Kryptographie

Ingo Blechschmidt
<iblech@web.de>

Augsburger Linux-Infotag
26. März 2011

Münzwurf über Telefon

- Kontext:
  Alice und Bob telefonieren.
  Sie müssen entscheiden, welcher von ihnen eine unliebsame Aufgabe übernimmt.

Münzwurf über Telefon

- Kontext:
  Alice und Bob telefonieren.
  Sie müssen entscheiden, welcher von ihnen eine unliebsame Aufgabe übernimmt.

1. Alice wählt Kopf, Bob wählt Zahl.
2. Alice wirft eine Münze.
3. Alice teilt Bob mit, dass die Münze Zahl anzeigt.
4. Bob muss die Aufgabe übernehmen.

Münzwurf über Telefon

- Kontext:
  Alice und Bob telefonieren.
  Sie müssen entscheiden, welcher von ihnen eine unliebsame Aufgabe übernimmt.

1. Alice wählt Kopf, Bob wählt Zahl.
2. Alice wirft eine Münze.
3. Alice teilt Bob mit, dass die Münze Zahl anzeigt.
4. Bob muss die Aufgabe übernehmen.

- Offensichtlich: Alice kann betrügen!

Inhalt

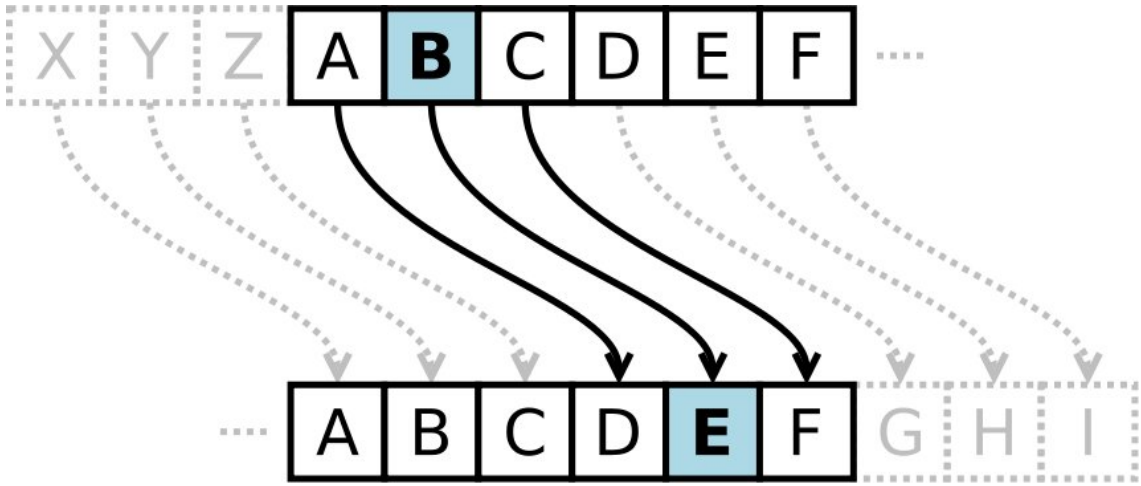
Motto

~~Sicherheit durch Unklarheit~~

- Stattdessen: Sicherheit durch Verlass auf mathematisch schwierige Probleme
- Formale Nachweismethoden
- Öffentliche Spezifikationen

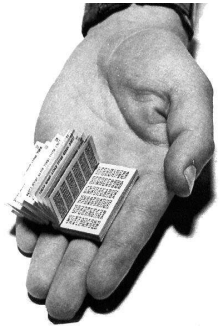
Caesar-Verschlüsselung

- Verschlüsselung durch Rotation des Alphabets
- Beispielklartext: `Linux macht Spass!`
  Verschlüsselung: `OlqxA pdfkw Vsdvv!`
- Unsicher:
  Nur 25 Möglichkeiten für die Rotation,
  außerdem Häufigkeitsanalyse möglich



Einmalverschlüsselung

- Verschlüsselung durch Addition, Entschlüsselung durch Subtraktion eines geheimen Schlüssels



- Beispielklartext: `Linux macht Spass!`
  Schlüssel: `GeheimerSchluessel`
  Verschlüsselung: `RMUYf QRuJa MTSkW!`

Einmalverschlüsselung

- Verschlüsselung durch Addition, Entschlüsselung durch Subtraktion eines geheimen Schlüssels
- Beispielklartext: `Linux macht Spass!`
  Schlüssel: `GeheimerSchluessel`
  Verschlüsselung: `RMUYf QRuJa MTSkW!`
- Sicher genau dann, wenn Schlüssel nur einmal verwendet werden und und nur die Kommunikationspartner die Schlüssel kennen.
- Problem: Wie Schlüsselaustausch bewerkstelligen?

