Leibniz Universität Hannover

Grundlagen der IT-Sicherheit
VL 4: Kryptographie 4

Prof. Dr. Markus Dürmuth, Wintersemester 2023/24

Unsere heutigen Themen…

- Digitale Signaturen
- Signaturen und Authentifizierung
- Message Authentication Codes (MACs)
- Definition von Sicherheit in der Kryptographie
  - IND-CPA und IND-CCA2
- PRP und PRF
- Abschluss Kryptographie

Leibniz Universität Hannover

DIGITALE SIGNATUREN

Digitale Signaturen

Alice sendet eine Nachricht *m* an Bob und signiert diese digital.

Ziele der digitalen Signatur:

- Fälschungssicher: Es sollte für jeden außer Alice schwer sein, die Signatur zu erstellen
- Authentisch: Bob kann leicht nachprüfen, dass Alice die Nachricht signiert hat
- Non-repudiation: Alice kann nicht leugnen, dass sie die Signatur erstellt hat
- Fälschungssicher: Die Nachricht *m* kann nach der Übermittlung nicht mehr verändert werden

Digitale Signaturen Setup

- (*sk, pk*) := generateKeyPair (*keysize*)
  - Erzeuge ein Schlüsselpaar in der gewünschten Schlüsselgröße
  - *sk* muss geheim gehalten werden und wird verwendet um Nachrichten zu **signieren**.
  - *pk* wird veröffentlicht und wird verwendet um Signaturen zu **verifizieren**.
- s := sign(m, sk_A)
  - Alice signiert die Nachricht mit ihrem geheimen Schlüssel und der Nachricht als Eingabe
- v := verify(m, sig, pk_A)
  - Bob bekommt den öffentlichen Schlüssel von Alice.
  - Bob verifiziert die Signatur mit der Nachricht, dem öffentlichen Schlüssel von Alice und der Signatur als Eingabe.
  - *v* ist **true**, wenn es sich um eine valide Signatur handelt, und **false**, wenn nicht.

Naive RSA-Signaturen

Öffentlicher Schlüssel (*e, n*) und privater Schlüssel (*d, n*). Berechnung wie letzte VL.

$$\mathrm{d} = \mathrm{e}^{-1} mod\ \varphi(n)$$

Signatur mit privatem Schlüssel (*d, n*)

Signiere die Nachricht m und erzeuge die Signatur $\mathbf{s} = \boldsymbol{m^d}\ \mathbf{mod}\ \boldsymbol{n}$

Verifiziere mit öffentlichem Schlüssel (*e, n*)

Verifiziere die Signatur s mittels Prüfung ob $\boldsymbol{s^e}\ =?\ \boldsymbol{m}\ \ \mathbf{mod}\ \boldsymbol{n}$

$\rightarrow$ **"true"** ***or*** **"false"**

Naive RSA-Signaturen

Erinnerung:

Signatur: $s = m^d \ mod \ n$

Verifizierung: $s^e =? \ m \ mod \ n$

Problem: Leicht zu fälschen!

- Wähle beliebige Signatur s
- Berechne passende Nachricht $m = s^e \ mod \ n$

→ *Verbesserte RSA-Signaturen*

Verbesserte RSA-Signaturen

- Kombiniert Hashes mit asymmetrischer Kryptographie.

- Alice berechnet den Hash $\mathrm{h} = \mathbf{H}(\mathbf{m})$
  - Verwendung einer „guten“ kryptographischen Hashfunktion H!
- Alice erzeugt die Signatur $\boldsymbol{s} = \boldsymbol{h}^{\boldsymbol{d}} = \mathbf{H}(\boldsymbol{m})^{\boldsymbol{d}} \ \boldsymbol{mod}\ \boldsymbol{n}$ und sendet sie zusammen mit der Nachricht m an Bob.
- Bob berechnet den Prüfwert $\boldsymbol{g} = \ \boldsymbol{s}^{\boldsymbol{e}} \mathbf{mod}\ \boldsymbol{n}$ und verifiziert ob $\boldsymbol{g} = \boldsymbol{H}(\boldsymbol{m}) \ \mathbf{mod}\ \boldsymbol{n}$

Verbesserte RSA-Signaturen

Öffentlicher Schlüssel (e, n) und privater Schlüssel (d, n). Berechnung wie letzte VL.

$$\mathrm{d} = \mathrm{e}^{-1} mod\ \varphi(n)$$

Signatur mit privatem Schlüssel (d, n)

Signiere die Nachricht m und erzeuge die Signatur $\mathbf{s} = \boldsymbol{H(m)^d} \ \mathbf{mod}\ \boldsymbol{n}$

Verifiziere mit öffentlichem Schlüssel (e, n)

Verifiziere die Signatur s mittels Prüfung ob $\boldsymbol{s^e} =? \boldsymbol{H(m)} \ \mathbf{mod}\ \boldsymbol{n}$

$\rightarrow$ **"true"** ***or*** **"false"**

Verbesserte RSA-Signaturen

Erinnerung:

Signatur: $s = H(m)^d \ mod\ n$

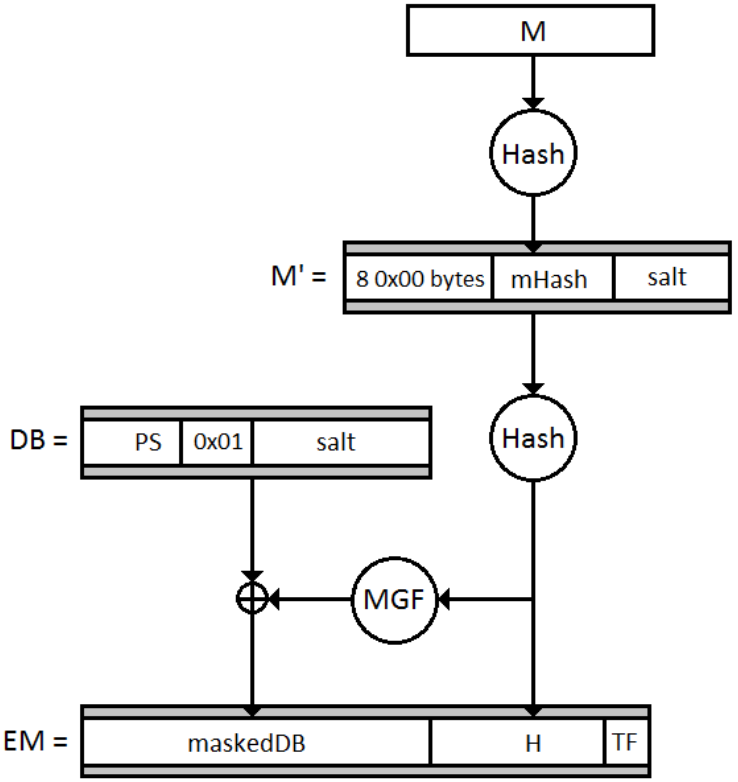
Verifizierung: $g = \ s^e \ =?\ H(m) = h \ mod\ n$

Schwieriger zu fälschen!

