Sicherheit

Die Mitarbeiter von http://mitschriebwiki.nomeata.de/

27. Dezember 2016

Inhaltsverzeichnis

I. Über dieses Skriptum

Dies ist ein erweiterter Mitschrieb der Vorlesung „Sicherheit“ von Herrn Prof. Müller-Quade im Sommersemester 2012 am Karlsruher Institut für Technologie (KIT).

I.1. Wer

Mitgeschrieben in der Vorlesung hat Sabine Oechsner.

I.2. Wo

Alle Kapitel inklusive LaTeX-Quellen können unter http://mitschriebwiki.nomeata.de abgerufen werden. Dort ist ein *Wiki* eingerichtet und von Joachim Breitner um die LaTeX-Funktionen erweitert. Das heißt, jeder kann Fehler nachbessern und sich an der Entwicklung beteiligen. Auf Wunsch ist auch ein Zugang über *Subversion* möglich.

II. Einleitung

II.1. Was ist Sicherheit?

Was soll geschützt werden?

- Geheimnisse
- Daten/Wissen
- Dienste/Ressourcen/Infrastruktur
- Kommunikation
- Ansehen
- Rechte (informelle Selbstbestimmung)/Souveränität
- Hardware, Maschinen, (Versorgungs)Anlagen
- Vermögen, Besitz
- Urheberschaft/-recht
- Gesundheit/Leben

Vor wem?

- einzelne (Gelegenheits)Kriminelle
- Laien/"Kinder", unerfahrene Benutzer
- Spionage, Geheimdienste
- organisierte Kriminalität
- Saboteure
- interessengesteuerte, nicht-böswilligeÖrganisationen (Staat, Firmen)
- interne Angreifer (Bekannte, Verwandte, Mitarbeiter)
- Trojaner
- Vandalismus (Skript-Kiddies)
- private Feinde/Konkurrenten

Wie?

- Obscurity, Steganographie
- Kryptographie (Verschlüsselung, Signaturen)
- physikalische Sicherheit (Bunker, Tresor)
- Security Awareness (Mitarbeiterschulung)
- Policies, vorgeschriebene Abläufe
- Honey Pots, Intrusion Detection

ganzheitliche Sicherheit (Vermeidung von Schwachstellen):

- Angriffe auf Algorithmenebene (Verschlüsselung brechen, Signatur fälschen)
- Angriffe auf Protokollebene (z.B. Replay-, Man-in-the-Middle-Attacken)
- Angriffe auf Implementierungsebene (z.B. Bugs ausnutzen, Overflows, Injections)
- Angriffe aus Betriebsumgebung (z.B. Power Analysis, Timingattacken)
- Angriffe über ëxterne Komponenten"(z.B. Social Engineering, Phishing)

II.2. Wichtigste Sicherheitsziele

- Confidentiality (Schutz vor unbefugten Lesezugriffen)
- Integrity (Schutz vor unbefugten Schreibzugriffen (Veränderung/Verfälschung)
- Availability (Möglichkeit, Ressourcen/Dienste in der vorgesehenen Form zu nutzen)

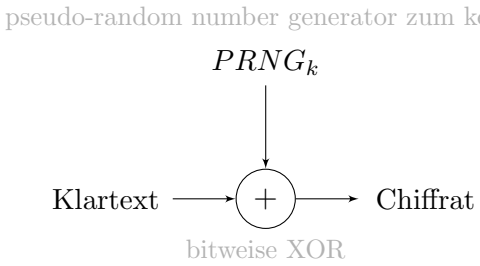
II.3. Praxisprobleme

- Abwägung Kosten/Nutzen
- gesetzliche Regelungen
- ethische/soziale Probleme
- Grunddilemma unterschiedlicher Begriffe und Definitionen
- Snake Oil (z.B. Enigma, Quantenkryptographie + One-Time-Pad)

III. Symmetrische Verschlüsselung

FIXME: Definition vernachlässigbare Funktion, S. 4

III.1. Stromchiffren



III.1.1. Anforderungen

- PRNG muss effizient berechenbar sein
- Pseudozufall ununterscheidbar von echtem Zufall
  (formal: gegeben Orakel $\mathcal{O}_{ideal}$, welches echten Zufall ausgibt, und $\mathcal{O}_{real}$, welches $PRNG_k$ mit geheimem Schlüssel k implementiert, gilt für alle (Polynomialzeit)Angreifer $\mathcal{A}$:

  $$\left|Pr[\mathcal{A}^{\mathcal{O}_{ideal}} \to 0] - Pr[\mathcal{A}^{\mathcal{O}_{real}} \to 0]\right|$$

  ist vernachlässigbar in $|k|$).

III.2. Blockchiffren

III.2.1. Definition

Seien k ein Schlüssel aus dem Schlüsselraum $\mathcal{K}$, A und B sind Ein- bzw. Ausgabealphabete und n und m die zugehörigen Blocklängen. Eine Blockchiffre ist eine Familie von injektiven Abbildungen $\{f_k \colon A^n \to B^m\}_{k \in \mathcal{K}}$.

FIXME: Bild Blockchiffre, S. 4

III.2.2. Anforderungen

- gegeben ein Orakel $\mathcal{O}_{ideal}$, welches eine zufällige Injektion $A^n \to B^m$ implementiert, und $\mathcal{O}_{real}$, welches f_k mit geheimem Schlüssel k implementiert, gilt für alle (Polynomialzeit)Angreifer $\mathcal{A}$: $\left|Pr[\mathcal{A}^{\mathcal{O}_{ideal}} \to 0] - Pr[\mathcal{A}^{\mathcal{O}_{real}} \to 0]\right|$ ist vernachlässigbar in $|k|$.
- gegeben k, müssen f_k und f_k^{-1} effizient berechenbar sein

III.2.3. Beispiel: DES (Data Encryption Standard)

FIXME: Bild DES, S. 5

Eigenschaften

- bis heute strukturell ungebrochen
- aber: Schlüssel zu kurz (Brute-Force-Attacken sind heute praktikabel)

→ Abhilfe: 3DES (Chiffrat = $DES_{k_3}(DES^{-1}_{k_2}(DES_{k_1}(Nachricht)))$)
→ Warum nicht 2DES? Antwort: Meet-in-the-Middle-Attacken

Meet-in-the-Middle (gegen 2DES):

FIXME: Bild Meet in the Middle, S. 6

Known-Plaintext-Angriff, gegeben ein Klartext-Chiffrat-Paar (M, C):

1. Vorwärts-Schritt: Tabelliere $(DES_k(M), k)$ für alle Schlüssel $k \in \{0,1\}^{56}$.
2. Sortiere die Tabelle.
3. Rückwärts-Schritt: Für jedes $k \in \{0,1\}^{56}$ berechne $(DES_k(C)$ und suche Tabelleneintrag mittels binärer Suche.