Einwegfunktionen

Definition (verkürzt)

Eine Rechenvorschrift H heißt genau dann *Einwegfunktion*, wenn es sehr schwierig ist, zu gegebenem Funktionswert y eine Stelle x mit $H(x) = y$ zu finden.

- Beispiele: Name $\longmapsto$ Telefonnummer
  Text $\longmapsto$ SHA-256-Hash
- kein Beispiel: Buch $\longmapsto$ ISBN

- zusätzliche Forderung: Kollisionsresistenz

Münzwurf über Telefon (Forts.)

- Vereinfachung:
  Alice und Bob sitzen an einem Tisch.
  Sie müssen entscheiden, welcher von ihnen eine unliebsame Aufgabe übernimmt.

Münzwurf über Telefon (Forts.)

- Vereinfachung:
  Alice und Bob sitzen an einem Tisch.
  Sie müssen entscheiden, welcher von ihnen eine unliebsame Aufgabe übernimmt.

1. Alice nimmt eine Münze und legt sie unter eine Tasse. Bob weiß nicht, welche Seite nach oben zeigt.
2. Bob entscheidet sich für Kopf oder Zahl.
3. Alice deckt die Tasse auf.

Münzwurf über Telefon (Forts.)

- Vereinfachung:
  Alice und Bob sitzen an einem Tisch.
  Sie müssen entscheiden, welcher von ihnen eine unliebsame Aufgabe übernimmt.

1. Alice nimmt eine Münze und legt sie unter eine Tasse. Bob weiß nicht, welche Seite nach oben zeigt.
2. Bob entscheidet sich für Kopf oder Zahl.
3. Alice deckt die Tasse auf.

- Kein Zufall, Alice kontrolliert die Münze!
- Sicherheit durch gezwungene Festlegung

Digitale Imitation

1. Alice entscheidet sich für Kopf (oder Zahl) und denkt sich ein Passwort aus:
   $M :=$ `GeheimesPasswort, Münze zeigt Kopf`
2. Alice berechnet eine Einwegfunktion und teilt Bob das Ergebnis mit:
   $H(M) =$ `ae30d422b3270dd66612c56637`
3. Bob entscheidet sich für Kopf oder Zahl und teilt Alice seine Entscheidung mit.
4. Alice teilt Bob den Text M mit.
5. Bob berechnet seinerseits $H(M)$ und überprüft so Alice' Ergebnis.

Digitale Imitation

1. Alice entscheidet sich für Kopf (oder Zahl) und denkt sich ein Passwort aus:
   $M :=$ `GeheimesPasswort, Münze zeigt Kopf`
2. Alice berechnet eine Einwegfunktion und teilt Bob das Ergebnis mit:
   $H(M) =$ `ae30d422b3270dd66612c56637`
3. Bob entscheidet sich für Kopf oder Zahl und teilt Alice seine Entscheidung mit.
4. Alice teilt Bob den Text M mit.
5. Bob berechnet seinerseits $H(M)$ und überprüft so Alice' Ergebnis.

Digitale Imitation

1. Alice entscheidet sich für Kopf (oder Zahl) und denkt sich ein Passwort aus:
   $M :=$ `GeheimesPasswort, Münze zeigt Kopf`
2. Alice berechnet eine Einwegfunktion und teilt Bob das Ergebnis mit:
   $H(M) =$ `ae30d422b3270dd66612c56637`
3. Bob entscheidet sich für Kopf oder Zahl und teilt Alice seine Entscheidung mit.
4. Alice teilt Bob den Text M mit.
5. Bob berechnet seinerseits $H(M)$ und überprüft so Alice' Ergebnis.

Digitale Imitation

1. Alice entscheidet sich für Kopf (oder Zahl) und denkt sich ein Passwort aus:
   $M :=$ `GeheimesPasswort, Münze zeigt Kopf`
2. Alice berechnet eine Einwegfunktion und teilt Bob das Ergebnis mit:
   $H(M) =$ `ae30d422b3270dd66612c56637`
3. Bob entscheidet sich für Kopf oder Zahl und teilt Alice seine Entscheidung mit.
4. Alice teilt Bob den Text M mit.
5. Bob berechnet seinerseits $H(M)$ und überprüft so Alice' Ergebnis.

Digitale Imitation

1. Alice entscheidet sich für Kopf (oder Zahl) und denkt sich ein Passwort aus:
   $M :=$ `GeheimesPasswort, Münze zeigt Kopf`
2. Alice berechnet eine Einwegfunktion und teilt Bob das Ergebnis mit:
   $H(M) =$ `ae30d422b3270dd66612c56637`
3. Bob entscheidet sich für Kopf oder Zahl und teilt Alice seine Entscheidung mit.
4. Alice teilt Bob den Text M mit.
5. Bob berechnet seinerseits $H(M)$ und überprüft so Alice' Ergebnis.

