"Practical Cryptography" Kapitel 8, 9, 15, 16 und 22 (Ferguson/Schneider)

Seminar Internetsicherheit
TU-Berlin

Martin Eismann

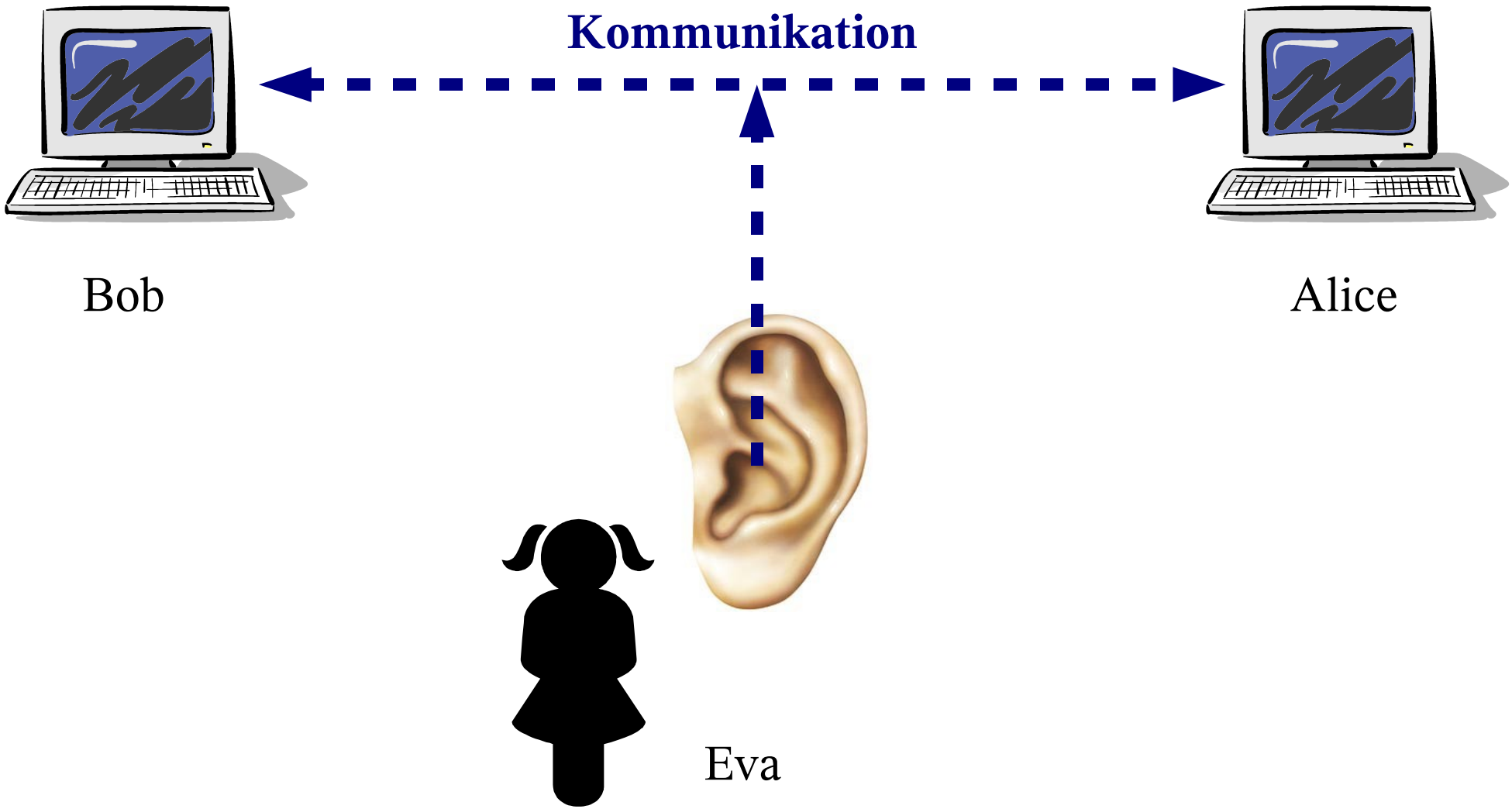
Was ist Sicherheit?

- Ausgetauschte Nachrichten schützen vor:
  - Mitlesen
  - Löschen
  - Modifikation
  - Abfangen
- Konkret heißt das:
  - Verschlüsseln
  - Authentifizieren

Übersicht

1. Das Diffie-Hellman-Protokoll
2. Mit Sitzungsschlüsseln arbeiten
3. Implementierungs-Schwächen
4. Sichere Datenspeicherung

1. Diffie-Hellman-Protokoll (1)



1. Diffie-Hellman-Protokoll (2)

Problem: Eva hört mit.

