Studie).

- Gängige Password-Cracker:
  - Crack, L0phtCrack
  - Cain & Abel, http://www.oxid.it/cain.html, für Windows-Systeme
  - John the Ripper (JtR), www.openwall.com/john, für Unix und Windows

Wahl guter Passwörter

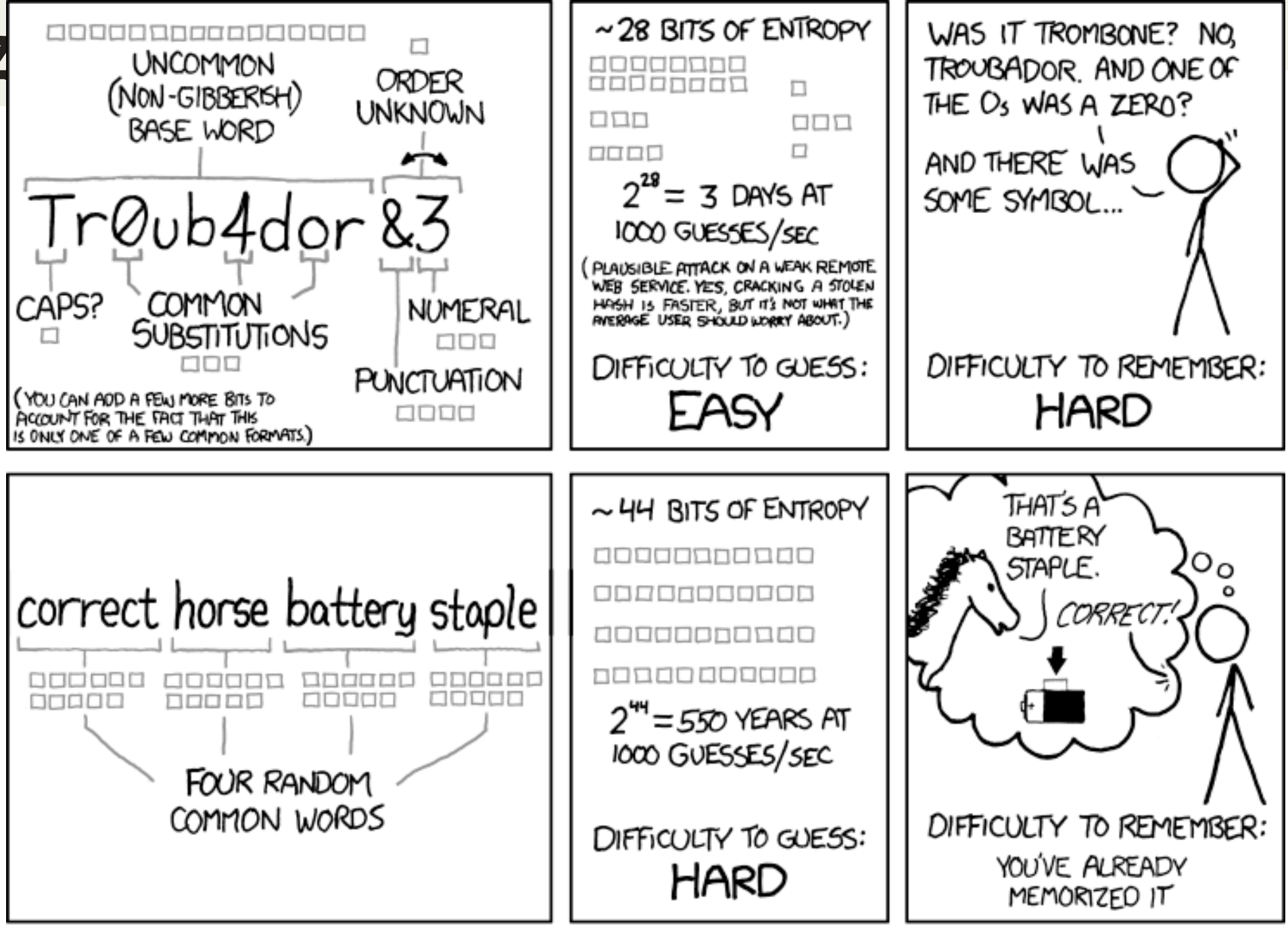
- Richtlinien für die Wahl guter Passwörter
  1. Nicht weniger als acht Zeichen
  2. Kein Name oder Wort, das in einem Wörterbuch nachschlagbar ist
  3. Mindestens ein Sonderzeichen
  4. Keine Folge von Zeichen, die auf einer Tastatur nebeneinander liegen

... und Passwörter sollten in regelmäßigen Abständen geändert werden!