Aufwand: $\approx 56 \cdot 2^{56}$ für das binäre Sortieren, $\approx 2^{56}$ für die binäre Suche, insgesamt also nur ≈ 56 mal mehr Aufwand als bei DES

III.2.4. Beispiel: Rijndael/AES (Advanced Encryption Standard)

- 128 bit Blocklänge
- 3 Varianten:
  - 128, 192, 256 bit Schlüssel
  - 10, 12, 14 Runden
- Darstellung von State und Rundenschlüssel als 4×4-Byte-Matrix
- Ablauf einer Runde in 4 Schritten: **FIXME:** Bild AES, S. 7
- Bestimmen der Rundenschlüssel:
  - teile Schlüssel in 4-Byte-Worte
  - berechen $W[i] := W[i-4] \oplus W[i-1]$ und ab und zu Byteinvertierungen
- mögliche Schwäche: AES lässt sich als geschlossene algebraische Gleichung schreiben und ist damit theoretisch mathematisch brechbar

III.2.5. Betriebsmodi

ECB (Electronic Codebook Mode)

FIXME: Bild ECB, S. 8

Nachteile:

- gleiche Klartextblöcke werden auf gleiche Chiffratblöcke abgebildet
- Angreifer kann Blöcke vertauschen, löschen, duplizieren

Vorteile:

- Übertragungsfehler (Bitflips) auf den betroffenen Block begrenzt[1]
- perfekt parallelisierbar (zum Ver- und Entschlüsseln der Blöcke ist jeweils nur der Schlüssel nötig)
- verschlüsselte Datenspeicher blockweise bearbeitbar

CBC (Cipher Block Chaining)

FIXME: Bild CBC, S. 8

Vorteile:

- Nachteile von ECB beseitigt
- Enschlüsselung selbstsynchronisierend (Klartextblock wird nur aus den letzten beiden Chiffratblöcken berechnet) → wahlfreier Lesezugriff

Nachteile:

- geringer Bandbreitenverlust, da Initialisierungsvektor übertragen werden muss
- Fehler breiten sich auf einen weiteren Block aus

OFB (Output Feedback Mode)

FIXME: Bild OFB, S. 9

Vorteile:

- Entschlüsselung muss nicht effizient sein
- keine Fehlerübertragung bei Bitflips
- Pseudozufallsstrom vorberechenbar

Nachteile:

- gleicher Initialisierungsvektor bewirkt: $c_1 \oplus c_2 = m_1 \oplus m_2$
- nicht robust gegen Verlorengehen ganzer Blöcke
- Angreifer kann gezielt Klartextbits kippen

[1] aber: betrifft den ganzen Block, da Verschlüsselung jedes einzelnen Bits im Block von jedem Bit im Block abhängig

CTR (Counter Mode)

FIXME: Bild CTR, S. 9

Nachteile:

- wie OFB

Vorteile (wie OFB und ECB):

- gut parallelisierbar
- Pseudozufallsstrom vorberechenbar
- Fehlerfortpflanzung auf Blöcke begrenzt
- wahlfreier Zugriff auf verschlüsselten Speicher
- muss nicht invertierbar sein

IV. Hashfunktionen

(hier immer krytographische)

IV.1. Anwendungen

- Passwortdateien
- Time Stamping
- Digitale Signaturen (s. VII)
- RSA-ES-OAEP (s. VI.4.3)

IV.2. Eigenschaften

einer Hashfunktion $h\colon \{0,1\}^* \to \{0,1\}^k$:

- Einwegfunktion[1]
- preimage resistant: gegeben $x \in \{0,1\}^k$ ist es schwierig, ein m zu finden mit $h(m) = x$
- collision resistant: es ist schwierig, m und m', $m \neq m'$, zu finden mit $h(m) = h(m')$
- die Ausgabelänge von Hashfunktionen sollte mindestens 160 bit sein (wg. Meet-in-the-Middle-Angriff)

IV.3. Merkle-Damgård-Konstruktion

gegeben: eine Kompressionsfunktion $f\colon \{0,1\}^{2k} \to \{0,1\}^k$ (Kandidat: Blockchiffre)

FIXME: Bild Merkle-Damgard-Konstruktion, S. 10

Die Sicherheit der so entstandenen Hashfunktion hängt nur von der Sicherheit von f ab. Aber: eine gegebene Kollision lässt sich verlängern.

IV.4. Das Random Oracle Model

Manchmal stellt man sich Hashfunktionen vor wie echt zufällige „Orakel“ . Beweise im Random Oracle Model sind trotzdem nur Heuristiken, da ein Angriff die innere Struktur der Hashfunktion ausnutzen kann.

[1] die Annahme der Existenz von Einwegfunktionen ist eine stärkere Annahme als $P \neq NP$! → Hashfunktionen sind eine noch stärkere Annahme

IV.5. Der Angriff von Wang

Der Angriff von Wang findet Kollisionen von MD5 zu gegebenem Initialisierungsvektor. Diese wirken wie zufällig.

Problem (Beispiel Postscript):

- gegeben: Dokumente P und Q und eine Kollision $h(S) = h(R)$
- hashe „if“
- nimm dies als Initialisierungsvektor: Kollision $h(\text{if } S) = h(\text{if } R)$
- MD5 ist eine Merkle-Damgård-Konstruktion $\rightarrow$ Erweitern der Kollision zu
  - if $S = S$ then display P else Q
  - if $R = S$ then display P else Q
- Ergebnis: zwei Dokumente mit gleichem Hash, aber unterschiedlicher Inhalt wird angezeigt (P oder Q)

Daher gibt es zur Zeit einen neuen Wettbewerb SHA3.

IV.6. Symmetrische Authentifikation (MAC - Message Authentification Code)

Ein MAC ist eine Abbildung $MAC\colon \{0,1\}^* \times \{0,1\}^k \rightarrow \{0,1\}^k$.

Sicherheitsbegriff: ein Angreifer $\mathcal{A}$ mit Zugriff auf ein Orakel, das gültige MACs ausrechnet, darf keinen gültigen MAC mit zugehöriger Nachricht finden können, ohne das Orakel nach dieser Nachricht gefragt zu haben.