Lösung:

1) Verschlüsseln
2) Authentifizieren

Aber wie gibt Bob Alice einen Schüssel, ohne das Eva ihn mithört?

1. Diffie-Hellman-Protokoll (3)

Wir stellen Eva ein unlösbares mathematisches Problem:

Das *Diskreter-Logarithmus-Problem*

Vorher noch ein bißchen Schulmathematik...

1. Diffie-Hellman-Protokoll (4)

Bob wählt Geheimzahl:

$x = ?$ *(geheim!)*

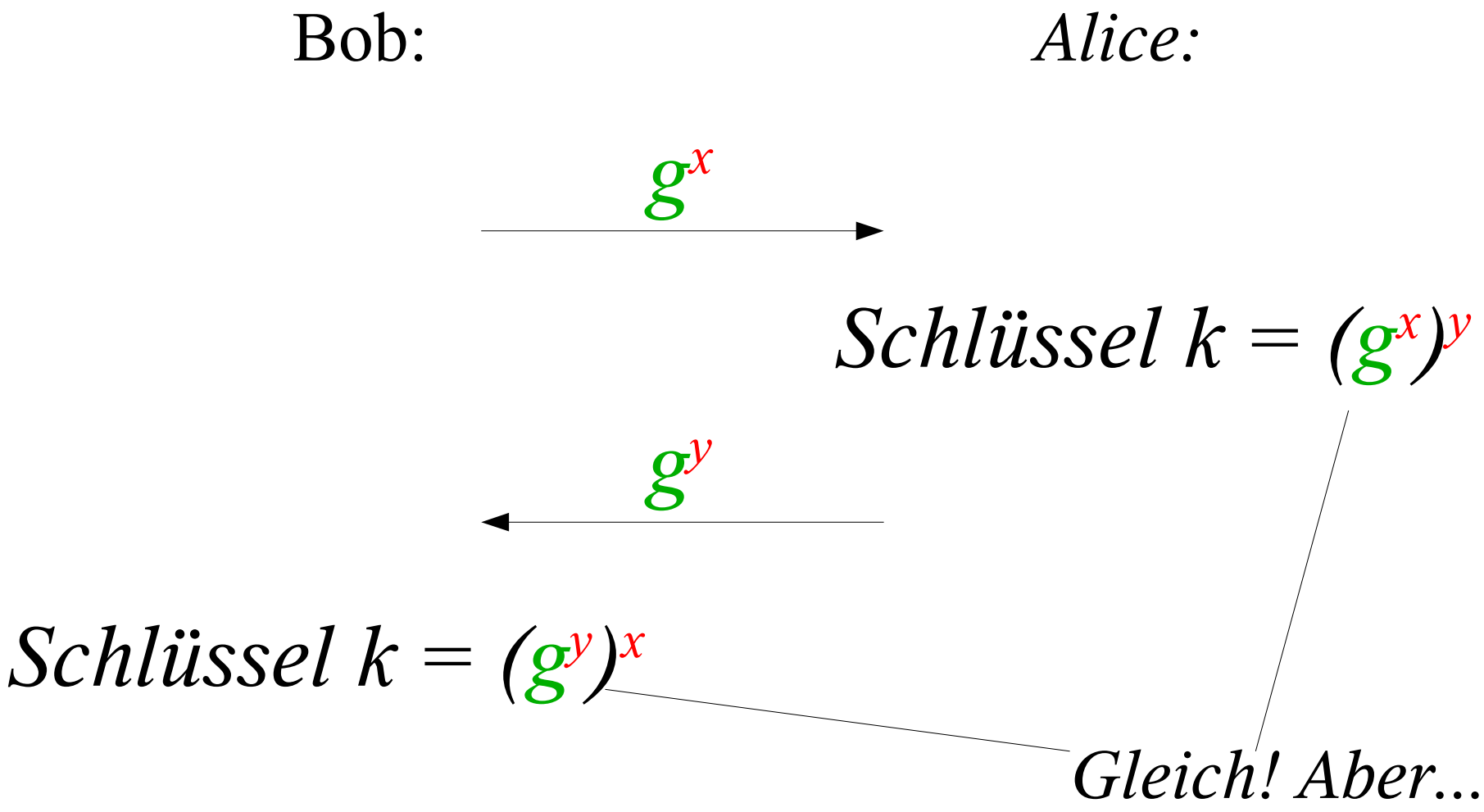
Alice wählt Geheimzahl:

$y = ?$ *(auch geheim)*

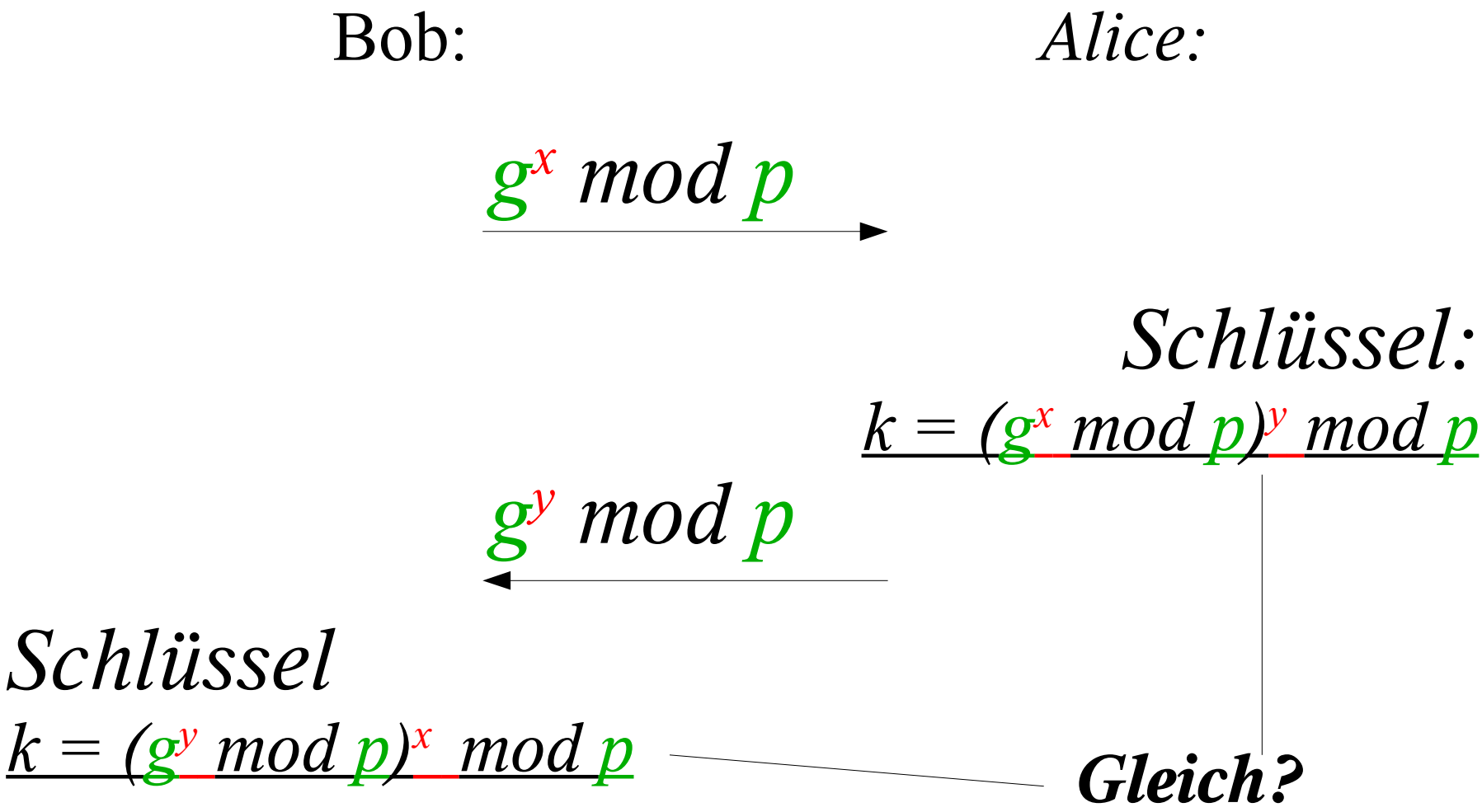
Für alle sichtbar wird festgelegt:

$g = 3$ *(Primzahl!)*

1. Diffie-Hellman-Protokoll (5)



1. Diffie-Hellman-Protokoll (6)



1. Diffie-Hellman-Protokoll (7)

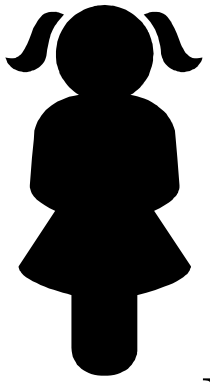
$g = 3$
$p = 19$

$x = 8$
$y = 11$

$$k = (g^y \bmod p)^x \bmod p = (3^{11} \bmod 19)^8 \bmod 19 = 17$$

$$k = (g^x \bmod p)^y \bmod p = (3^8 \bmod 19)^{11} \bmod 19 = 17$$

1. Diffie-Hellman-Protokoll (8)

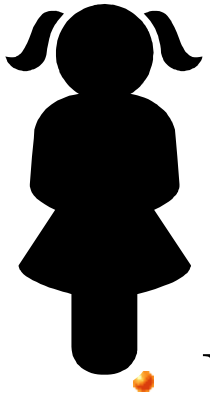


Und Eva?