Passwortauthentifizierung

- Kontext:
  Nutzer sollen sich mittels Name und Passwort authentifizieren.
- Frage: Wie Passwörter in Datenbank speichern?

1. Naiver Ansatz: Passwörter im Klartext
2. Besser: Gehashte Passwörter
   Speichere $H(\texttt{Passwort})$ statt $\texttt{Passwort}$.
3. Empfehlung: Gesalzt-gehashte Passwörter
   Speichere $\texttt{Nonce} + H(\texttt{Nonce} + \texttt{Passwort})$.
   Kompromittierung der Datenbank nicht fatal!

Passwortauthentifizierung

- Kontext:
  Nutzer sollen sich mittels Name und Passwort authentifizieren.
- Frage: Wie Passwörter in Datenbank speichern?

1. Naiver Ansatz: Passwörter im Klartext
2. Besser: Gehashte Passwörter
   Speichere $H(\texttt{Passwort})$ statt $\texttt{Passwort}$.
3. Empfehlung: Gesalzt-gehashte Passwörter
   Speichere $\texttt{Nonce} + H(\texttt{Nonce} + \texttt{Passwort})$.
   Kompromittierung der Datenbank nicht fatal!

Passwortauthentifizierung

- Kontext:
  Nutzer sollen sich mittels Name und Passwort authentifizieren.
- Frage: Wie Passwörter in Datenbank speichern?

1. Naiver Ansatz: Passwörter im Klartext
2. Besser: Gehashte Passwörter
   Speichere $H(\texttt{Passwort})$ statt $\texttt{Passwort}$.
3. Empfehlung: Gesalzt-gehashte Passwörter
   Speichere $\texttt{Nonce} + H(\texttt{Nonce} + \texttt{Passwort})$.
   Kompromittierung der Datenbank nicht fatal!

Passwortauthentifizierung

- Kontext:
  Nutzer sollen sich mittels Name und Passwort authentifizieren.
- Frage: Wie Passwörter in Datenbank speichern?

1. Naiver Ansatz: Passwörter im Klartext
2. Besser: Gehashte Passwörter
   Speichere $H(\texttt{Passwort})$ statt $\texttt{Passwort}$.
3. Empfehlung: Gesalzt-gehashte Passwörter
   Speichere $\texttt{Nonce} + H(\texttt{Nonce} + \texttt{Passwort})$.
   Kompromittierung der Datenbank nicht fatal!

Challenge–Response

- Frage: Wie Passwörter an Server übertragen?

1. Naiver Ansatz: im Klartext. . .
2. Besser: verschlüsselt (wie?)
3. Empfehlung: Challenge–Response

Challenge–Response

- Frage: Wie Passwörter an Server übertragen?

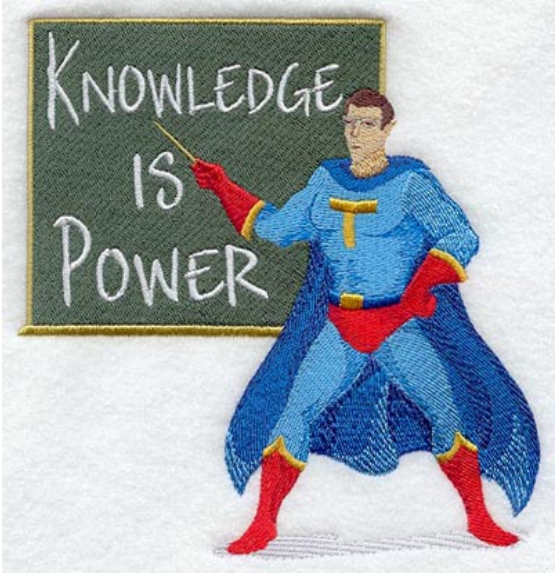
1. Naiver Ansatz: im Klartext...
2. Besser: verschlüsselt (wie?)
3. Empfehlung: Challenge–Response

1. Server → Client:
   `zufällige Nonce` + `Salz vom Passwort`
2. Client → Server:
   $H(\texttt{Nonce} + H(\texttt{Salz} + \texttt{Passwort}))$

- Immunität gegen Wiedereinspielungsangriffe!

Zero-Knowledge-Beweise

- Kontext:
  Alice möchte Bob davon überzeugen, dass sie ein bestimmtes Geheimnis kennt, ohne das Geheimnis preiszugeben.
- Illustration: Wo ist Waldo?



