Vorlesung Sicherheit

Dennis Hofheinz

ITI, KIT

14.04.2014

Überblick

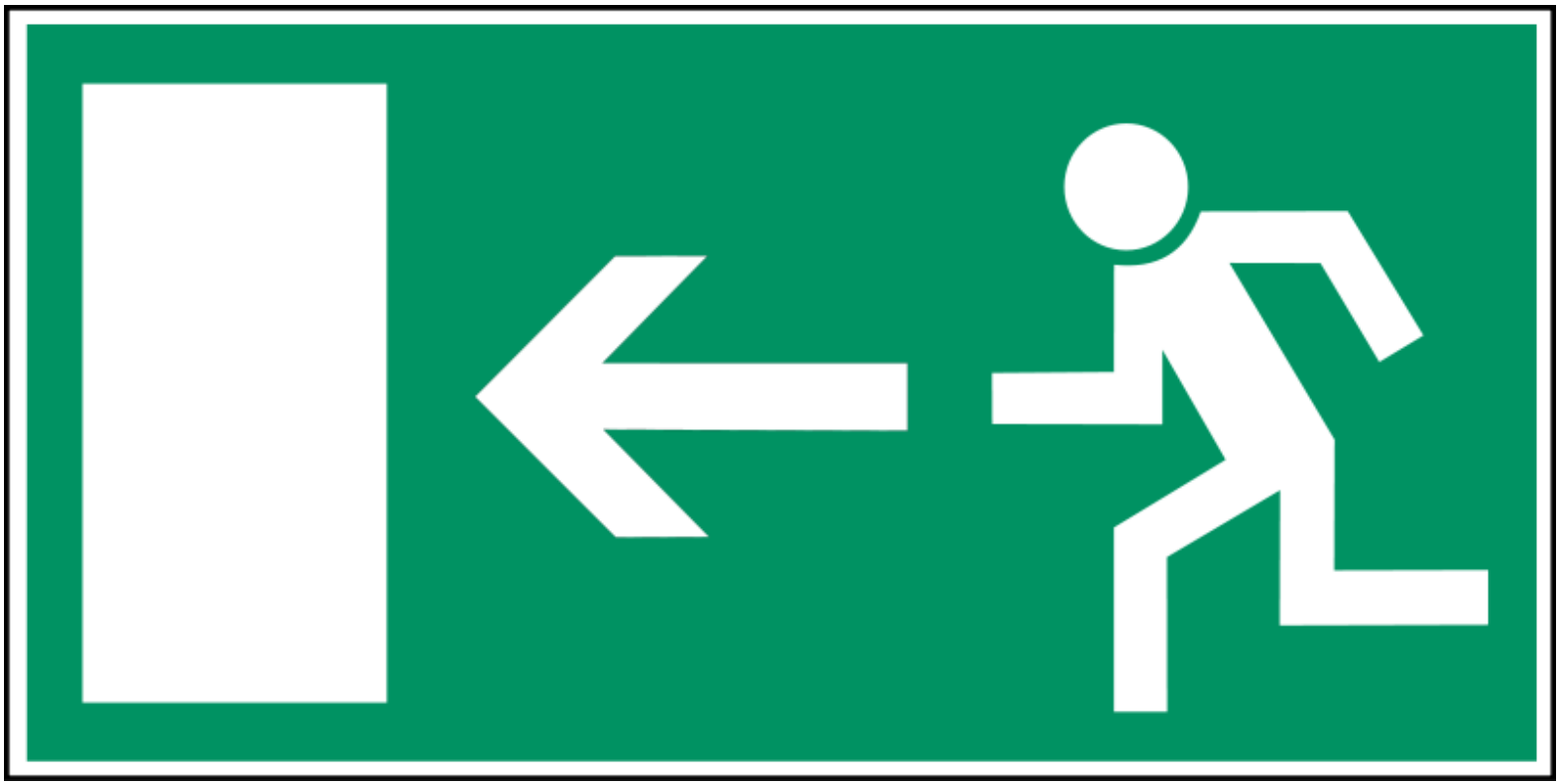
Was ist Sicherheit?

Quelle: Wikipedia

Was ist Sicherheit?

Quelle: Wikipedia

Was ist Sicherheit?



Quelle: Wikipedia

Was ist Sicherheit?