Wahl guter Passwörter

- Richtlinien für die Wahl guter Passwörter
  1. Nicht weniger als acht Zeichen
  2. Kein Name oder Wort, das in einem Wörterbuch nachschlagbar ist
  3. Mindestens ein Sonderzeichen
  4. Keine Folge von Zeichen, die auf einer Tastatur nebeneinander liegen

... und Passwörter sollten in regelmäßigen Abständen geändert werden!



THROUGH 20 YEARS OF EFFORT, WE'VE SUCCESSFULLY TRAINED EVERYONE TO USE PASSWORDS THAT ARE HARD FOR HUMANS TO REMEMBER, BUT EASY FOR COMPUTERS TO GUESS.

http://xkcd.com/936/

NIST SP 800-63B (June 2017)

- “**Verifiers SHALL require subscriber-chosen memorized secrets to be at least 8 characters in length. Verifiers SHOULD permit subscriber-chosen memorized secrets at least 64 characters in length. Truncation of the secret SHALL NOT be performed. […]** Verifiers SHOULD offer guidance to the subscriber, such as a password-strength meter [Meters], to assist the user in choosing a strong memorized secret.”

  …/

NIST SP 800-63B (June 2017)

- "**Verifiers SHOULD NOT impose […] composition rules** (e.g., requiring mixtures of different character types or prohibiting consecutively repeated characters) for memorized secrets**. Verifiers SHOULD NOT require memorized secrets to be changed arbitrarily (e.g., periodically).** However, verifiers SHALL force a change if there is evidence of compromise of the authenticator."

https://pages.nist.gov/800-63-3/sp800-63b.html:
5.1.1.2 Memorized Secret Verifiers

Sicherheit von Unix-Passwörtern

- Design-Fehler im ursprünglichen Unix:
  - Passwort-Datei `/etc/passwd` ist zwar verschlüsselt, aber world-readable!
  - Gefahr eines Wörterbuchangriffs durch einen Password-Cracker (offline!)
  - Lösung des Problemes:
    shadow-Passwort Datei; `/etc/passwd` enthält dann Dummy-Einträge

- Sicherheit von Unix-Passwörtern
  - Unix-Passwörter können maximal acht Zeichen lang sein,
  - 95 druckbare Zeichen möglich

  Es gibt somit $\mathbf{95^8 \approx 2^{52.6}}$ Möglichkeiten zur Passwortwahl.
  Eine gut ausgestattete Organisation (z.B. NSA) kann also jedes Unix-Passwort knacken.

Zusammenfassung: Authentisierung durch Passwörter

- Passwort-Authentisierung angreifbar
  - Dictionary-Angriffe
  - Sniffing
  - Social Engineering
  - Angriffe auf die Passwort-Speicherung
- **Lösungen: u.a.**
  - Einmal-Passwort-Verfahren, OTP (One-time Passwort)
    z.B. S/Key (Hash-Kette) oder RSA SecureID („Key Fob“)
  - Authentisierung zwischen Systemen: Ticket-basiert mit Nachweis, dass man der Eigentümer des Tickets ist (z.B. Kerberos)
- **Vielen dieser Lösungen ist gemeinsam**:
  Sie basieren auf einem Challenge-Response Verfahren (später).