- Vorschlag 1: $h(key \| m) \rightarrow$ funktioniert nicht **FIXME:** warum?
- Vorschlag 2: $h(m \| key) \rightarrow$ funktioniert nicht **FIXME:** warum?
- HMAC: $h((key \oplus o\_pad) \| h((key \oplus i\_pad) \| m))$

FIXME: Bild HMAC, S. 12

V. Schlüsselaustausch

Verschlüsselung und Authentifikation benötigen einen gemeinsamen geheimen Schlüssel:

V.1. 3-Pass

(Schlüssel-/Nachrichtenaustausch mit mittelalterlichen Mitteln)

FIXME: Bild 3-Pass oben, S. 14

Problem: Man-in-the-Middle-Angriff möglich (Bote befestigt eigenes Schloss, statt Kiste zu Bob zu bringen) → Abhilfe: z.B. schwer fälschbare Siegel auf dem Schloss

modernere Idee:

Nachrichtentransfer über authentifizierten, aber nicht abhörsicheren Kanals mittels OTP (One-Time-Pad):

FIXME: Bild 3-Pass unten, S. 14

Problem: das funktioniert so nicht, da

$$(M \oplus OTP\_Alice) \oplus (M \oplus OTP\_Alice \oplus OTP\_Bob) = OTP\_Bob$$

V.2. Wide-Mouth-Frog

FIXME: Bild Wide-Mouth-Frog, S. 15

Probleme:

- benötigt vertrauenswürdige Zentrale
- Zentrale zuständig für Verbindungsaufbau zu Bob → DoS-Attacke möglich
- Alice benötigt guten Zufallsgenerator

V.3. Kerberos

FIXME: Bild Kerberos, S. 15

1. $(\text{alice}, \text{bob})$
2. $(Enc_{k_{AZ}}(T_Z, L, K_{AB}, \text{bob}), Enc_{k_{BZ}}(T_Z, L, K_{AB}, \text{alice}))$

3. $(Enc_{k_{AB}}(\text{alice}, T_A), Enc_{k_{BZ}}(T_Z, L, K_{AB}, \text{alice}))$
4. $Enc_{k_{AB}}(T_A + 1)$

Probleme:

- benötigt synchrone Uhren
- Chiffre muss non-malleable[1] sein, sonst evtl. $Enc_{T_{AB}}(T_A{+}1)$ aus $Enc_{T_{AB}}(T_A)$ berechenbar $\rightarrow$ Confidentiality $\not\Rightarrow$ Non-Malleability!

Die ältere Variante Needham-Schroeder hatte noch keine Zeitstempel, sondern Zufallszahlen. $\rightarrow$ Replay-Attacken möglich (z.B. Alice ist eine Kamera und der Angreifer spielt alte Aufnahmen wieder ein)

V.4. Merkle Puzzle

Alice wählt zufällig k Paare von Schlüssel und zufälliger Schlüsselnummer und verschlüsselt die Paare mit einer symmetrischen Chiffre mit immer neuem Schlüssel. Sie schickt alles an Bob zusammen mit genügend Hinweisen, sodass Bob jedes der verschlüsselten Paare „brechen“ kann. Bob entschlüsselt eines der Paare und antwortet Alice mit der Schlüsselnummer des von ihm gebrochenen Paares und die beiden kennen nun einen gemeinsamen Schlüssel.
Der „Vorteil“ von Alice und Bob gegenüber einem Lauscher ist leider nur gering (quadratisch).

V.5. Diffie-Hellman-Schlüsselaustausch (DH)

Seien eine (öffentlich bekannte) Gruppe G sowie ein Erzeuger g dieser Gruppe gegeben[2] (z.B. $\mathbb{F}_p^\times, p \in \mathbb{P}$ oder eine Gruppe auf einer Elliptischen Kurve):

FIXME: Bild DH, S. 16

Diffie-Hellman ist (beweisbar) sicher relativ zur Decisional-Diffie-Hellman-Annahme.

V.5.1. Decisional-Diffie-Hellman-Annahme

Asymptotisch ist es nicht in Polynomialzeit möglich, folgende Verteilungen zu unterscheiden: (g, g^a, g^b, g^{ab}) und (g, g^a, g^b, g^c) mit a, b, c zufällig.

V.5.2. Man-in-the-Middle-Angriff

FIXME: Bild Man-in-the-Middle-Angriff, S. 17

Verhindern von Man-in-the-Middle-Angriffen:

- symmetrische Authentifikation (s. IV.6) mit einem Langzeitgeheimnis
- digitale Signaturen (s. VII)

[1] aus einem Chiffrat darf sich kein anderes gültiges berechnen lassen
[2] hier muss der diskrete Logarithmus schwierig zu berechnen sein!

VI. Public-Key-Kryptographie

Ein bisschen zu den mathematischen Grundlagen findet sich im Kapitel **??**.

VI.1. Definition

FIXME: Definition

VI.2. Sicherheitsbegriff: IND-CCA2-Sicherheit

FIXME: Definition

VI.3. Beispiel: Elgamal

Seien G eine Gruppe mit Erzeuger g, a zufällig.

- öffentlicher Schlüssel von Alice: g^a
- geheimer Schlüssel von Alice: a
- Verschlüsseln einer Nachricht m:
  - wähle b zufällig und berechne g^b
  - verschicke $(g^b, g^{ab} \oplus m)$[1]
- Entschlüsseln eines Chiffrats c: mithilfe von a

VI.4. Beispiel: RSA

Sei $N \in \mathbb{N}$ mit $N = p \cdot q$ für p, q prim. Wähle ein zu $\varphi(N) = (p-1)(q-1)$ teilerfremdes e mit $1 < e < \varphi(N)$ und berechne $d := e^{-1} \mod \varphi(N)$.

- öffentlicher Schlüssel: (e, N)
- geheimer Schlüssel: (d, N)
- Verschlüsseln einer Nachricht m: $c = m^e \mod N$
- Entschlüsseln eines Chiffrats c: $m = c^d \mod N$

VI.4.1. Die RSA-Funktion

Die Funktion $x \mapsto x^e \mod N$ wird auch als RSA-Funktion bezeichnet. Sie ist eine Permutation auf $(\mathbb{Z}/N\mathbb{Z})^\times$.

[1] oder $(g^b, g^{ab} \cdot m)$

VI.4.2. Textbook-RSA

Das oben beschriebene Verfahren wird oft auch als Textbook-RSA bezeichnet. Es ist nicht sicher, da gleiche Klartexte immer auf gleiche Chiffrate abgebildet werden.

Beispiel Auktionsangriff

FIXME: Bild Auktionsangriff, S. 20

VI.4.3. RSA-ES-OAEP

Hier kommt bei der Berechnung des Chiffrats eine Zufallszahl r ins Spiel: $c = ((m+h(r))\|(h(m+h(r))+r))^e$. Zur Entschlüsselung bilde zunächst c^d. Dann hashe den ersten Teil der Nachricht, um durch Addieren des Hashs zum zweiten Teil der Nachricht den Zufall zu bekommen. Nun kann die eigentliche Nachricht berechnet werden.

Sicherheit

RSA-ES-OAEP ist beweisbar sicher im Random Oracle Model: Ein Angreifer, der das IND-CCA2-Spiel gewinnt, kann benutzt werden, um die RSA-Funktion zu invertieren.