Quelle: Wikipedia

Was ist Sicherheit?

Quelle: Wikipedia

Was ist Sicherheit?

Was Sicherheit für uns ist

- Generell Sicherheit weit gefasster Begriff
- Unser Fokus: Sicherheit in der Informationstechnik
- Keine Definition, sondern Beispiele:
  - Schutz von Daten (z.B. E-Mails) vor unberechtigtem Zugriff
  - Zugriffskontrolle (z.B. Rechteverwaltung von Betriebssystem)
  - Privatsphäre in verteilten Systemen (z.B. sozialen Netzwerken)
- Beobachtung: *Bausteine* (z.B. Verschlüsselung) entscheidend
- Genaue Definition von Szenario abhängig

Überblick

Ziele der Vorlesung

- Fokus: entscheidende *Bausteine* für sichere digitale Systeme
- Insbesondere: kryptographische Bausteine
- **Nicht:** vollständige Systeme
- Beispiele:
  - Verschlüsselungsverfahren (symmetrisch und asymmetrisch)
  - Authentifikationsmechanismen (z.B. Signaturen, Protokolle)
  - Strategien der Zugriffskontrolle (z.B. Bell-LaPadula)
- **Nicht**-Beispiele:
  - Sichere Softwareentwicklung (aber: ausgewählte Fehler)
  - Sichere Systementwicklung (z.B. durch Virtualisierung)
  - Schadsoftware (z.B. Viren)

Vorgehensweise

- Orientiert an bisheriger Sicherheits-Vorlesung
- Augenmerk auf formaler Vorgehensweise
  - Definitionen/Beweise, wenn möglich
  - Aber: generell Überblickscharakter
  - Zur Auflockerung Anwendungen/Ausblick/aktuelle Forschung
- Schnittstelle für weiterführende Vorlesungen

Vorlesungsmaterialien

- Vorlesungswebseite

  `https://crypto.iti.kit.edu/index.php?id=sic-sose14`

- Vorlesungsfolien ($\rightarrow$ Webseite)
- Skript (wird angepasst, $\rightarrow$ Webseite)
- Gelegentlich Verweise auf Webseiten
- Buchempfehlung (nur ausgewählte Kapitel):
  - Katz/Lindell: Introduction to Modern Cryptography
  - Anderson: Security Engineering (Link auf Webseite)
- **Gesucht:** Hiwi zur Skriptanpassung

Übung und Klausur

- Übung: Wiederholung/Vertiefung/Anwendung von Vorlesung
- Mittel: Übungsblätter
  - Nicht bewertet, in Übung besprochen
  - **Blatt 1 verfügbar** ($\rightarrow$ Webseite)
- Übung donnerstags, etwa 14-tgl. ab 24.04.
- Übungsleiter `Florian.Boehl@kit.edu`
- Klausur schriftlich, 60min
  - Klausur Dienstag, 22.07.2014 14:00
  - Nachklausur Montag, 29.09.2014, 14:00
  - Frühere Klausuren auf Webseite

Fragen?

- . . . Fragen?

Überblick

Motivation: zwei Vorgehensweisen

- „Arms race security“:
  1. Designer entwirft System
  2. Angreifer findet Schwachstelle
  3. Designer repariert Schwachstelle
  4. Goto 2

  - Theoretisch unbefriedigend
- „Provable security“:
  1. Designer entwirft Sicherheitsmodell und System
  2. Designer versucht, System als sicher zu beweisen
  3. Angreifer findet Schwachstelle im Sicherheitsmodell
  4. Designer repariert Sicherheitsmodell und System
  5. Goto 2

  - Unvollkommen, aber verlässlicher und theoretisch zugänglicher

Provable security

- Zentraler Bestandteil von „Provable security“: Modularität
  - Subsysteme geben fest definierte Garantien
  - Beispiel: Chiffrate von Verschlüsselung nicht effizient unterscheidbar von Zufall
  - Sicherheitsgarantien größerer Systeme werden aus Sicherheitsgarantien von Subsystemen abgeleitet
- Deshalb folgendes Vorgehen:
  - Bausteine mit wohldefinierten Sicherheitsgarantien entwickeln
  - Danach komplexere Systeme aus Bausteinen zusammensetzen
  - Beispiel: TLS aus Schlüsselaustausch und Verschlüsselung zusammengesetzt