- Hat zwei Gleichungen...
- ... mit zwei Unbekannten:

$$\textbf{\textit{I:}}\ g^{x} \bmod p = \textit{abgehörter Wert 1}$$
$$\textbf{\textit{II:}}\ g^{y} \bmod p = \textit{abgehörter Wert 2}$$

1. Diffie-Hellman-Protokoll (9)



Und Eva?

- Hat zwei Gleichungen...
- ... mit zwei Unbekannten:

$$\textbf{\textit{I:}}\ g^x \bmod p = \textit{abgehörter Wert 1}$$
$$\textbf{\textit{II:}}\ g^y \bmod p = \textit{abgehörter Wert 2}$$

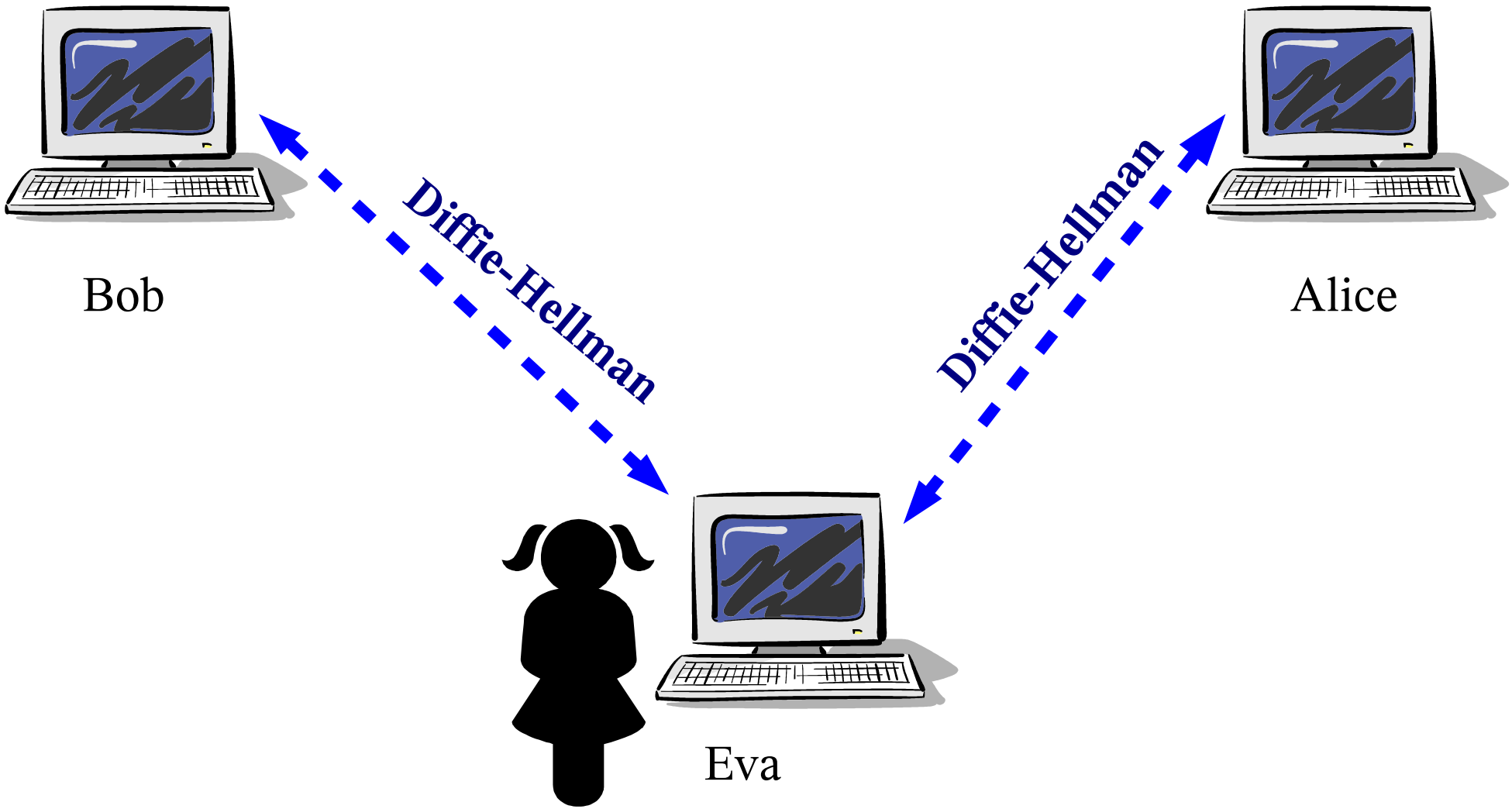
- Problem des *Diskreten Logarithmus*

1. Diffie-Hellman-Protokoll (10)

- Bis jetzt: Erfolgreich **verschlüsselt**!
- **Authentifizierung**?