VII. Digitale Signaturen

VII.1. Begriffe

- Authentizität (d.h. einer Person eindeutig zugeordnet)
- Integrität (d.h. unverändert)
- Unabstreitbarkeit (non repudiation)
- Praktikabilität (d.h. kurz im Vergleich zum Dokument)

VII.2. Beispiel: Signieren mit RSA (anschaulich)

Man „entschlüsselt“ das Dokument, als wäre es ein Chiffrat. Dies kann nur der Besitzer des Secret Keys. Prüfen kann aber jeder, der den Private Key kennt: „Verschlüsseln“ der Signatur ergibt die Nachricht.
Signatur zu m wäre dann $m^d \mod N = \sigma$ und überprüfen via $\sigma^e \mod N \stackrel{?}{=} m$.

So ist das aber noch nicht sicher:

1. zu zufälligem r wirkt r wie eine gültige Signatur von $r^e \mod N$
2. $\sigma_1 \cdot \sigma_2 \mod N$ ist eine gültige Signatur, da $(m_1^d) \cdot (m_2^d) = (m_1 \cdot m_2)^d \mod N$

Abhilfe: Hash-then-Sign (beweisbar sicher im Random Oracle Model)

VII.3. Definition Signatur

FIXME: Definition Signatur, S. 25

VII.4. Sicherheitsbegriff: EUF-CMA

VII.5. Beispiel: Elgamal-Signaturen

Sei G eine Gruppe mit Erzeuger g ($G = \mathbb{F}_p$ für p prim)[1]. Wähle x zufällig.

- öffentlicher Schlüssel (Verifikationsschlüssel): $vk = g^x$
- signing key: $sk = x$

Signatur „naiv“ :

- signieren: $M \equiv ax \mod (p-1) \rightarrow$ Signatur: a

[1] Rechnen: bei Gruppenelementen $\mod p$, im Exponenten $\mod (p-1)$

- prüfen: $g^M \equiv g^{ax} \equiv vk^a \mod (p-1)$
- aber: $x \equiv Ma^{-1} \mod (p-1)$ und jeder kann nun den sk ausrechnen

deshalb:

- signer Bob wählt k zufällig mit $ggT(k, p-1) = 1$
- $a :\equiv g^k \mod p$
- berechne b mit $m \equiv x \cdot a + k \cdot b \mod (p-1)$
- Signatur zu m ist (a, b)
- Prüfen der Signatur: $g^m \equiv g^{xa+kb} \equiv g^{xa} \cdot g^{kb} \equiv vk^a \cdot a^b \mod p$

VII.5.1. Probleme

1. niemals ein k zweimal verwenden → mit $m_1 = xa + kb_1$ und $m_2 = xa + kb_2$ lässt sich x berechnen
2. es ist möglich, gültige Signaturen zu (unsinnigen) Nachrichten zu generieren → Lösung: hash-then-sign

VII.6. Beispiel: DSA (Digital Signature Algorithm)

$g \in \mathbb{F}_p^\times$ erzeuge eine multiplikative Gruppe der Ordnung $q \in \mathbb{P}$, $(q|p-1)$.

- $sk = x$
- $vk = g^x$

Signieren:

- wähle $k \in \{0, dotsc, q-1\}$ zufällig
- berechne $r :\equiv g^k(\mod p) \mod q$
- berechne s mit $h(m) \equiv k \cdot s - x \cdot r \mod q$
- dann ist (r, s) eine Signatur zu m

Prüfen: $r \equiv (g^{s^{-1}h(m)} vk^{s^{-1}} r \mod p) \mod q$

VII.7. Beispiel: One-Time-Signaturen (aus Hashfunktionen)

sk besteht aus zufälligen Strings $\in \{0,1\}^k$

$$r_1^0\ r_2^0\ r_3^0\ \dots\ r_k^0$$
$$r_1^1\ r_2^1\ r_3^1\ \dots\ r_k^1$$

vk ist

$$h(r_1^0)\ h(r_2^0)\ h(r_3^0)\ \dots\ h(r_k^0)$$

$$h(r_1^1)\ h(r_2^1)\ h(r_3^1)\ \ldots\ h(r_k^1)$$

Beispielsignatur zu $01\ldots1$: $r_1^0\ r_2^1\ \ldots\ r_k^1$

One time signatures darf man nur einmal verwenden! Alternativ kann man einen doppelt so langen vk benutzen: Signiere die Nachricht und einen neuen public key (der wieder zwei Nachrichten signieren kann). Zum Verifizieren benötigt man eine Kette signierter vks, die zum ursprünglichen public key führen. Wenn die Nachrichten länger als k bit sind, verwendet man hash-then-sign.

VII.8. Ist EUF-CMA genug?

VII.8.1. Key Substitution Attacks

- Alice hat vk und signiert m mit σ
- Bob wählt (böse) einen vk', sodass seine Signatur zu m exakt σ ist

Beispiel (RSA)

1. Wähle $\bar{p}$ und $\bar{q}$ so, dass $\bar{p}-1$ und $\bar{q}-1$ in kleine Primfaktoren zerfallen und σ und $h(m)$ $\mathbb{F}_{\bar{p}}^{\times}$ und $\mathbb{F}_{\bar{q}}^{\times}$ generieren.
2. Pohlig-Hellman-Algorithmus löst den dlog $\mod \bar{p}$ und $\mod \bar{q}$.
3. Berechne x_1, x_2, sodass $\sigma^{x^1} \equiv h(m) \mod \bar{p}$ und $\sigma^{x^2} \equiv h(m) \mod \bar{q}$.
4. Ist $ggT(\bar{p}-1, \bar{q}-1) = 2$, so gibt es ein eindeutiges $\bar{e} < \varphi(\bar{p}\bar{q})$ mit $\bar{e} \equiv x_1 \mod \bar{p}-1$ und $\bar{e} \equiv x_2 \mod \bar{q}-1$ (Chinesischer Restsatz).
5. Gib $(\bar{n} = \bar{p} \cdot \bar{q}, \bar{e})$ als vk aus.

Dann gilt: $\sigma^{\bar{e}} \equiv h(m) \mod \bar{n}$, d.h. Signatur gilt, weil $\sigma^{\bar{e}} \equiv h(m) \mod \bar{p}$ und $\sigma^{\bar{e}} \equiv h(m) \mod \bar{q}$.

Lösungen

- DSA ist sicher gehen (starke) Key Substitution
- vk immer mitsignieren

VII.8.2. Subliminal Channel

Gut Simmons hat gemerkt, dass Signaturen Nachrichten enthalten können (subliminal channels).

Beispiel:

RSA-PSS signiert eine spezielle Kodierung:

FIXME: Bild RSA-PSS, S. 29

VIII. Key Management

oder Wie benutzt man Public-Key-Kryptographie?