- Es ist möglich, eine Signatur s' zu wählen und ein passendes g' zu finden, doch damit hat man nur den Hash – und keine Nachricht
  - Erinnerung: Aus einem kryptographischen Hash kann ich nicht (oder nur sehr schwer) eine Nachricht erhalten („Einwegfunktion“)
- Varianten dieses Ansatzes werden als sicher angesehen. Annahmen:
  - RSA ist ausreichend schwer
  - Hashfunktion gleicht einem „Random Oracle“ (sehr starke Annahme)
  - Full-domain hash (FDH)
- Bonus: Lange Nachrichten werden „gratis“ bearbeitet

Real-world Beispiel: RSA-PSS

- Probabilistic Signature Scheme (Bellare, Rogaway, 1996)
- Beweisbar sicher im “Random Oracle Modell”
- Bessere “Sicherheitsreduktion” als Verfahren auf der letzten Folie
- PKCS #1 v2.1



SIGNATUREN UND AUTHENTIFIZIERUNG

Digitale Signaturen helfen uns, das Authentifizierungs-Problem zu lösen!

Das Authentifizierungsproblem

Problem: Asymmetrische Verschlüsselung benötigt den Austausch von öffentlichen Schlüsseln. Doch wie kann ich sicherstellen, dass ich den „richtigen“ öffentlichen Schlüssel für die „richtige“ Kommunikationspartei habe?

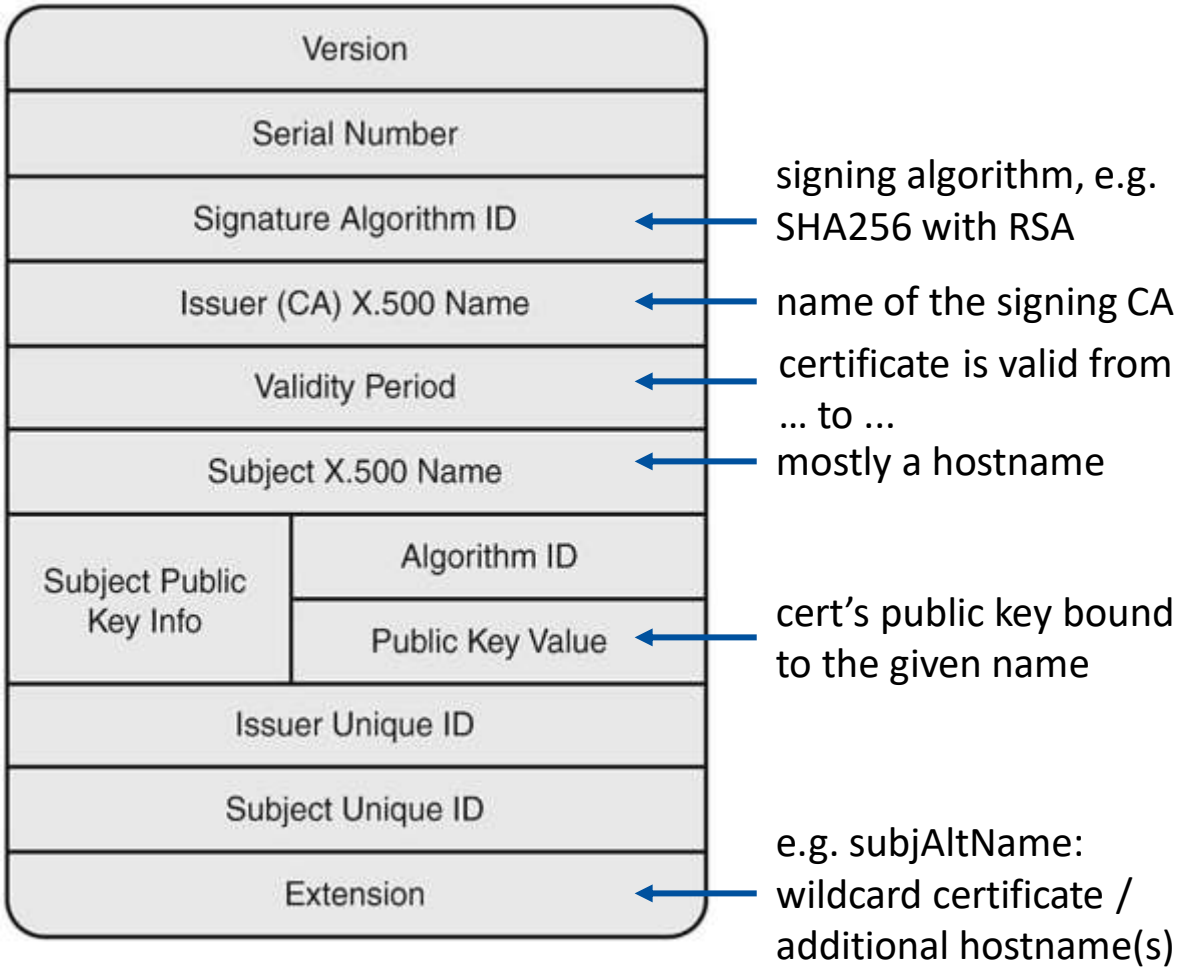
Lösung: Digitale Zertifikate:

„Ich bestätige: Dieser Schlüssel gehört zu diesem Subjekt“

Digitale Zertifikate und Authentifizierung

X.509 Digitales Zertifikat

Bestätigt die **Zugehörigkeit eines Öffentlichen Schlüssels** (z.B. RSA) **zu einem benannten Subjekt** (z.B. Hostname / Domain) sowie die erwartete Verwendung dieses Schlüssels.