„The-(wo)man-in-the-middle-attack:“

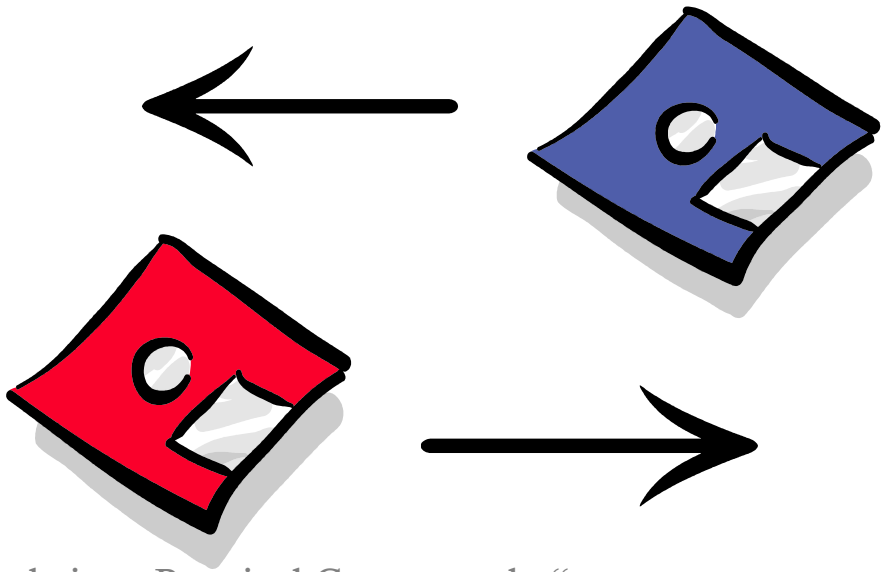
1. Diffie-Hellman-Protokoll (11)



1. Diffie-Hellman-Protokoll (12)

Lösung:

- Authentifizierungsschlüssel festlegen
- ... und austauschen

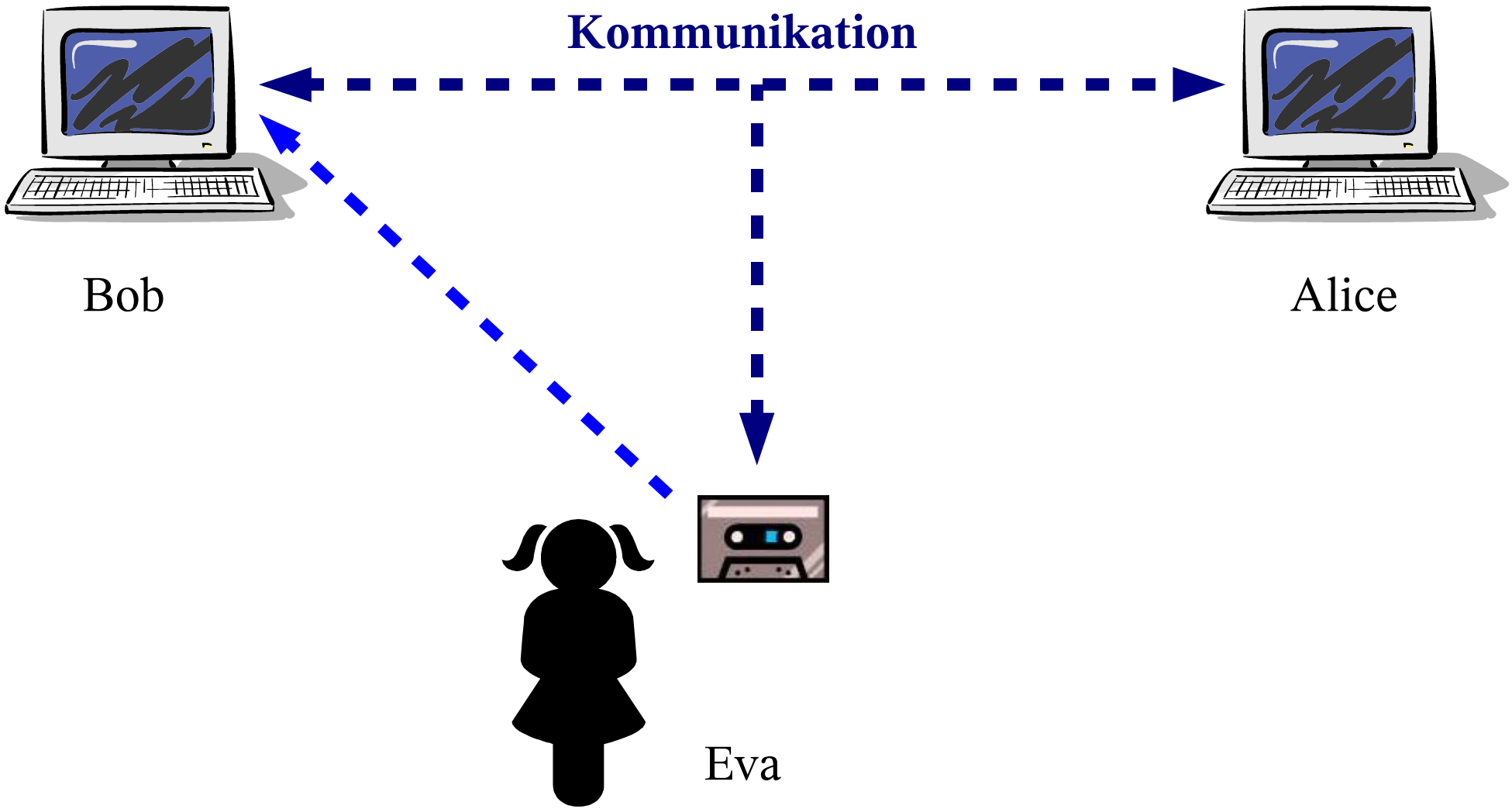


1. Diffie-Hellman-Protokoll (13)

Noch ein letztes Problem beheben:

Replaying

1. Diffie-Hellman-Protokoll (14)



1. Diffie-Hellman-Protokoll (15)

Lösung zu Replaying:

- p, q für jede Sitzung neu zufällig auswählen

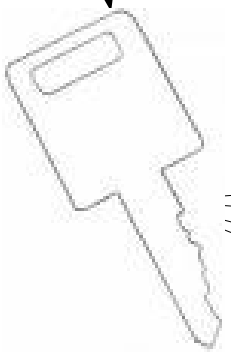
$$(g_1^{\,x} \bmod p_1) \rightarrow (g_2^{\,x} \bmod p_2)$$
$$(g_1^{\,y} \bmod p_1) \rightarrow (g_2^{\,y} \bmod p_2)$$

Übersicht

1. Das Diffie-Hellman-Protokoll
2. Mit Sitzungsschlüsseln arbeiten
3. Implementierungs-Schwächen
4. Sichere Datenspeicherung