VIII.1. PKI (Public-Key-Infrastruktur)

VIII.1.1. „Definition“

Ein digitales Zertifikat ordnet einem öffentlichen Schlüssel eindeutig einen Inhaber zu.
Eine Public-Key-Infrastruktur ist die Gesamtheit der organisatorischen Maßnahmen, die für eine vertrauenswürdige Ausgabe und Verifikation von Zertifikaten notwendig ist.

Aufgaben einer PKI sind z.B. die Ausstellung, Verteilung, Prüfung und der Widerruf digitaler Zertifikate. (Impliziert dies, dass der Inhaber seinen eigenen secret key kennt?)

VIII.2. Beispiel: X.509-Zertifikat

(die folgende Liste ist nicht vollständig)

1. Versionsnummer $\in \{1, 2, 3\}$, je nachdem, welche erweiterten Elemente vorhanden sind
2. Seriennummer, wird von Certification Authority (CA) gewählt und sollte pro CA eindeutig sein
3. Signaturalgorithmus und Parameter
4. Distinguished Name (DN) des Ausstellers
5. Gültigkeitsdauer
6. Distinguished Name des Inhabers
7. Public-Key-Informationen: Algorithmus, Parameter, public key
8. Erweiterungen
9. Signatur auf 1.-8.

VIII.3. Certificate Revocation

mittels Certificate Revocation Lists (CRL) (Listen einer CA, die widerrufene Zertifikate enthalten)

Diese enthalten ein Ausstellungsdatum und das Datum der nächsten geplanten CRL. Zertifikate bleiben auf der CRL, bis sie abgelaufen sind. (Genauer: Online Certificate Status Protocols)

VIII.4. Web of Trust

$\widehat{=}$ sozialem Netzwerk von „Mini-CAs“.

- Key-Signing-Partys zum Signieren
- Jeder kann selbst festlegen, welchen anderen Usern er vertraut.

Im Einzelfall unsicherer (schlecht geschützte CAs), aber Angriffe skalieren nicht so.

VIII.5. TLS (Transport Layer Security)

- Standard im Internet
- Nachfolger von SSL-Protokollen

VIII.5.1. Ablauf

FIXME: Bild TLS, S. 31
Jeder kann eine key-renegotiation beantragen und das Protokoll startet dann neu.

VIII.5.2. Besonderheiten

- Schutz gegen Downgrade (kein Versionswechsel während Protokollablauf)
- die Nachricht „Finish“ enthält einen Hash über alle bisherigen Protokollnachrichten der Session
- die verwendete pseudozufällige Funktion teilt den Input in zwei Hälften, hasht mit MD5 und SHA1 und XORs die Ergebnisse $\rightarrow$ das bleibt sicher, auch wenn beispielsweise MD5 Schwächen zeigt („Robust Combiner“)

VIII.6. Key Renegotiation Attack

auf TLS

VIII.6.1. Ziel

Client und Server sollen sich in unterschiedlichen Anwendungskontexten befinden, was der Angreifer ausnutzen kann (z.B. Client will Passwort senden, Server will nächste Nachricht an Adresse XY weiterleiten)

VIII.6.2. Ablauf

FIXME: Bild Key Renegotiation Attack, S. 32

IX. Netzwerksicherheit

IX.1. CIA-Paradigma

- Confidentiality
- Integrity
- Availability

(das ist noch keine vollständige Spezifikation, aber ein „Template“)

IX.2. Sicherheitsbegriff

- Vertraulichkeit
- Integrität
- Data origin authentification
- Peer entity authentification
- Non repudiation (Nichtabstreitbarkeit)
  - proof of origin
  - proof of delivery/submission
  - proof of receipt

IX.3. Das ISO/OSI-Referenzmodell

7 Anwendungsschicht
6 Präsentationsschicht
5 Sitzungsschicht
4 Transportschicht
3 Netzwerkschicht
2 Verbindungsschicht
1 Physikalische Schicht

Sicherheitseigenschaften sollen/können durch darunterliegende Schichten nicht gefährdet werden. (Ausnahme: Anonymität)

IX.4. IPsec

Protokollsuite, die Authentifikation und Verschlüsselung von IP-Paketen sowie entity authentification und key exchange erlaubt (bringt Sicherheit schon auf Ebene 3).

Schneier, Ferguson: „IPsec was a great disappointment to us.“

Im Wesentlichen: zu komplex.

IX.5. Bedrohungen für Rechner in Netzwerken

(Liste nicht vollständig)

- Belauschen, Unterdrücken, Verfälschen von Daten
- Portscan (automatisches Scannen von Schwachstellen)
- Fingerprinting (z.B. Ermitteln der OS-Version)
- Angriffe auf das Routing
- Schwachstellen in anderen Protokollen (z.B. DNS)
- Denial-of-Service-Attacken (DoS)
- Angriffe von innen
- Viren, Würmer, Trojaner
- Backdoors

IX.6. Schutzmaßnahmen

IX.6.1. Firewalls

- Paketfilter (Aussortieren nach Adressen/Diensten)
- Stateful Inspection (berücksichtigt zusätzliche Verbindungsinformationen)

FIXME: Bilder Firewall, S. 34

Firewalls sind eine Präventivmaßnahme, die ergänzt werden sollten durch Monitoring.

IX.6.2. Monitoring

Intrusion Detection Systems

IX.6.3. Honeypots

IX.6.4. Datendiode

Datenversand nur in eine Richtung möglich

FIXME: Bild Datendiode, S. 35

X. Zugriffskontrolle

(wirkt ein bisschen altmodisch, heute usage control)

X.1. Bell-LaPadula-Modell

Das Bell-LaPadula-Modell ist ein statisches Zugriffskontrollmodell. Oberstes Schutzziel: Confidentiality

X.1.1. Definition

- Subjektmenge $\mathcal{S}$
- Objektmenge $\mathcal{O}$
- Menge von Zugriffsoperationen $\mathcal{A} = \{read, write, append, execute\}$
- Menge $\mathcal{L}$ von Security Levels mit einer partiellen Ordnung $\leq$

Dabei implizieren *write*-Rechte die *read*-Rechte. (Beispiel: unclassified $\leq$ confidential $\leq$ secret $\leq$ top secret)

Im Bell-LaPadula-Modell ist der Systemzustand ein Element aus $\mathcal{B} \times \mathcal{M} \times \mathcal{F}$, wobei:

- $\mathcal{B} = \mathcal{P}(\mathcal{S} \times \mathcal{O} \times \mathcal{A})$ aktuelle Zugriffe beschreibt (wer hat Zugriff auf was mit welcher Zugriffsart)
- $\mathcal{M}$ die Menge der Zugriffskontrollmatrizen (ACM) bezüglich der Subjekte in $\mathcal{S}$ und der Objekte in $\mathcal{O}$ ist. Die Elemente von $\mathcal{M}$ haben die Form $M = (M_{so})_{s \in \mathcal{S}, o \in \mathcal{O}}$ mit $M_{so} \subseteq \mathcal{A}$ für alle $s \in \mathcal{S}, o \in \mathcal{O}$.
- $\mathcal{F}$ Dreitupel aus Funktionen enthält, den sog. Security Level Assignments. Hier gilt $\mathcal{F} \subseteq \mathcal{L}^{\mathcal{S}} \times \mathcal{L}^{\mathcal{S}} \times \mathcal{L}^{\mathcal{O}}$. Ein Dreitupel (f_s, f_c, f_o) hat dabei folgende Form und Bedeutung:
  - $f_s \colon \mathcal{S} \to \mathcal{L}$ gibt für jedes $s' \in \mathcal{S}$ den maximalen Sicherheitslevel an
  - $f_c \colon \mathcal{S} \to \mathcal{L}$ gibt den gegenwärtigen (current) Sicherheitslevel an
  - $f_o \colon \mathcal{O} \to \mathcal{L}$ gibt für jedes $o' \in \mathcal{O}$ den Sicherheitslevel an

  Dabei gilt: f_c muss von f_s dominiert werden: $\forall s' \in \mathcal{S} \colon f_c(s') \leq f_s(s')$.

Ein Systemzustand heißt sicher, wenn er die folgenden drei Eigenschaften erfüllt:

- Simple Security Property (ss-Eigenschaft):
  ein Zustand (b, M, f) genügt der ss-Eigenschaft, falls $\forall (s', o', a') \in b$ mit $a \in \{read, write\}$ gilt: $f_s(s') \geq f_o(o')$ („no read up“)

- Star Property (*-Eigenschaft):
  ein Zustand (b,M,f) erfüllt die *-Eigenschaft, falls $\forall(s', o', a') \in b$ mit $a \in \{append, write\}$ gilt: $f_c(s') \leq f_o(o')$ („no write down“)
  Weiterhin, falls ein $(s', o', a) \in b$ mit $a \in \{append, write\}$ und ein $(\hat{s}, \hat{o}, \hat{a}) \in b$ mit $s' = \hat{s}$ und $\hat{a} \in \{read, write\}$ existiert, dann muss $f_o(\hat{o}) \leq f_o(o')$ gelten. („Kein Nachrichtenfluss von high level object zu low level object.“)

- Discretionary Security Property (ds-Eigenschaft):
  ein Zustand (b, M, f) erfüllt die ds-Eigenschaft, falls $\forall(s', o', a) \in b$ stehts $a \in M_{s'o'}$ gilt

X.1.2. Basic Security Theorem

Werden ausgehend von einem sicheren Initialzustand nur sichere Übergänge durchgeführt, so erhält man einen sicheren Systemzustand.

X.1.3. Nachteile

Leider sammelt sich im Bell-LaPadula-Modell Information „oben“, ein Deklassifizieren ist nicht möglich. In der Praxis werden etwa „trusted subjects“ eingeführt, um ein Deklassifizieren von Daten zu erlauben.

Weitere Nachteile:

- Integrität der Daten wird nicht mitbetrachtet (niedrigstufige user/Prozesse können evtl. höher eingestufte Objekte verändern)
- keine Forderung an die ACM, etwa darf allen $s \in \mathcal{S}$ alle Rechte gegeben werden
- verdeckte Kanäle, z.B. die (Nicht)Existenz von Dateien, bleiben unberücksichtigt

X.1.4. Vorteile

- handhabbar
- formal (d.h. für Beweise geeignet)

X.2. Chinese-Wall-Modell

Zugriffsrechte hängen von der Vergangenheit ab → z.B. kein Informationsfluss zwischen konkurrierenden Firmen (z.B. bei Unternehmensberatung)

X.2.1. Definition

- Menge $\mathcal{C}$ von Firmen
- Objektmenge $\mathcal{O}$ (jedes Objekt gehört einer Firma)
- Subjektmenge $\mathcal{S}$ (die Berater)
- Funktion $y\colon \mathcal{O} \to \mathcal{C}$, welche jedem Objekt die zugehörige Firma zuordnet
- Funktion $x\colon \mathcal{O} \to \mathcal{P}(\mathcal{C})$, welche jedem Objekt eine conflict-of-interest-Klasse zuordnet

Die Sicherheitsmarke (security label) eines Objekts $o \in \mathcal{O}$ ist das Tupel $(x(o), y(o))$. Im Falle $x(o) = \emptyset$ spricht man von „sanitized information“.

Eine Matrix M enthält Informationen über Zugriffe $M = (M_{so})_{s \in \mathcal{S}, o \in \mathcal{O}}$ mit

$$M_{so} = \begin{cases} true, & \text{falls } s \text{ Zugriff auf } o \text{ hatte,} \\ false, & \text{sonst.} \end{cases}$$

Der Initialzustand ist $M = (false)_{s \in \mathcal{S}, o \in \mathcal{O}}$.

X.2.2. Eigenschaften

- Simple Security Property (ss-Eigenschaft):
  $s \in \mathcal{S}$ erhält Zugriff auf $o \in \mathcal{O}$ nur, falls $\forall o' \in \mathcal{O}$ mit $M_{so'} = true$ gilt: $y(o) = y(o')$ oder $y(o) \notin x(o')$.
- Star Property (*-Eigenschaft):
  ein $s \in \mathcal{S}$ erhält Schreibzugriff auf ein Objekt $o \in \mathcal{O}$ nur, falls s aktuell keinen Lesezugriff auf $o' \in \mathcal{O}$ hat mit $y(o) \neq y(o')$ oder $x(o') = \emptyset$. (Die *-Eigenschaft verhindert die Weitergabe von Daten über Dritte.)

XI. Zero Knowledge

Idee: Wir wollen (interaktiv) beweisen, dass wir ein Geheimnis kennen, ohne das Geheimnis selbst zu verraten.

XI.1. Eigenschaften:

- Korrektheit: Ein Beweiser, der das Geheimnis nicht kennt, soll einen Prüfer nur mit vernachlässigbarer Wahrscheinlichkeit davon überzeugen können, es zu kennen.
- Zero Knowledge: Für jeden Prüfer gibt es einen Simulator, der einen Mitschrieb der Beweiskommunikation selbst erzeugen kann, ohne den Beweis zu kennen.