Zertifizierungsstellen

Zertifizierungsstellen („Certificate Authorities“, CAs) validieren eine **Zertifikatsignierungsanforderung** („Certificate Signing Request“, CSR) und bestätigen dann die Bindung zwischen öffentlichem Schlüssel und Namensobjekt, indem sie es **digital signieren**.

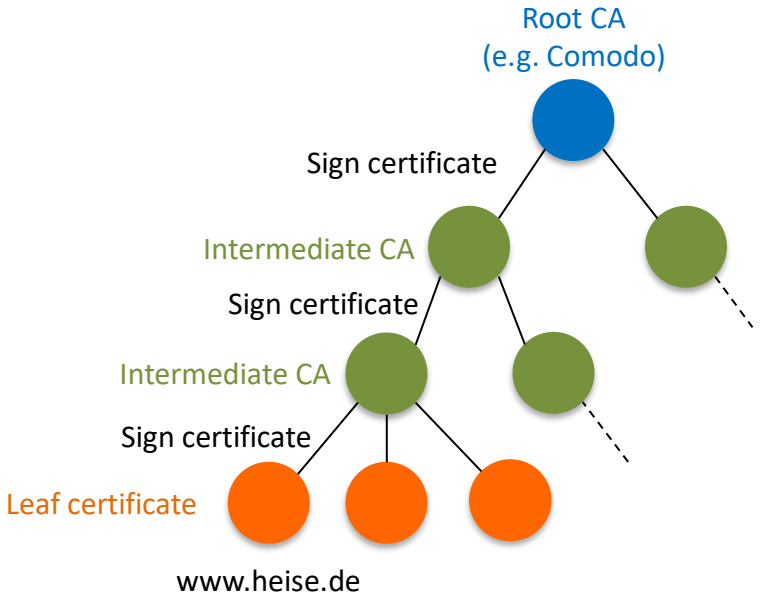
Verschiedene Arten der CSR-Validierung:

- Domänen-Validierung
- Organisations-Validierung
- Erweiterte Validierung

→ Verschiedene Prozeduren, ein CSR zu validieren

Zertifizierungsstellen

- CAs sind in einer hierarchischen, so genannten **Public Key Infrastructure** (PKI) organisiert.
- Ein Leaf-Zertifikat kann **von beliebigen (möglicherweise mehreren)** Root- oder Intermediate CAs signiert werden.



- Die meisten Systeme oder Browser haben CAs vorinstalliert

Root CA Zertifikate für Firefox

Mozilla Included CA Certificate List

As of July 19, 2017

Owner	Certificate Issuer Organization	Certificate Issuer Organizational Unit	Common Name or Certificate Name	Certificate Serial Number	SHA-256 Fingerprint
AC Camerfirma, S.A.	AC Camerfirma SA CIF A82743287	http://www.chambersign.org	Chambers of Commerce Root	00	0C:25:8A:12:A5:67:4A:EF:25:F2:8B:A7:DC:FA:E E:E6:3B:71:B3:61:60:6A:C3
AC Camerfirma, S.A.	AC Camerfirma S.A.		Chambers of Commerce Root - 2008	00a3da427ea4b1aeda	06:3E:4A:FA:C4:91:DF:D3:32:F3:08:9B:85:42:E A7:93:7E:E2:9D:96:93:C0
AC Camerfirma, S.A.	AC Camerfirma SA CIF A82743287	http://www.chambersign.org	Global Chambersign Root	00	EF:3C:B4:17:FC:8E:BF:6F:97:87:6C:9E:4E:CE:3 :8B:7D:11:C0:B2:29:8C:ED
AC Camerfirma, S.A.	AC Camerfirma S.A.		Global Chambersign Root - 2008	00c9cdd3e9d57d23ce	13:63:35:43:93:34:A7:69:80:16:A0:D3:24:DE:72: D:74:78:16:EE:BE:BA:CA
Actalis	Actalis S.p.A./03358520967		Actalis Authentication Root CA	570a119742c4e3cc	55:92:60:84:EC:96:3A:64:B9:6E:2A:BE:01:CE:0 5:AF:C1:55:B3:7F:D7:60:66
Amazon	Amazon		Amazon Root CA 1	066c9fcf99bf8c0a39e2f0788a43e6 96365bca	8E:CD:E6:88:4F:3D:87:B1:12:5B:A3:1A:C3:FC:B F:E1:CB:97:C6:AE:98:19:6E
Amazon	Amazon		Amazon Root CA 2	066c9fd29635869f0a0fe58678f85b 26bb8a37	1B:A5:B2:AA:8C:65:40:1A:82:96:01:18:F8:0B:EC 19:C3:9C:01:1E:A4:6D:B4
Amazon	Amazon		Amazon Root CA 3	066c9fd5749736663f3b0b9ad9e89 e7603f24a	18:CE:6C:FE:7B:F1:4E:60:B2:E3:47:B8:DF:E8:6 5:69:F5:03:43:B4:6D:B3:A4
Amazon	Amazon		Amazon Root CA 4	066c9fd7c1bb104c2943e5717b7b2 cc81ac10e	E3:5D:28:41:9E:D0:20:25:CF:A6:90:38:CD:62:3 C2:2B:0B:FB:25:89:70:92
Amazon	Starfield Technologies, Inc.		Starfield Services Root Certificate Authority - G2	00	56:8D:69:05:A2:C8:87:08:A4:B3:02:51:90:ED:CF 29:0F:CB:2A:E6:3E:DA:B5
AS Sertifitseerimiskeskuse (SK)	AS Sertifitseerimiskeskus		EE Certification Centre Root CA	5480f9a073ed3f004cca89d8e371e 64a	3E:84:BA:43:42:90:85:16:E7:75:73:C0:99:2F:09: :CC:BA:8A:22:9B:8A:76

- Firefox hat 182 vorinstallierte root CA Zertifikate
- Firefox vertraut ihnen für alle Verbindungen

https://ccadb-public.secure.force.com/mozilla/IncludedCACertificateReport

MEHR DAZU IN DER VORLESUNG ZUR NETZWERKSICHERHEIT

MESSAGE AUTHENTICATION CODES (MACS)

Message Authentication Codes (MACs)

- MACs sind das symmetrische Analog zu Signaturen

- Bieten Data Integrity…
- … aber keine non-repudiation, da sowohl Sender als auch Empfänger den Schlüssel kennen (müssen)

MACs Setup

- *k* := generateKey (*keysize*)
  - Erzeuge Schlüssel (typischerweise uniform aus einer Schlüsselmenge)
  - Schüssel *k* muss geheim gehalten werden und darf nur den beiden Kommunikationspartnern bekannt sein.