2. Schlüsselverwendung (1)

$k = (g^y \bmod p)^x \bmod p$ oder
$k = (g^x \bmod p)^y \bmod p$

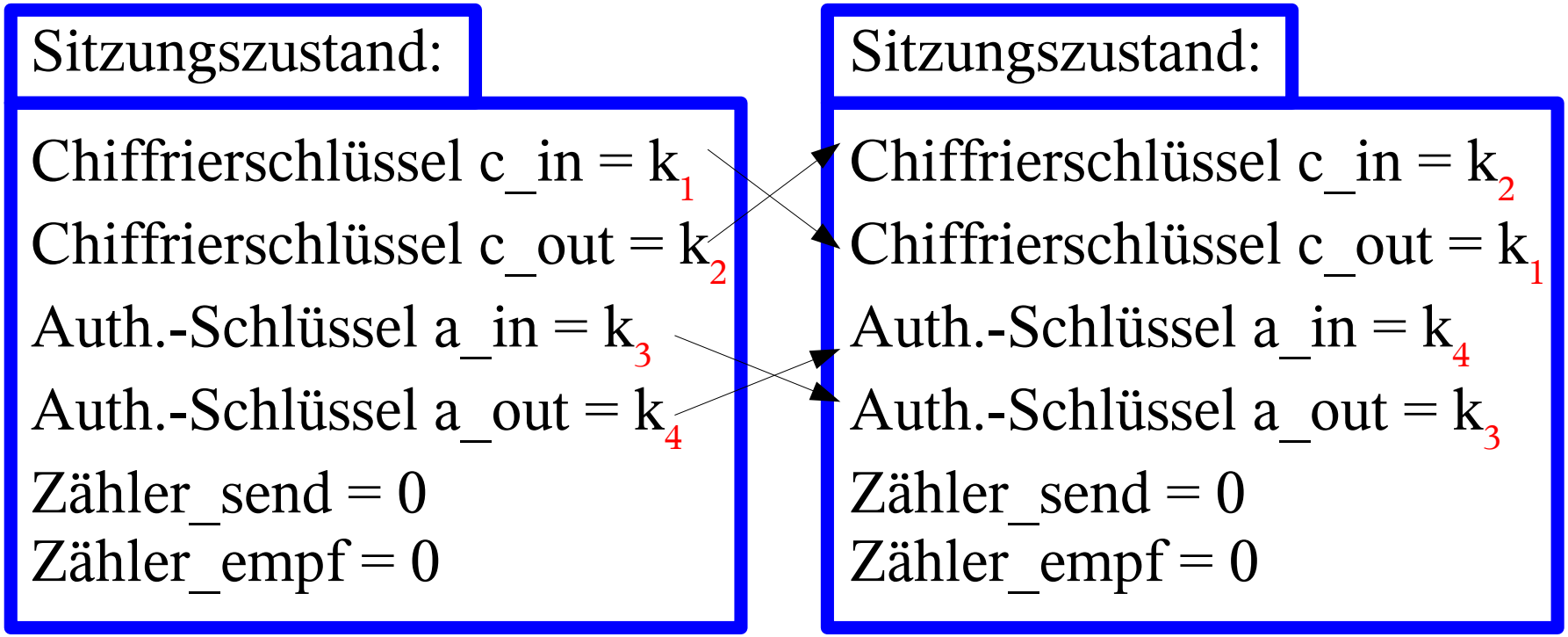


4x Ableiten

- Chiffrierschlüssel c_in
- Chiffrierschlüssel c_out
- Authentif.-Schlüssel a_in
- Authentif.-Schlüssel a_out

2. Schlüsselverwendung (2)

- Alice
- Bob



Anfängliche Objekte des Sitzungszustandes auf beiden Seiten

2. Schlüsselverwendung (3)

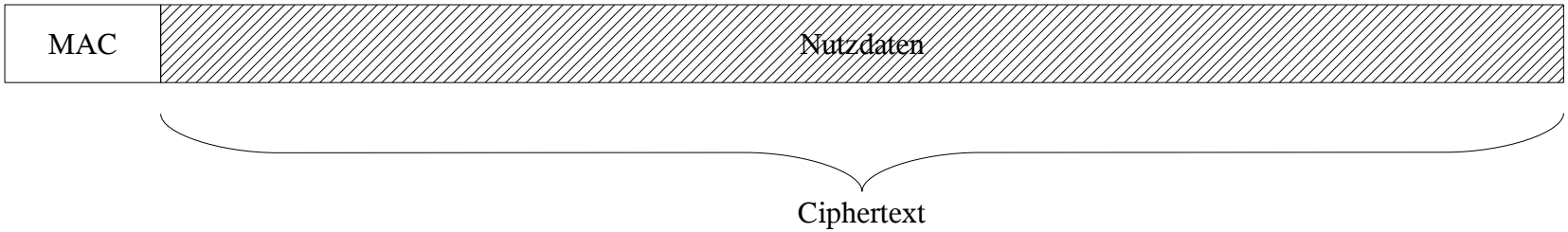


Abb. 1: Verschlüsseln zuerst

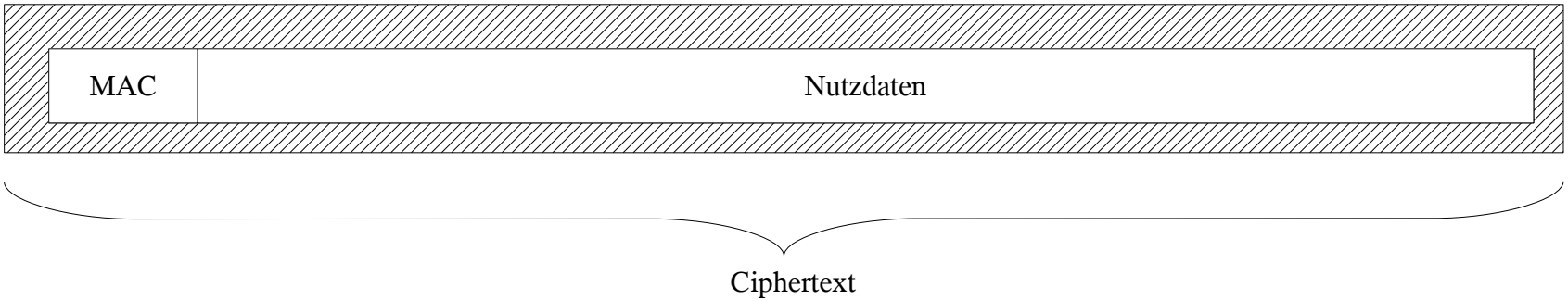


Abb. 2: Authentifizieren zuerst

2. Schlüsselverwendung (4)

Was passiert, wenn der Nummernraum des Zählers erschöpft ist?