    (Allerdings: TLS nicht das beste Beispiel für modulares Design)

Konkreter inhaltlicher Plan

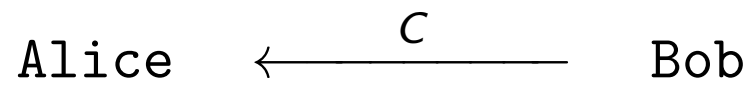
1. Kryptographische Bausteine
   - Symmetrische Verschlüsselung (Stromchiffren, Blockchiffren)
   - Hashfunktionen (Eigenschaften, Konstruktion, Beispiele)
   - Public-Key-Verschlüsselung (RSA, ElGamal, sichere Varianten)
   - Authentifikation von Nachrichten (MACs, Signaturen)
2. Anwendungen: komplexere Protokolle
   - Schlüsselaustausch/-verteilung (Kerberos, TLS, Angriffe)
   - Identifikationsprotokolle (auch Zero-Knowledge-Protokolle)
3. Weitere Anwendungen:
   - Nutzerauthentifikation (Passwortabfragen, Angriffe, mehr)
   - Zugriffskontrollmechanismen
4. Häufige Sicherheitsprobleme (Buffer Overflows, XSS,  et al.)

Überblick

Überblick

Ziel

- Sichere Nachrichtenübermittlung auf unsicherem Kommunikationskanal:



- Statt Nachricht M wird Chiffrat C übertragen
- Anforderungen:
  - `Bob` muss Chiffrat aus Nachricht berechnen
  - `Alice` muss Nachricht aus Chiffrat berechnen
  - Chiffrat soll „Außenseiter“ keinen Hinweis auf Nachricht geben

Überblick

Geheime Verfahren

- Veraltet: Verschlüsselungsverfahren geheimhalten

$$\texttt{Alice}_{\mathrm{Dec}} \xleftarrow{C := \mathrm{Enc}(M)} \texttt{Bob}_{\mathrm{Enc}}$$

- Chiffrat wird durch Funktion Enc erzeugt: $C := \mathrm{Enc}(M)$
- Nachricht mit Funktion Dec berechnen: $M = \mathrm{Dec}(C)$
- *Wichtig:* nur `Alice` und `Bob` kennen Dec und Enc

Geheime Verfahren: Beispiel

$$\texttt{Alice}_{\mathrm{Dec}} \xleftarrow{C:=\mathrm{Enc}(M)} \texttt{Bob}_{\mathrm{Enc}}$$

- Cäsar$_3$-Verschlüsselung:
  - Interpretiere M als Zahlenfolge $M = (M_i)_{i=1}^n \in \{0, \ldots, 25\}^n$
  - Jede Zahl steht für einen Buchstaben ($\texttt{A} \leftrightarrow 0, \ldots, \texttt{Z} \leftrightarrow 25$)
  - Enc verschiebt jede Stelle von M zyklisch um 3:

$$\mathrm{Enc}(M) := (C_i)_{i=1}^n \quad \text{mit} \quad C_i := M_i + 3 \bmod 26$$

  - Dec verschiebt jede Stelle von M zyklisch um -3
  - Beispiel: $\mathrm{Enc}(\texttt{KIT}) = \texttt{NLW}$
- **Fun fact:** Cäsar$_3$ immer noch verwendet

Überblick

Kerckhoffs' Prinzip

- Problem bei geheimen Verfahren: unsicher, sobald Verfahren bekannt
- Auguste Kerckhoffs (1835–1903): parametrisierte Verfahren
  - Enc und Dec haben Schlüssel K als zusätzliche Eingabe
  - Enc und Dec dürfen bekannt sein, K muss geschützt werden
  - Vorteil: Enc und Dec von vielen Benutzern gleichzeitig nutzbar

  (Vorteil nur konzeptionell: grundsätzlich sind parametrisierte Verfahren als unparametrisierte Verfahren interpretierbar)

Symmetrische Verfahren

- Traditionell: symmetrische Verschlüsselung

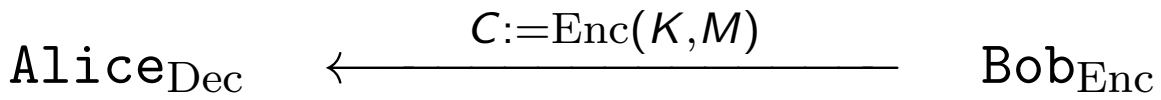
$$\texttt{Alice}_K \xleftarrow{C:=\mathrm{Enc}(K,M)} \texttt{Bob}_K$$