- *mac* := MAC(*m, k*)
  - Alice berechnet den MAC mit dem Schlüssel und der Nachricht als Eingabe

- v := verify(*m, mac, k*)
  - Bob verifiziert die MAC mit der Nachricht, dem Schlüssel und dem MAC als Eingabe.
  - v ist **true**, wenn es sich um einen validen MAC handelt, und **false**, wenn nicht.

Hash-based MAC (HMAC)

- MAC basierend auf einer kryptographischen Hashfunktion *H*

  HMAC(*k, m*) = H(*k* ⊕ *opad* ‖ H(*k* ⊕ *ipad* ‖ *m*))

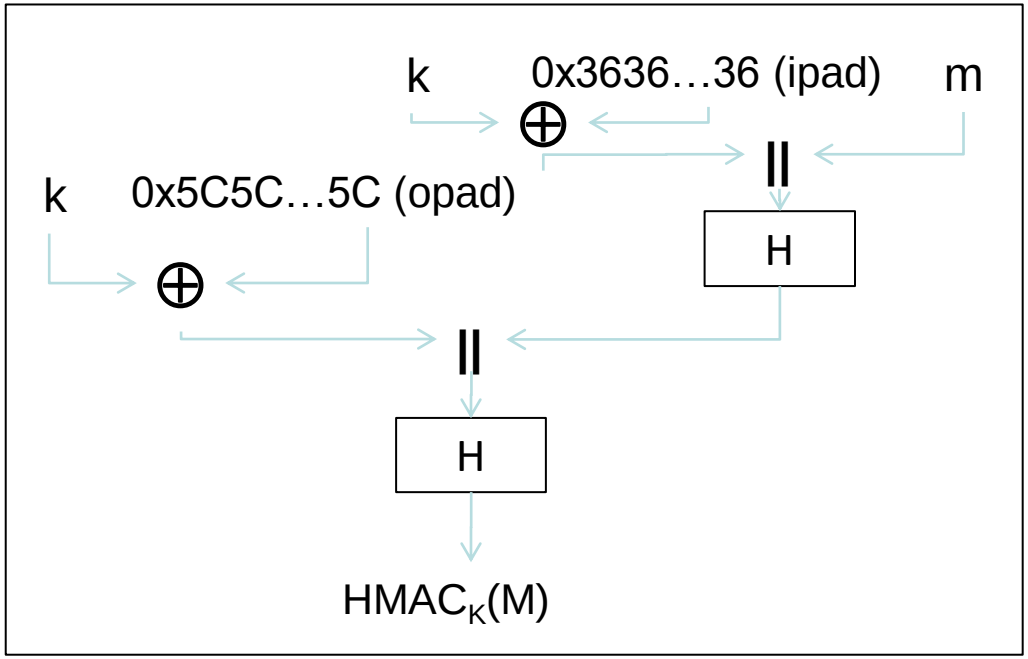
  wobei

  opad = 0x5c5c…5c

  ipad = 0x3636…36

- SHA256 -> HMAC-SHA256

- Warum so kompliziert?



Andere Konstruktionen

- Einfachere Konstruktionen sind nicht immer sicher:
  - für viele Hash Funktionen ist das “anhängen” von Daten leicht möglich (vgl. Merkle-Damgard Konstruktion (!))

- Würde $mac := H(k \| m)$ berechnet können an einen MAC Daten angehängt werden, ohne dass dafür der Schlüssel k benötigt wird (!)

- $H (k \| H(k \| m)$ vermutlich ebenfalls sicher

DEFINITION VON SICHERHEIT IN DER KRYPTOGRAPHIE

Sicherheit von Kryptosystemen

Leibniz Universität Hannover

Frage: Wann ist ein (asymmetrisches) Verschlüsselungsverfahren „sicher“?

Ansatz 1: „AngreiferIn kann die Nachricht *m* nicht erfahren, wenn er/sie den Chiffretext *c* hat.

Doch was ist, wenn der/die Angreifer/in ableiten kann …

- Die erste Häfte der Nachricht m
- Die zweite Hälfte der Nachricht m
- Die Parität der Nachricht?

Sicherheit von Kryptosystemen

Leibniz Universität Hannover

Frage: Wann ist ein (asymmetrisches) Verschlüsselungsverfahren „sicher“?

Ansatz 2: „AngreiferIn kann **keine Information** über die Nachricht m erhalten, wenn er/sie den Chiffretext c hat.

Das würde bedeuten …

- Keine Buchstabenhäufigkeit
- Keine Hälfte (oder auch nur ein winziger Bruchteil) der Nachricht m
- Keine Länge der Nachricht
- …

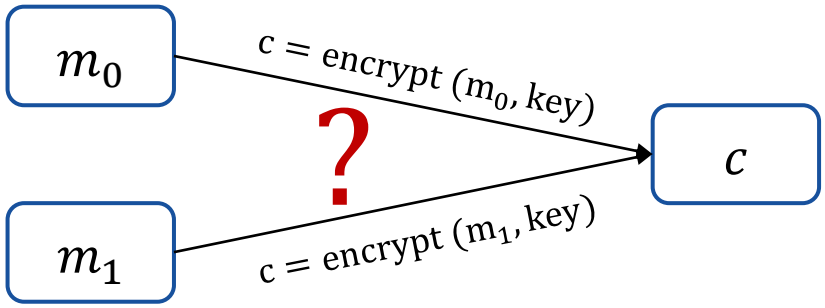
Diese Definition kann (außer mittels One-Time Pad) nicht realisiert warden

- (Und selbst beim OTP erfährt der Angreifer die Länge der Nachricht (!))