Übersicht

1. Das Diffie-Hellman-Protokoll
2. Mit Sitzungsschlüsseln arbeiten
3. Implementierungs-Schwächen
4. Sichere Datenspeicherung

3. Implementierungs-Schwächen (1)

Praxis: Sichere Verbindungen werden durch fehlerhafte Implementierung geknackt, nicht durch fehlerhafte Verschlüsselung

Beispiele für typische Fehler in diesem Abschnitt

3. Implementierungs-Schwächen (2)

- Unsichere Betriebssysteme
- Keine „korrekten“ Programme – Spezifikation oft schon unklar
- Test & Fix – Abwesenheit von Fehlern?
- Laxe Einstellung der IT-Industrie
- Sichere Software vs. Korrekte Software

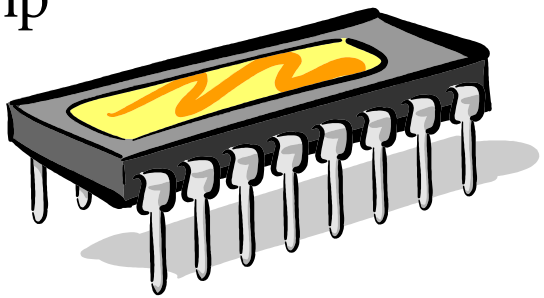
3. Implementierungs-Schwächen (3)

”Standard-Implementierungstechniken sind völlig ungeeignet um sicheren Code zu erstellen.“

Niels Ferguson und Bruce Schneier

3. Implementierungs-Schwächen (4)

- Sicherheitskritische Implementierungsfehler im einzelnen:
  - Schwachstellen in der Speicherverwaltung
    - Zustandsobjekt des Protokolls (mit Schlüsseln)
    - C: memset; Java?
    - Virtueller Speicher
    - Windows: Debugger; UNIX: Coredump



3. Implementierungs-Schwächen (5)

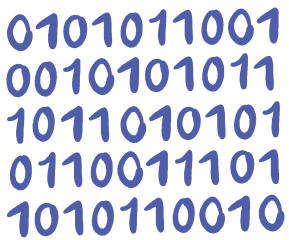
- Mißachtung von Programmierprinzipien:
  - Einfachheit
  - Modularisierung
  - Zusicherungen
  - Pufferüberläufe
  - Testläufe

3. Implementierungs-Schwächen (6)

- Seitenkanalangriffe:
- Seitenkanäle sind z.B. die
  - verbrauchte Rechenzeit,
  - der Stromverbrauch, (SmartCards)
  - Elektromagnetische Abstrahlung,
  - Speicherbedarf,
  - Reaktionen auf Falscheingaben
- Können gezielt abgeschwächt werden

3. Implementierungs-Schwächen (7)

- Große-Ganzzahlen-Rechnungen:
  - Etablierte Bibliotheken auf Fehler testen
  - Nur auf Assembler-Ebene testbar → Test zur Laufzeit
  - Wooping:
    - $(10 * 11) \bmod 7 = 5$
    - $[(10 \bmod 7) * (11 \bmod 7)] \bmod 7 = 5$



3. Implementierungs-Schwächen (8)

- Timeout-Verhalten richtig implementieren:
- Verbindungsaufbau-Bombardement kann zum Speicherüberlauf führen (Protokoll-Zustandsobjekte)

Übersicht

1. Das Diffie-Hellman-Protokoll
2. Mit Sitzungsschlüsseln arbeiten
3. Implementierungs-Schwächen
4. Sichere Datenspeicherung

4. Sichere Datenspeicherung (1)

- Memorisierbare Paßwörter: Kürze? Sprachschatz?
- Salzen-und-Strecken-Methode – kurze Paßwörter zu starken Schlüsseln machen:

  - $x_0 := 0$
  - $x_i := h(x_{i-1} \| p \| s)$ *mit $i = 1,...,r$; $p = PW$; $s = Salz$*
  - $K := x_r$

4. Sichere Datenspeicherung (2)

- Secure Tokens
- Fingerabdruckscanner
- Single-Sign-On-Systeme
- Schlüssel mehrfach hinterlegen – Banktresor
- Geheimnisse von permanenten Medien löschen,
- am besten zerstören, wenn

  nicht mehr gebraucht