XI.2. Beispiel: Graph-Isomorphismus

XI.2.1. Ablauf

- allen bekannt: zwei Graphen G_1 und G_2
- Geheimnis des Beweisers: ein Graph-Isomorphismus zwischen G_1 und G_2
- der Beweiser erzeugt einen weiteren zufälligen Graphen, der isomorph zu G_1 und G_2 ist, und gibt ihn dem Prüfer
- der Prüfer fordert den Isomorphismus vom neuen Graphen zu einem der beiden Graphen G_i
- der Beweiser gibt den Isomorphismus bekannt
- das Spiel wird häufig wiederholt

XI.2.2. Eigenschaften

- kennt der Beweiser das Geheimnis, so kann er dem Prüfer immer den geforderten Isomorphismus angeben
- Korrektheit: kennt er ihn nicht, kann er nur einen der beiden Isomorphismen angeben (den, mit dem er den neuen Graphen erzeugt hat), und wird bei häufigem Wiederholen erwischt
- Zero Knowledge: weiß der Beweiser vorher, welchen Isomorphismus der Prüfer sehen will, so bereitet er genau diesen vor

XI.3. Beispiel: Graph-3-Färbbarkeit

Mit Graph-3-Färbbarkeit wird die Frage bezeichnet, ob zu einem gegebenen Graphen eine Knotenfärbung mit 3 Farben existiert, sodass direkt verbundene Knoten immer ungleiche Farben haben. Graph-3-Färbbarkeit ist ein NP-vollständiges Problem.

XI.3.1. Ablauf

- allen bekannt: ein Graph G
- Geheimnis des Beweisers: eine 3-Färbung von G
- der Beweiser benennt die Farben zufällig um (dabei bleibt die Färbung eine 3-Färbung)
- der Beweiser verschlüsselt alle Knoten und zeigt dies dem Prüfer
- der Prüfer wählt eine Kante
- der Beweiser entschlüsselt die zugehörigen Knoten
- dieses Spiel wird häufig wiederholt (mit immer neuer Vertauschung der drei Farben, damit der Prüfer das Geheimnis nicht lernen kann)

XI.3.2. Eigenschaften

- Korrektheit: gibt es keine 3-Färbung, so sind immer irgendwo zwei Farben gleich und der Beweiser wird bei häufigem Wiederholen erwischt
- Zero Knowledge: weiß der Beweiser vorher, welche Kante gefragt wird, verschlüsselt er dort zwei zufällige, verschiedene Farben

XII. Authentifikation

XII.1. Definition

Authentifizierung bindet eine Identität an ein Subjekt.

XII.2. Ansätze

- Entität weiß etwas (Kennwort)
- Entität ist etwas (Biometrie)
- Entität hat etwas (Chipkarte)
- Entität kann etwas (Captcha)
- Entität befindet sich an einem bestimmten Ort

XII.3. Komponenten

- Menge A der Authentifizierungsinformation
  (Information, mit der Identität bewiesen wird)
- Menge C der Komplementärinformationen
  (was das System speichert, um Authentifizierung zu validieren)
- Menge $\mathcal{F} \in C^A$ der Komplementierungsfunktionen
  (leitet aus gegebenem $a \in A$ das entsprechende $c \in C$ ab)
- Menge $L \subseteq \{true, false\}^{A \times C}$ der Authentifikationsfunktionen
  (verifiziert Identifikation)
- Menge S der Auswahlfunktionen
  (zum Anlegen, Ändern, Entfernen von Entitäten und entsprechenden Daten)

XII.4. Typische Anwendung: Kennworte

- 1. Ansatz: System speichert Kennworte explizit
  $\rightarrow$ Problem: Diebstahl des Passwordfiles
- 2. Ansatz: kryptographische Hashwerte der Kennworte speichern
  $\rightarrow$ Problem: Offline-Wörterbuchattacke sehr effizient
- 3. Ansatz: Saltung (pro Benutzer andere Hashfunktion): $H_s(pw) := H(pw \| s)$
- 4. Ansatz: „Remote-Login“ mit dediziertem Authentifizierungsserver

XII.5. Maßnahmen gegen Offline-Attacken

XII.5.1. Wahl guter Kennworte

- vorgegebene Zufallsstrings (werden aufgeschrieben und am Rechner deponiert)
- „Key-Crunching“ → Hashing langer Passphrases
- Verschleierung aufgeschriebener Kennworte (einfache Transformation)
- proaktive Kennwortwahl
- zeitliche Variation (ganz gut: ab und zu Kennwort verlängern)
- Security Awareness

XII.6. Maßnahmen gegen Online-Attacken

- Backoff → nach n Fehleingaben Sperrung für x_n Sekunden
- Disconnection → nach n Fehleingaben Verbindungstrennung
- Jailing → begrenzter Zugriff wird trotz Fehleingabe gewährt, oft mit Honeypots kombiniert

XII.7. Beispiel: CAPTCHAs

automatisch generierte Rätsel, die Maschinen nur sehr schwer lösen können, Menschen dagegen sehr leicht

XII.8. Raffiniertere Verfahren (Challenge-Response)

XII.8.1. Schema

Benutzer hat Geheimnis s

FIXME: Bild Schema, S. 53

Das Geheimnis S soll nicht aus c und $c(r, s)$ rekonstruierbar ein (selbst bei böswillig gewähltem c).

XII.8.2. Beispiele

RSA-Signaturen

Server schickt String, lässt ihn sich signieren → in der Praxis manchmal zu aufwändig

mittels Hashfunktion oder Verschlüsselung

$r(s.c) = h(s, c)$ oder $r(s, c) = Enc_s(c)$ → Server muss Geheimnis S kennen

Zero Knowledge

→ in der Praxis zu aufwändig

SPEKE (Simple Password Encrypted Key Exchange)

Parameter:

- $p = 2q + 1$, $p, q \in \mathbb{P}$ („safe prime“)
- Hashfunktion H
- $g = H(Passwort)^2 \mod p$ (erzeugt die Gruppe der quadratischen Reste $\mod p$)

Ablauf: wie Diffie-Hellman, aber $key := H(g^{ab})$ sowie mit Key Confirmation

FIXME: Bild Ablauf, S. 53

möglicher Angriff: schicke $g^a = 1$ oder g^a mit kleiner Ordnung → Schlüssel unabhängig von Passwort → Lösung: Protokollabbruch, falls $Ord(g^{ab} < q$

XIII. Seitenkanalangriffe

FIXME: schreib mich ;)

XIV. Implementierungsfehler

XIV.1. in Programmen

- Stack/Heap Overflows
- Integer Overflows, Signed-Unsigned-Bugs
- falsche Unicodebehandlung
- Ausnutzen von Race Conditions
- „Environment Creep“ (es war unter anderen Umständen sicher, aber die Umstände haben sich geändert)
- Ändern temporärer Dateien

XIV.2. in Webanwendungen

- SQL injection
- XSS (Cross Site Scripting)
- GIFAR (Datei entspricht gif- und jar-Standard)
- Variablenvorbelegung bei PHP

XIV.3. Lektion fürs Leben

- Komplexität ist ein Feind der Sicherheit.
- Sicherheit komponiert nicht.