Sicherheit von Kryptosystemen

Frage: Wann ist ein (asymmetrisches) Verschlüsselungsverfahren „sicher“?

Ansatz 3: „AngreiferIn kann bei zwei möglichen Nachrichten m_0 und m_1 und gegebenem Chiffretext c nicht entscheiden, welche der möglichen Nachrichten zu *c* verschlüsselt wurde.



Das ist (im Wesentlichen) die anerkannte Definition (Indistinguishability).

Sicherheitsbeweise von Verschlüsselungsverfahren

Grundidee: Ciphertext Indistinguishability (IND)

- „Ununterscheidbarkeit von Geheimtexten“

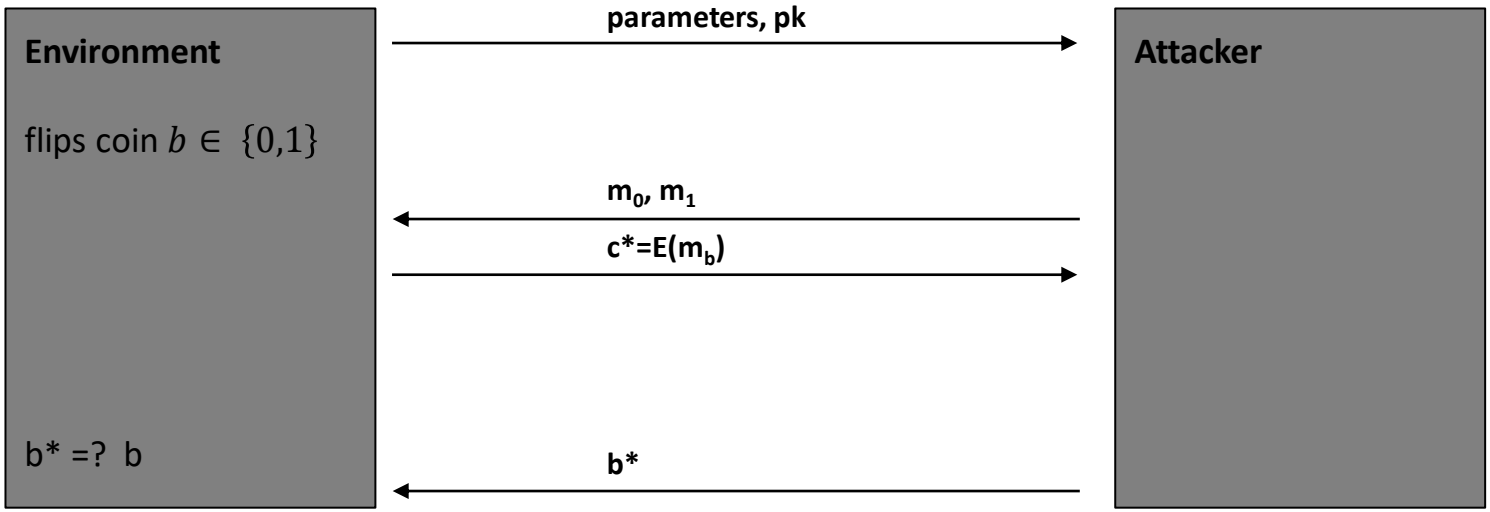
Möglichkeiten des Angreifers/der Angreiferin:

- Chosen Plaintext Attack (CPA)-AngreiferIn kann Klartexte an ein „Orakel“ schicken und bekommt sie verschlüsselt zurück
- Chosen Ciphertext Attack (CCA)-AngreiferIn kann Chiffretexte an ein „Orakel“ schicken und bekommt sie entschlüsselt zurück

Zwei „Verfahren“ der Sicherheitsbeweise:

- IND-CPA: Ununterscheidbarkeit bei Chosen Plaintext Angriffen
- IND-CCA2: Ununterscheidbarkeit bei Chosen Ciphertext Angriffen
- (Es gibt auch CCA1 (schwächer), doch das spielt keine praktische Rolle)

Ausblick: Sicherheitsdefinition: IND-CPA



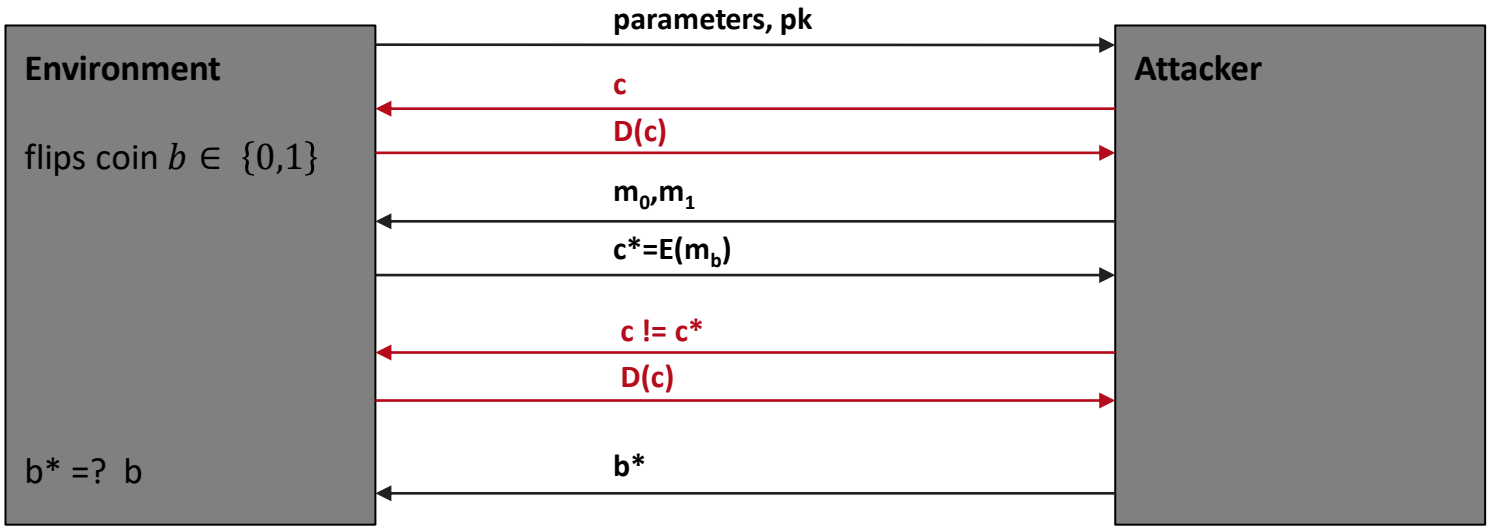
- AngreiferIn bekommt Parameter und öffentlichen Schlüssel pk.
- AngreiferIn macht beliebige Berechnungen. Sucht sich zwei Nachrichten m_0 und m_1 mit gleicher Länge.
- Environment (Verschlüsselungssystem) sucht sich ein zufälliges Bit $b \in \{0,1\}$, verschlüsselt $c^* = E(sk, m_b)$. Schickt c^* an AngreiferIn.
- AngreiferIn macht beliebige Berechnungen. Entscheidet sich für ein Output-Bit b^*
- AngreiferIn „gewinnt“ wenn $b^* = b$
- System ist „sicher“, wenn die Wahrscheinlichkeit, dass AngreiferIn das korrekte b findet, etwa ½ ist: $Pr(b^* = b) \cong ½$