- `Alice` und `Bob` besitzen gemeinsames Geheimnis K
- Chiffrat wird durch Funktion[1] Enc erzeugt: $C := \mathrm{Enc}(K, M)$
- Nachricht mit Funktion Dec berechnen: $M = \mathrm{Dec}(K, C)$

[1] Enc kann allerdings auch probabilistisch sein

Überblick

Das Cäsar-Verfahren



- Cäsar-Verschlüsselung:
  - Interpretiere M als Zahlenfolge $M = (M_i)_{i=1}^n \in \{0, \ldots, 25\}^n$
  - Jede Zahl steht für einen Buchstaben ($\mathtt{A} \leftrightarrow 0, \ldots, \mathtt{Z} \leftrightarrow 25$)
  - Enc verschiebt jede Stelle von M zyklisch um K:

$$\mathrm{Enc}(K, M) := (C_i)_{i=1}^n \quad \text{mit} \quad C_i := M_i + K \bmod 26$$

  - Beispiel: $\mathrm{Enc}(25, \mathtt{KIT}) = \mathrm{Enc}(-1, \mathtt{KIT}) = \mathtt{LHS}$
  - **Frage:** Warum immer noch schlechte Idee?

Unsicherheit des Cäsar-Verfahrens

- Erinnerung: $\mathrm{Enc}(K, M) = (M_i + K \bmod 26)_{i=1}^{n}$
- Problem: kleiner Schlüsselraum ($K \in \{0, \ldots, 25\}$)
- Annahme: M „sinnvoller“ Text

  (genauer: $K' \neq K \Rightarrow \mathrm{Dec}(K', C)$ als unsinnig erkennbar)
- Dann: „Durchprobieren“ von $K' \in \{0, \ldots, 25\}$ liefert K
- **Frage:** Was, wenn M kein sinnvoller Text?

Überblick

Generalisierung: das Vigenère-Verfahren

- Cäsar$_3$: $(C_1, C_2, \dots) = (M_1 + 3, M_2 + 3, \dots) \bmod 26$
- Cäsar: $(C_1, C_2, \dots) = (M_1 + K, M_2 + K, \dots) \bmod 26$
- Vigenère: $K \in \{0, \dots, 25\}^m$ und

$$\begin{aligned}(C_1, C_2, \dots) = (&M_1, M_2, \dots, M_m, M_{m+1}, \dots) \\ &+ (K_1, K_2, \dots, K_m, K_1, \dots) \bmod 26\end{aligned}$$

- Allgemein: $C_i = M_i + K_{(i-1 \bmod m)+1} \bmod 26$

Unsicherheit des Vigenère-Verfahrens

- Erinnerung:

$$(C_1, C_2, \dots) = (M_1, M_2, \dots, M_m, M_{m+1}, \dots) + (K_1, K_2, \dots, K_m, K_1, \dots) \bmod 26$$

- Beobachtung:

$$(C_1, C_{m+1}, \dots) = (M_1 + K_1, M_{m+1} + K_1, \dots) \bmod 26$$

- Unterfolge $(C_1, C_{m+1}, C_{2m+1}, \cdots)$ Cäsar-verschlüsselt mit K_1
- Ansatzpunkt für Kryptoanalyse (rate m, finde $K_1, K_2, \dots$)

Überblick

Weitere einfache Verfahren

- Substitutionschiffren (Substitution von Buchstaben)
- Permutationschiffren (Reihenfolge ändern)
- Kombinationen der Konzepte (z.B. $\vec{C} = \mathbf{M} \cdot \vec{M}$)
  - Schlüssel $K = \mathbf{M}$ kann z.B. mit linearer Algebra gefunden werden, wenn Klartext-/Chiffratpaare bekannt

Überblick

One-Time-Pad

- Hauptproblem des Vigenère-Verfahrens: periodische Wiederverwendung des Schlüssels
- One-Time-Pad (auf Bits, d.h. $M \in \{0,1\}^n$):
  - Schlüssel so lang wie Nachricht: $K \in \{0,1\}^n$
  - $\mathrm{Enc}(K, M) = C = M \oplus K \in \{0,1\}^n$
  - $\mathrm{Dec}(K, C) = C \oplus K$
- Wichtig: $K \in \{0,1\}^n$ gleichverteilt gezogen