XV. Sicherheitsbewertung/Zertifizierung

Beurteilung durch eine vertrauenswürdige Instanz (bescheinigt Eigenschaften) → meist wird allerdings nicht ein Produkt, sondern der Entwicklungsprozess zertifiziert

XV.1. Gründe für eine Zertifizierung

- gesetzliche Bestimmungen (Datenschutz)
- Werbung
- überzeugt Nichtexperten
- günstigere Versicherung
- Vergleich von Produkten „einfacher“

XV.2. Common Criteria (ISO 15408)

Zertifizierungsstelle in Deutschland: BSI (Bundesamt für Sicherheit in der Informationstechnik)

- ToE (Target of Evaluation)
- Protection Profile → Forderungen an das ToE
  - Descriptive Elements:
    - worum geht es? (Smartcard, Firewall, ...)
    - was ist das Problem, das damit gelöst werden soll?
  - Rationale (Zuordnung zwischen Bedrohungen/Angriffen und Sicherheitseigenschaften):
    - welche Betriebsumgebung?/welches Einsatzszenario?
    - welche Bedrohungen/Angriffe?
    - welche Sicherheitseigenschaften?
  - Functional Requirements: funktionale Spezifikation des ToE
  - Evaluation Assurance Requirements:
    - wie/was wird evaluiert? (es gibt umfangreiche Beispielkataloge)
    - wie intensiv wird evaluiert?

XV.2.1. Evaluation Insurance Levels

numerische Bewertung der Prüfstrenge

- EAL 1: funktional getestet
- EAL 2: strukturell getestet

- EAL 3: methodisch getestet und überprüft
- EAL 4: methodisch entwickelt, getestet und durchgesehen
- EAL 5: semiformal entworfen und getestet
- EAL 6: semiformal verifizierter Entwurf und getestet
- EAL 7: formal verifizierter Entwurf und getestet

XVI. Data Base Privacy

XVI.1. k-Anonymität

FIXME: Bild k-Anonymität, S. 57

für jede Kombination der QI in der Datenbank gibt es k Zeilen mit dieser Kombination (z.B. durch Vergröbern der QIs)

XVI.1.1. Kritik

1. nur das Ergebnis wird beurteilt, nicht der Prozess (Seitenkanäle)
2. komponiert nicht
3. Homogenitätsattacke

XVI.2. Differential Privacy

Datenbanken sind Mengen von Tupeln aus $\mathbb{R}^d$. Ein Release Mechanism nimmer eine Datenbank und liefert eine Zahl aus $\mathbb{R}$. Zwei Datenbanken haben „Hammingabstand“ 1, wenn sie sich nur in einem Tupel unterscheiden. Für zwei beliebige Datenbanken (aus einem gegebenen Universum) mit Hammingabstand 1 soll der Release sich höchstens um ϵ unterscheiden: „ϵ-differentially private“.

Methode für Release: Verrauschen

XVII. Secure Function Evaluation

XVII.1. Beispiel: Datingproblem

Alice und Bob haben ein Rendezvous, vermittelt von Datingagentur. Wie entscheiden, ob man sich wieder trifft? (peinlich, falls nur einer will...)

XVII.1.1. Lösung: Secure AND

jeder gibt geheimes Bit ein und lernt nur das AND-Ergebnis

Realisierung

- Input Alice: Bit a
- Input Bob: Bit b

1. Alice generiert Signaturkey x und sendet Verifikationskey y an Bob
2. Alice generiert (e, d, N) und sendet (e, N) an Bob
3. Bob wählt s, berechnet $r_b := Enc(s)$, $r_{1-b} := N - b$ und sendet (r_0, r_1) an Alice
4. Alice sendet $c_0 = sign(0) + Dec(r_0)$ und $c_1 = sign(a + Dec(r_1))$ an Bob
5. Bob berechnet $sign(ab) = c_b - s$ und teilt Alice das Ergebnis mit

Wofür Signieren? → sonst kann Bob immer $b = 1$ wählen und Alice über das Ergebnis belügen (und a herausfinden).
Wenn Alice nur eine Signatur richtig berechnet, hängt es von b ab, ob das Protokoll regulär durchgeht. → Lösung: ZK-Beweis

Achtung: Signatur darf keinen Subliminal Channel haben

XVII.2. allgemeine Secure Function Evaluation

XVII.2.1. Baustein: „oblivious transfer“ (OT)

- Input Alice: Strings a_0, a_1
- Input Bob: Bit b
- Output Alice: nichts
- Output Bob: a_b
- Alice lernt nichts über b
- Bob lernt nichts über a_0 oder nichts über a_1

XVII.2.2. Realisierung

1. Alice generiert (e, d, N) und sendet (e, N) an Bob
2. Bob wählt s, berechnet $r_b = Enc(s)$, $r_{1-b} = N - b$ und sendet (r_0, r_1) an Alice
3. Alice sendet $c_0 = a_0 + Dec(r_0)$ und $c_1 = a_1 + Dec(r_1)$ an Bob
4. Bob berechnet $a_b = c_b - s$

XVII.2.3. Realisierung mit Funktion f

hier ist der Input von Bob ein String b

- Alice stellt Wertetabelle F für $f(a, \cdot)$ auf
- Für jedes Bit von b sendet Alice zwei Zufallswerte k_0, k_1 via OT an Bob; es bezeichne (k_0^i, k_1^i) das zu b_i gehörige Tupel. Bob lernt jeweils $k_{b_i}^i$.
- Alice verschlüsselt F zeilenweise; die zu b zugehörige Zeile von F ist verschlüsselt mit dem XOR über $k_{b_i}^i$.
- Alice sendet die verschlüsselten Zeilen an Bob.
- Bob kann genau die zu b gehörige Zeile entschlüsseln und lernt $f(a, b)$.

Probleme:

1. Wie sicherstellen, dass Alice die richtige Funktion f codiert? → ZK-Beweis
2. Wie sicherstellen, dass Bob über das Ergebnis nicht lügen kann? → Signaturen
3. Tabelle F ist exponentiell groß in $|b|$ → „Yao's Garbled Circuits“

XVII.2.4. Yao's Garbled Circuits

- Zerlege Schaltkreise von f in einzelne Gatter und wende obiges Protokoll auf jedes einzelne Gatter an.
- Bob darf keine Zwischenergebnisse lernen, deshalb werden Gatterergebnisse direkt durch entsprechende Schlüssel ersetzt, statt die Schlüssel via OT zu übertragen (natürlich nur für Gatter, deren Ergebnis nicht Teil der Ausgabe des Schaltkreises ist).
- Bob weiß nicht mehr, welche Zeile einer Tabelle für ein Gatter innerhalb des Schaltkreises er entschlüsseln muss; deshalb werden verschlüsselte Nullstrings an jede Tabellenzeile angehängt, Bob entschlüsselt immer die ganze Tabelle und verwirft „falsche“ Zeilen.
- Bob kann noch aus der Position der „richtigen“ Zeile einers Gatters die Eingangsbelegung erkennen; daher werden jeweils die Zeilen einer Tabelle zufällig permutiert.