Hardware-basierte OTP-Verfahren: ID-Token

- OTP-Verfahren zur Authentisierung beim Server
  Beispiel: Online-Banking
- Benutzer erhält ein Hardware-Token
- Token besitzt eindeutige Nummer; Server kennt diese
- Token berechnet periodisch eine neue Zahl (Code), z.B. alle 60 Sekunden, Code ist das OTP
- Tokencode hängt von einem Seed ab, der Zeit, der Seriennummer des Tokens und ggf. einer PIN
- Benutzer: Eingabe des OTP an einem Terminal
  dazu: ablesen des Passwortes vom Display des Tokens
- Server muss Seed, Zeit, Seriennummer, PIN kennen, generiert ebenfalls das Passwort und vergleicht beide

Beispiel: RSA SecureID Token

- Basis für 2-Faktor-Authentisierung (Wissen und Besitz)

  - Zeit-synchronisiertes Vorgehen
    - Synchronisation von Server (RSA ACE/Server) und Token
  - Admin des Servers richtet Benutzer-Account ein, mit:
    - Token-Nummer und 64-Bit Seed s
    - Seed s wird auch auf Token gespeichert
    - Token wird an Benutzer ausgegeben

  - Erzeugen von OTPs:
    - alle 60 Sekunden generieren Token u. Server neues Passwort:

AES-Hashwert: Tokencode = AES(TokenId | s | Zeit)

Beispiel: S/Key

- Einmal-Passwörter über Hash-Ketten generieren
- Benutzer und System vereinbaren Passwort *s*
- Hash-Funktion *f* wird *i*-mal wiederholt: f^i
- Benutzer erhält *s* und einen PDA mit *f*
  - Alternativ: Benutzer erhält für n Einlogvorgänge $f^i(s)$ für $0 < i < n$
- System fordert mit *n* heraus (und merkt sich $i=n$)
  - Benutzer antwortet mit $f^i(s)$
- Beim nächsten mal fordert System mit $i:=i-1$ heraus
  - Benutzer antwortet mit $f^{i-1}(s)$ (nicht aus $f^i(s)$ ableitbar!)

Probleme von S/Key

- Wie alle Passwort-Mechanismen anfällig gegen Middle-Person-Angriff
- Untergeschobener Server kann Passwort für *j* abfragen mit $j < i$; daraus kann $f^i(s)$ berechnet werden

- Lösung: Server-Authentisierung

Authentisierung durch Biometrie (1)

- ***„Gilead besetzte die nach Efraim führenden Übergänge über den Jordan. Und wenn efraimitische Flüchtlinge (kamen und) sagten: Ich möchte hinüber!, fragten ihn die Männer aus Gilead: Bist du ein Efraimiter? Wenn er Nein sagte, forderten sie ihn auf: Sag doch einmal «Schibbolet». Sagte er dann «Sibbolet», weil er es nicht richtig aussprechen konnte, ergriffen sie ihn und machten ihn dort an den Furten des Jordan nieder. So fielen damals zweiundvierzigtausend Mann aus Efraim.“, Die Bibel, Richter 12:5-6***

- Erster (?) dokumentierter militärischer Einsatz eines Protokolls zur Authentisierung basierend auf einem biometrischen Merkmal:
  - Hier: der Akzent
- Was ich bin („something you are“)

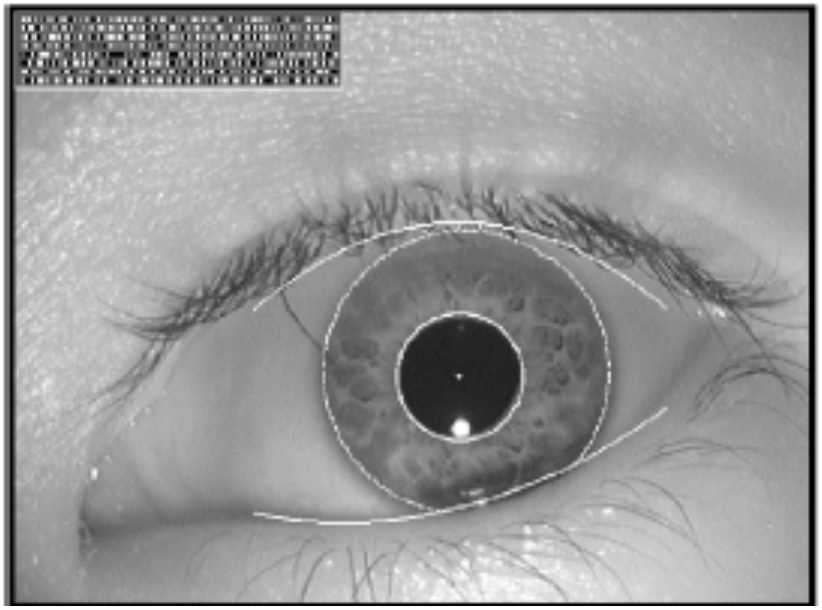
Authentisierung durch Biometrie (2)