Zero-Knowledge-Beweise

- Kontext:
  Alice möchte Bob davon überzeugen, dass sie ein bestimmtes Geheimnis kennt, ohne das Geheimnis preiszugeben.
- Illustration: Wo ist Waldo?

- Variante: Alice und Bob möchten überprüfen, ob sie beide dasselbe Geheimnis kennen, ohne es preiszugeben.

Siehe auch

- obligatorische Wikipedia-Artikel
- Off-the-Record Messaging
- Kurt Stenzel et al:
  Vorlesung Software- und Systemsicherheit
- Mark Jason Dominus:
  You Can't Get There From Here
- Ross Anderson und Roger Needham:
  Programming Satan's Computer

Siehe auch

- obligatorische Wikipedia-Artikel
- Off-the-Record Messaging
- Kurt Stenzel et al:
  Vorlesung Software- und Systemsicherheit
- Mark Jason Dominus:
  You Can't Get There From Here
- Ross Anderson und Roger Needham:
  Programming Satan's Computer

Danke!

Siehe auch

- obligatorische Wikipedia-Artikel
- Off-the-Record Messaging
- Kurt Stenzel et al:
  Vorlesung Software- und Systemsicherheit
- Mark Jason Dominus:
  You Can't Get There From Here
- Ross Anderson und Roger Needham:
  Programming Satan's Computer

Fragen?

Bonusfolien

Mögliche Forderungen

- **Vertraulichkeit**
  Nur der Empfänger soll die Nachricht lesen können.
- **Integrität**
  Manipulationen der Nachricht sollen erkennbar sein.
- **Authentizität**
  Die Urheberschaft soll für den Empfänger überprüfbar sein.

Mögliche Forderungen (Forts.)

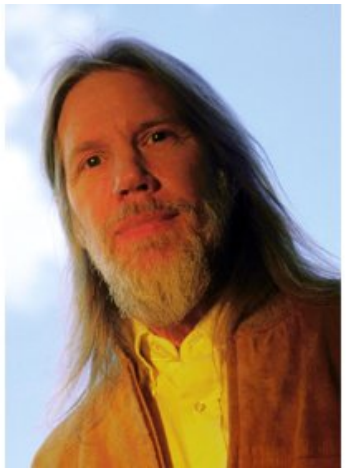
- **Vertraulichkeit**
- **Integrität**
- **Authentizität**

- **Nichtabstreitbarkeit**
  Der Absender soll seine Urheberschaft nicht abstreiten können.
- oder auch: **Abstreitbarkeit**

- **Perfekt fortgesetzte Geheimhaltung**
  Auch bei Kompromittierung sollen bereits verschickte Nachrichten sicher bleiben.

Diffie–Hellman

- Kontext:
  Alice und Bob wollen ohne sonstige vorherige Absprachen ein gemeinsames Geheimnis ausmachen.
- Annahme:
  Die Leitung wird abgehört, aber nicht manipuliert.



Diffie–Hellman (Forts.)

1. Fest: p Primzahl, g Primitivwurzel modulo p
2. Alice und Bob erzeugen je eine Zufallszahl, a bzw. b.
3. Alice $\rightarrow$ Bob: $A :\equiv g^a \mod p$
   Bob $\rightarrow$ Alice: $B :\equiv g^b \mod p$
4. Alice und Bob berechnen das Geheimnis:
   Alice: $K :\equiv B^a \mod p$
   Bob: $K :\equiv A^b \mod p$

- Gleiche Ergebnisse K!
- Sicherheit beruht i. W. auf der Schwierigkeit diskreter Logarithmen

Bildquellen

- http://biblioragazzi.files.wordpress.com/2008/04/reference.jpg
- http://i34.tinypic.com/51ptu0.jpg
- http://imgs.xkcd.com/comics/code_talkers.png
- http://one-time-pad.tripod.com/otp.jpg
- http://upload.wikimedia.org/wikipedia/commons/2/2b/Caesar3.svg
- http://www.bryx.de/wp-content/uploads/2008/09/800px-zeichen_220svg.png
- http://www.cellphones.ca/news/upload/2008/09/knowledge1.jpg
- http://www.gpuri.com/images/213/21325.jpg
- http://www.hirt-institut.de/de/Media/Shop/CategoryTextMedia/hirt_motiv_ihre_ziele.jpg
- http://www.kveller.com/images/Article_images/wheres_waldo.jpg
- http://www.marketoracle.co.uk/images/coin-toss.jpg
- http://www.treachery.net/images/why_security_through_obscurity_isnt.jpg
- http://www.waleed-security.com/wp-content/uploads/2008/11/bruceab.jpg