Möglichkeiten des Angreifers/der Angreiferin

- In der obigen Definition bedeutet CPA "chosen plaintext attack", d. h. der Angreifer/die Angreiferin kann von ihm/ihr gewählte Klartexte verschlüsseln.
- Dies gilt immer für Public-Key-Systeme, und die entsprechende Definition für symmetrische Verschlüsselung gibt ihm/ihr Zugang zu einem Orakel, das gewählte Klartexte verschlüsselt.

- Angriffe auf ausgewählte Chiffretexte (CCA) bedeuten, dass der Angreifer auch Zugang zu einem Entschlüsselungsorakel erhält, das die Chiffretexte entschlüsselt (außer dem Herausforderungs-Chiffretext c*, natürlich)

Ausblick: Sicherheitsdefinition: IND-CCA2



Stärkere Notation: AngreiferIn kann zusätzlich das „Entschlüsselungs-Orakel“ verwenden, d.h. er/sie kann beliebige Nachrichten (ausgenommen der Challenge) verschicken und bekommt die Entschlüsselung zurück.

Ansonsten ist das Verfahren identisch zu IND-CPA.

PRP UND PRF

Pseudo-Random Permutations (PRPs)

PRPs sind ideale Formulierungen von Blockchiffren.

Wir betrachten die Funktionen $E: K \times \mathrm{X} \to \mathrm{X}$

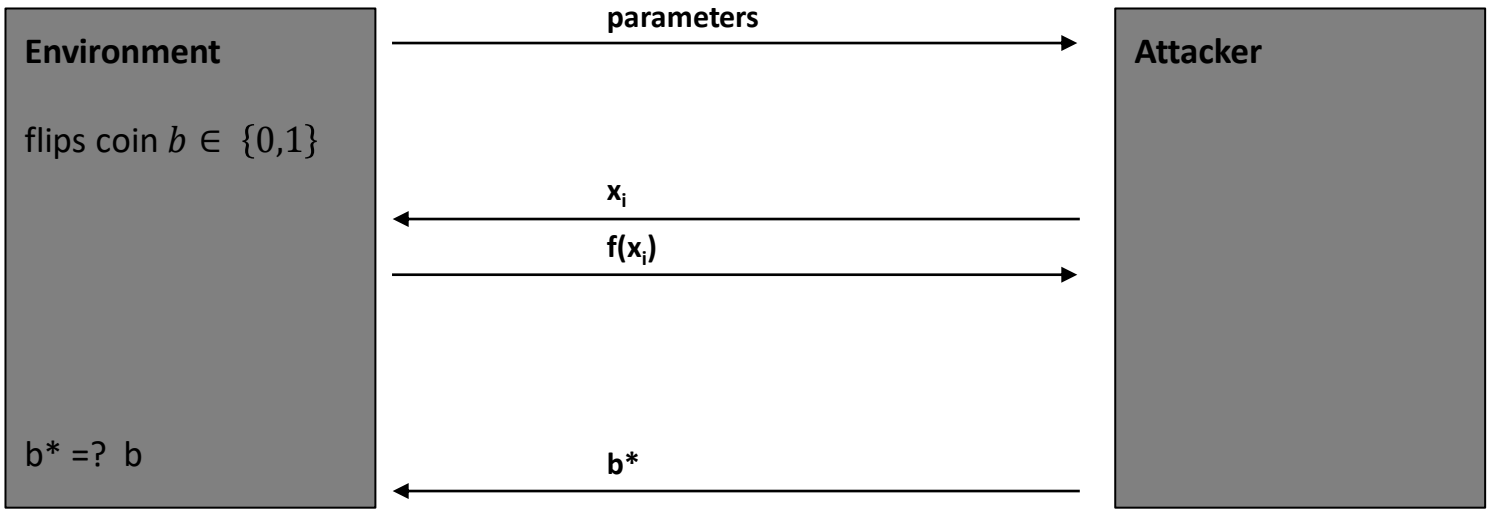
$\mathrm{Perms}[\mathrm{X}]$ sei die Menge aller Permutationen auf X.

Eine PRP ist eine Funktion $E: K \times \mathrm{X} \to \mathrm{X}$, die

- „effizient“ berechnet werden kann
- eine Inverse besitzt
- deren Inverse effizient berechnet werden kann

Eine PRP (mit Zufallsschlüssel) ist sicher, wenn sie von einer zufälligen Permutation aus $\mathrm{Perms}[\mathrm{X}]$ nicht zu unterscheiden ist.