Sicherheit des One-Time-Pad

- Erinnerung: $\mathrm{Enc}(K, M) = C = M \oplus K$
- Gegeben C ist jedes M möglich (und gleich wahrscheinlich)

$\Rightarrow$ OTP hat *informationstheoretische* Sicherheitseigenschaften

  - Selbst unbeschränkter Angreifer erhält durch C keine Information über M
  - Kann (und soll) formalisiert werden, heute aber noch nicht

- **Aber:** OTP oft missverstanden als Patentlösung

Unsicherheit des One-Time-Pad

- Erinnerung: $\mathrm{Enc}(K, M) = C = M \oplus K$
- Nachteil: Schlüssel unhandlich (so lang wie Nachricht)
- . . . zudem: Schlüssel darf nicht wiederverwendet werden
  - Aus $C = M \oplus K$ und $C' = M' \oplus K$ folgt $C \oplus C' = M \oplus M'$, eine nichttriviale Information über M und M'
- Problem: Chiffrate verwundbar ($C \oplus X = (M \oplus X) \oplus K$)
  - Beispiel: wenn $M \in \{\texttt{ja}, \texttt{nein}\}$, dann kann ein Chiffrat C mittels $C \oplus (\texttt{ja} \oplus \texttt{nein})$ „semantisch“ verfälscht werden

Überblick

Motivation

- One-Time-Pad unhandlich (langer Schlüssel $K \in \{0,1\}^n$)
- Idee: OTP-„Simulation“ mit kurzem $K \in \{0,1\}^k$ (mit $k < n$)
  - Erweitere K zunächst: $K' := \mathrm{G}(K) \in \{0,1\}^n$ für geeignetes G
  - Verwende dann K' für OTP: $C := M \oplus K'$
  - Ziel: Pseudozufall $\mathrm{G}(K)$ soll „aussehen“ wie echter Zufall
- Gesucht: geeignete Funktionen G
  - Wunsch: G soll guter Zufallsgenerator sein
- **Frage:** kann so Sicherheit (im selben Sinne wie beim OTP) gegen unbeschränkte Angreifer erzielt werden?

Struktur von Stromchiffren

- Verfahren hat internen Zustand $K \in \{0,1\}^k$
- In jedem Schritt Ausgabe und Zustandsupdate:

$$\mathrm{SC}(K) := (b, K') \in \{0,1\} \times \{0,1\}^k$$

- Extrahiere Bitfolge $\mathrm{G}(K) := (b^{(1)}, \ldots, b^{(n)})$ aus K:

$$K^{(0)} := K$$
$$(b^{(i+1)}, K^{(i+1)}) := \mathrm{SC}(K^{(i)})$$

- Setze $C := M \oplus \mathrm{G}(K)$

Beispiel: Schieberegister (LFSRs)

- Schlüssel K bitweise in Speicherzellen angeordnet:

K_1	K_2	$\ldots$	K_k

- Bei Zustandsupdate wird neues Bit erzeugt ($\alpha_i \in \{0, 1\}$):

K_1	K_2	$\ldots$	K_k
$\downarrow \cdot\alpha_1$	$\downarrow \cdot\alpha_2$	$\ldots$	$\downarrow \cdot\alpha_n$

$$\longrightarrow \quad K_{k+1} := \sum_{i=1}^{n} \alpha_i K_i \bmod 2$$

- Ausgabe ist $b := K_1$, neuer Zustand ist:

K_2	$\ldots$	K_k	K_{k+1}

- Historisch interessant, aber unsicher (selbst wenn α_i geheim, siehe Vorlesung „symmetrische Verschlüsselungsverfahren“)
- LFSRs keine guten **Zufallsgeneratoren** (formalisierbar)

Zusammenfassung Stromchiffren

- Schnell (besonders in Hardware) implementierbar
- Gängige Konstruktionen mittels (mehrerer) LFSRs
- Oft algebraische Angriffe möglich
- Wegen Schlüsselupdate Synchronisation nötig
- Wie bei OTP Chiffrate verwundbar