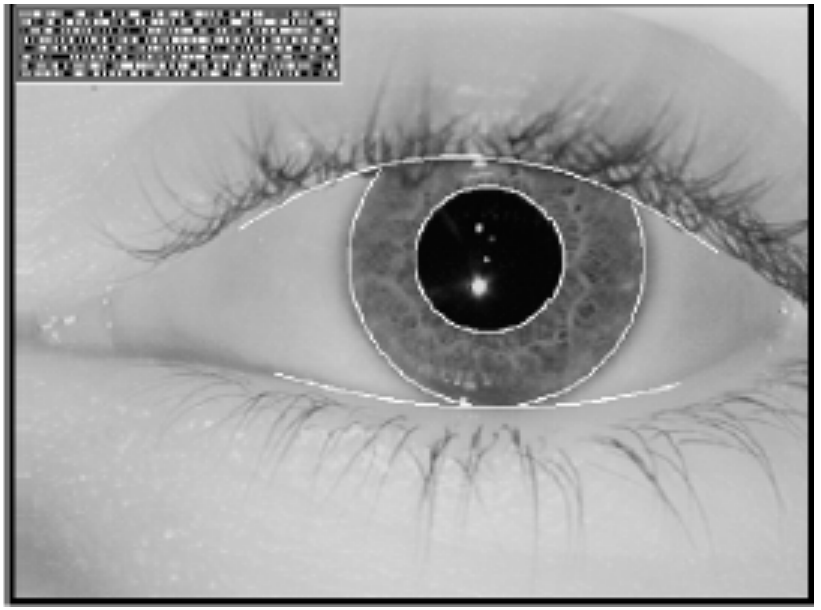
- **Biometrisches Merkmal:**
  Verhaltenstypische oder physiologische Eigenschaft eines Menschen, die diesen eindeutig charakterisieren
- Anforderungen an biometrische Merkmale:
  - **Universalität**: Jede Person besitzt das Merkmal.
  - **Eindeutigkeit**: Merkmal ist für jede Person verschieden
  - **Beständigkeit**: Merkmal ist unveränderlich
  - quantitative **Erfassbarkeit** mittels Sensoren
  - **Performance**: Genauigkeit und Geschwindigkeit
  - **Akzeptanz** des Merkmals beim Benutzer
  - **Fälschungssicherheit**

Biometrische Verfahren: Beispiele

- **Fingerabdruckerkennung:**
  - Weit verbreitetes Verfahren für viele Anwendungen: preiswerte, kompakte Sensoren sind verfügbar (auf Mobiltelefon, PDA, Laptop, ...)
- **Schreibdynamik ist** besonders geeignet für Willenserklärung (z.B. Signaturen): Unterschreiben als bewusster, willentlicher Akt
- **Iris-Erkennung** sehr fälschungssicher, aber aufwendige Technik und geringe Akzeptanz (vgl. Fraport-Pilot-Projekt)
- **Gesichtskontrolle:**
  - Gute Akzeptanz, aber z.Zt. noch nicht ausgereift, vgl. http://www.bsi.bund.de/literat/studien/biop/biopabschluss.pdf; Studie zu Gesichtserkennungssystemen zum Einsatz bei Lichtbildausweisen

Iris-Erkennung





Problembereiche

- Abweichungen zwischen Referenz- und Verifikationsdaten sind unvermeidlich.
- Zwei Fehlertypen:
  - Berechtigter Benutzer wird abgewiesen
    ⇨ Akzeptanzproblem (***false negative, FRR false rejection rate***)
  - Unberechtigter wird authentisiert, Kontrollen zu locker
    ⇨ Sicherheitsproblem (***false positive, FAR false acceptance rate***)

  Leistungsmaße zur Bewertung der Güte eines Systems
- Ergebnisse einer dreijährigen Studie BioTrusT: www.biotrust.de:

  Aktuelle Produkte besitzen noch erhebliche Fehlerraten
  (Stand 2002, aber durchaus noch aktuell).