Ausblick: Definition von PRPs



- AngreiferIn bekommt Parameter.
- Environment (Verschlüsselungssystem) sucht sich ein zufälliges Bit $b \in \{0,1\}$
  - Wenn $b = 0$ $k \leftarrow K, f \leftarrow PRP(k,\cdot)$ // f ist die PRP
  - Wenn $b = 1$ $f \leftarrow Perms[X]$ // f ist eine zufällige Permutation
- AngreiferIn macht beliebige Berechnungen. Kann nach x_i fragen und erhält $f(x_i)$ (“Verschlüsselungsorakel”)
- AngreiferIn entscheidet sich für ein Output-Bit b^*. „Gewinnt“ wenn $b^* = b$
- System ist „sicher“, wenn die Wahrscheinlichkeit, dass AngreiferIn das korrekte b findet, etwa ½ ist: $Pr(b^* = b) \cong ½$

Pseudo-Random Functions (PRFs)

- Etwas schwächer als PRPs, doch ähnliche Funktionsweise

Wir betrachten die Funktionen $\mathrm{F}: K \times \mathrm{X} \to \mathrm{Y}$

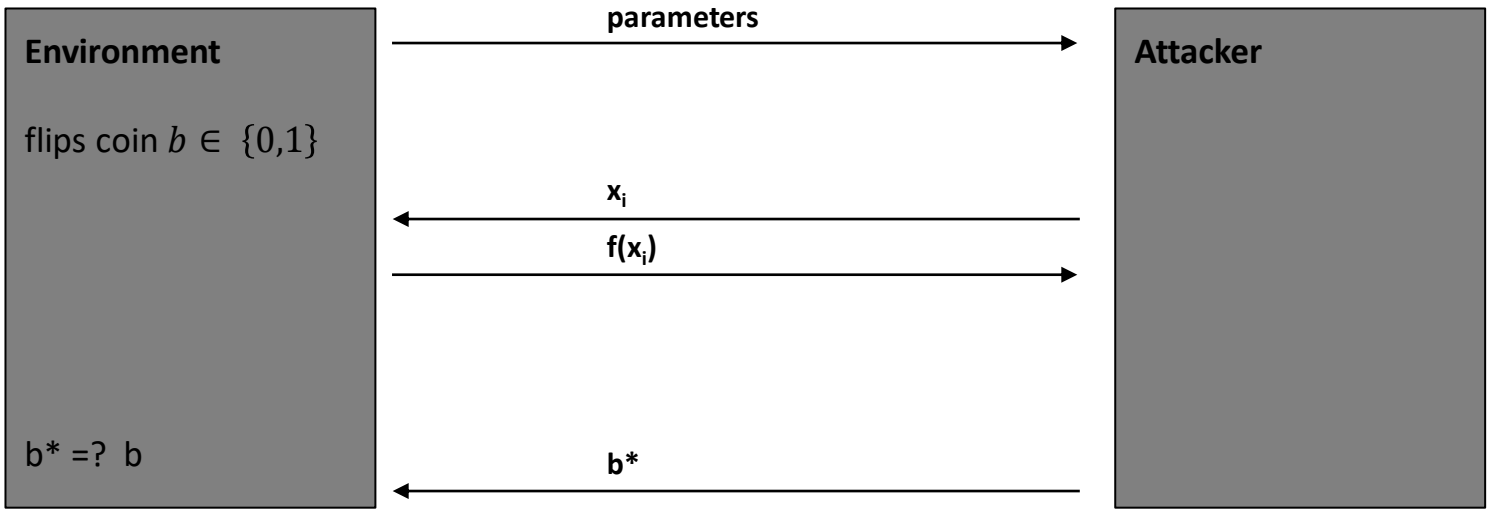
$\mathrm{Funs}[\mathrm{X}, \mathrm{Y}]$ sei die Menge aller Funktionen von X nach Y.

Eine PRF ist eine Funktion $E: K \times \mathrm{X} \to \mathrm{Y}$, die

- „effizient“ berechnet werden kann

Eine PRF ist sicher, wenn sie von einer zufälligen Funktion aus $\mathrm{Funs}[\mathrm{X}, \mathrm{Y}]$ nicht zu unterscheiden ist.

Ausblick: Definition von PRFs



- AngreiferIn bekommt Parameter.
- Environment (Verschlüsselungssystem) sucht sich ein zufälliges Bit $b \in \{0,1\}$
  - Wenn $b = 0$ $k \leftarrow K, f \leftarrow PRF(k,\cdot)$ // f ist die PRF
  - Wenn $b = 1$ $f \leftarrow Funs[X]$ // f ist eine zufällige Funktion
- AngreiferIn macht beliebige Berechnungen. Kann nach x_i fragen und erhält $f(x_i)$ ("Verschlüsselungsorakel")
- AngreiferIn entscheidet sich für ein Output-Bit b^*. „Gewinnt" wenn $b^* = b$
- System ist „sicher", wenn die Wahrscheinlichkeit, dass AngreiferIn das korrekte b findet, etwa ½ ist: $Pr(b^* = b) \cong ½$

Beispiel (unsichere) PRF

Es sei $K=X=\{0,1\}^n$

Wir betrachten die PRF $F(k,x) = k \oplus x$.

Wähle zwei $x_0,x_1 \in \{0,1\}^n$, $x_0 \neq x_1$.

Anfrage von $y_0 = f(x_0)$ and $y_1 = f(x_1)$.

Wenn $x_0 \oplus x_1 = y_0 \oplus y_1$ ist es (fast sicher) die PRF, sonst (sicher) die zufällige Funktion

PRP-PRF Switching Lemma

- Jede PRP ist auch eine PRF
  (einfach das Inverse „vergessen“)
- Jede sichere PRP ist auch eine sichere PRF (!)

- Intuitiv muss man **viele** Werte abfragen, um eine **„Kollision“** zu finden, d.h. um zu erkennen, dass es sich um eine PRF und nicht eine PRP handelt
- Das gilt, wenn X „groß“ ist

- Eine PRF ist **nicht** notwendigerweise eine PRP
  Das Inverse existiert nicht notwendigerweise und es ist möglicherweise nicht bekannt, wie es berechnet werden kann

- Aber: Wir können aus jeder PRF eine PRP konstruieren! → Feistel-Netzwerk

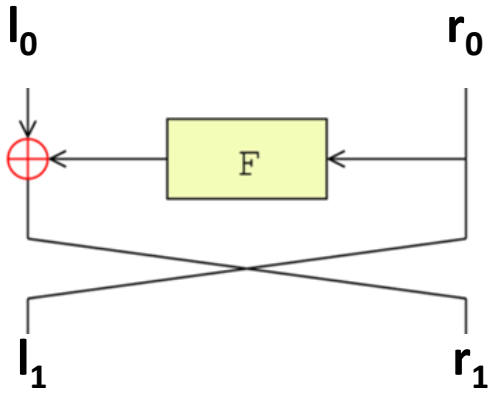
Nochmal Feistel

PRF F wird in Feistel-Netzwerk verwendet.

Ist eine Runde genug?

(Offenkundig) Nein!

Die rechte Hälfte (neue linke Hälfte) ist unverändert.



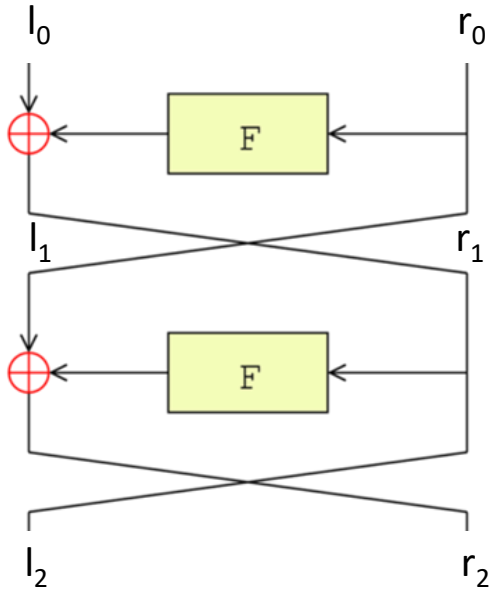
Nochmal Feistel

PRF F wird in Feistel-Netzwerk verwendet.

Sind zwei Runden genug?

Nein!

Es gilt: $l_0 \oplus l_2 = F(r_0)$



Das bedeutet: Wenn wir r_0 nicht verändern und in l_0 Bits flippen, dann sind dieselben Bits in l_2 geflippt.

Das ermöglicht es der Angreiferin oder dem Angreifer, die Konstruktion von einer zufälligen Permutation zu unterscheiden.

Ausblick: Nochmal Feistel

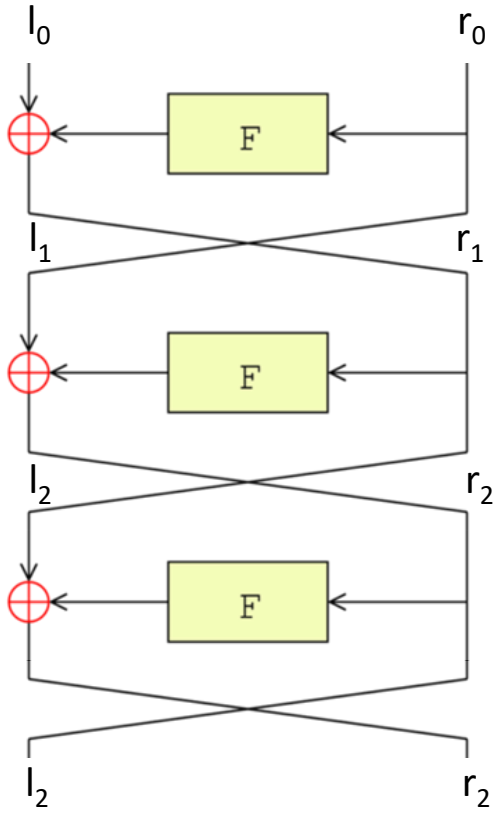
PRF F wird in Feistel-Netzwerk verwendet.

Sind drei Runden genug?

Ja. Allerdings kann bewiesen werden, dass vier Runden ein PRP mit zusätzlichen Sicherheitsgarantien geben

(Warum hat DES dann 16 Runden?

- Die Rundenfunktion ist schwach und keine PRF.)



Leibniz Universität Hannover

ABSCHLUSS KRYPTOGRAPHIE

Überblick: „Starke“ Kryptosysteme

Wichtig: Immer auf dem aktuellen Stand bleiben! Ständige Veränderungen!

Symmetrische Verfahren

- AES, 256 bits, CBC
- DES sollte nicht verwendet werden!
- Vorsicht bei der Verwendung von Strom-Chiffren!
    - RC4 wird nicht mehr empfohlen!

Hashfunktionen

- SHA-512
- MD-5 oder SHA-1 sollten nicht verwendet werden (außer es sind keine kryptographischen Eigenschaften notwendig)

Asymmetrische Verfahren

- RSA mit 2048 Bits
    - Keine kleineren Schlüsselgrößen verwenden!
    - 3072 Bits sollten verwendet werden wenn eine Anwendung über das Jahr 2030 hinaus geplant ist.
- Empfehlung: Elliptische Kurven (kein Thema dieser Vorlesung)

Spickzettel

Asymmetrische Verschlüsselung:

- Verwende den öffentlichen Schlüssel, um den Klartext zu verschlüsseln
- Verwende den privaten Schlüssel, um den Chiffre-Text zu entschlüsseln

Digitale Signaturen:

- Verwende den privaten Schlüssel, um eine Nachricht zu signieren
- Verwende den öffentlichen Schlüssel, um eine Nachricht zu verifizieren

Zusammenfassung

Asymmetrische Kryptographie

- Asymmetrische Schlüssel → einfacher Schlüsselaustausch
- Skalierbare Verschlüsselung mit unterschiedlichen Kommunikationsparteien

Sicherheit basiert auf der „Falltür“ Einwegfunktion

- Faktorzerlegung großer Zahlen → RSA Algorithmus
- Diskrete Logarithmen → Diffie-Hellman Schlüsselaustausch
- Sicherheit basiert auf der großen Schlüsselgröße (>2.000 Bits)

FRAGEN BIS HIERHER?

EVALUATION DER HEUTIGEN VORLESUNG

… oder über Stud.IP im Ordner der Vorlesung