

\int Skripte

Kryptologie ICS.KRYPTO

Folien zur Präsenz 12 «Einführung in Kryptographische Protokolle», FS 24, V6.2



© Goscinny/Uderzo, Grosser Asterix-Band X, „Asterix als Legionär“, EHAPA-Verlag GmbH, Stuttgart, 1973, p. 41

*„Ich denke, also bin ich“

©Josef Schuler, dipl. math., dipl. Ing. NDS ETHZ, MSc Applied IT-Security, Feldhof 25, 6300 Zug, j.schuler@bluewin.ch resp. josef.schuler@hslu.ch

EINLEITUNG



Inhaltsübersicht und Agenda

● ***Einführung in (Sicherheits-)Protokolle***

- Einführung, Definitionen und Übersicht
- Authentisierprotokolle
- Schlüsselaustauschprotokolle
- Ausblick auf weitere Protokollarten (Zero Knowledge u.a.)

● ***Lernziele:***

- Ich erhalte einen Überblick über das weitläufige Thema «Kryptographische Protokolle».
- Der Aspekt der Horizonterweiterung ist grundsätzlich wichtiger als das spezifische Vermitteln von detaillierten Algorithmen resp. Protokollen.
- Ich kann die Grundbegriffe der Protokolle aufzählen.
- Ich kann wichtige Protokolle (z.B. Mutual Authentication) aufzeichnen.
- Ich kann die Grundprinzipien des Key Establishment aufzählen.
- Ich habe einen Eindruck erhalten, wie man Protokolle analysiert.

Die Slogan zu dieser Präsenz

Titelbild = Parallel Session Attacke nach Gallier Art.

Verweise zur Literatur

- **Für die Theorie:**

- JS Skript „Einführung in die Kryptologie“, Kap. 3.2.2, 10.6.2, 12, 13 & 16.5.
- In [CP-D], Kap. 13.

- **Aufgaben**

- Aufgaben in den Folien.
- JS Skript „Aufgaben und Lösungen zum Modul KRYPT an der HSLU-I“, Kap. 2.6, „Aufgaben zur Präsenz PR 12“.

- **Beachten Sie, dass...**

- ... die vorliegenden Folien für das Thema der PR 12 die Hauptliteratur sind. Die Kapitel 12 & 13 in JS Skript „Einführung in die Kryptologie“ und in [CP-D] sind eher eine Ergänzung und müssen nicht notwendigerweise im Detail durchgearbeitet werden. Insbesondere in [CP-D] hat es aber einige Aspekte, die wir schon in anderen Präsenzen betrachtet haben.
- ... viele Aspekte schon im Modul ISF resp. in meinen Präsenzen behandelt wurden. Daher werden einige Folien nur kurz besprochen.

Einführung in Protokolle

Einleitung Protokolle

Ein Protokoll ist nichts anderes als ein festgelegter Ablauf.

- In unserem täglichen Leben halten wir uns oft an festgelegte Abläufe (ggf. Rituale), wie z.B.
 - Begrüssung
 - Ablauf beim Benzin tanken
 - Tanken mit anschliessendem Zahlen im Shop
 - Tanken an einer Säule ohne bedienten Shop
- Netzwerkprotokolle, z.B.
 - http oder https
 - ftp, usw.
- Sicherheitsprotokolle, z.B.
 - SSL, TLS, IPSec
 - Kerberos, usw.
- Krypto-Protokolle sind spezifische Sicherheitsprotokolle, z.B.
 - Authentisierprotokolle
 - Schlüsselaustauschprotokolle, usw.

Einleitung Protokolle, Fortsetzung

- Protokolle sind ein ganzes Thema für sich. So gibt es im (Fern-) Masterstudiengang Applied IT Security der Ruhr-Uni Bochum drei 10 ETCS Vorlesungen zu den Protokollen:
 - Ein Modul zu Netzwerkprotokolle wie SSL und IPSec.
 - Ein Modul zur Theorie und Analyse von Protokollen.
 - Ein Modul zu weiterführenden Protokolle wie Secure Multi-Party Computation, Zero Knowledge, Bit Commitment usw.
- Wir haben schon mit Krypto-Protokollen zu tun gehabt, z.B.
 - Wir haben 2 Angriffe kennengelernt, die nur mit Einbezug von Protokollen verhindert werden können (*).
 - Das Diffie-Hellman Schlüsselaustausch Protokoll (DHKE).
 - Das Elgamal Verschlüsselungsverfahren (EEP).
 - Doppelunterschriften (unterschiedliche Abläufe bei der additiven und bei der multiplikativen Aufteilung des Secret Keys).
 - Blinde Signaturen beim Erstellen von anonymen digitalem Geld.

Aufgabe 1: Zu (*): Welche Angriffe sind das?

Definition «Protokoll» und weitere Begriffe

- Unter einem **Protokoll** versteht man eine streng geregelte Abfolge von Schritten (sprich: Meldungen), um ein vordefiniertes Ziel damit zu erreichen.
 - Ein **kryptographisches Protokoll** ist demzufolge ein Protokoll, welches auf kryptographischen Grundbausteinen beruht und **vordefinierte** Sicherheitsziele erreichen will.
 - Authentisierprotokolle → (praktisch ausschliesslich) Challenge-Response (= Anfrage – Antwort) Protokolle
 - Schlüsselaustauschprotokolle (z.B. Diffie-Hellman)
 - Commitment Protokolle (Meinung festlegen)
 - Protokolle zur sicheren Berechnung (SMPC = Secure Multi-Party Computation)
 - Protokolle, um etwas zu beweisen (Aussage, Wissen, Identität) → in der Regel Zero Knowledge Protokolle.
 - Ein „normales“ **Sicherheitsprotokoll** hat ebenfalls eine erhöhte Sicherheit zum Ziel und ist oft eine Kombination von kryptographischen wie anderen Protokollen resp. Protokollteilen.
 - SSL/TLS
 - IPSec

Übersicht zu den kryptographischen Protokollen

- Authentisierprotokolle
 - Authentisierung von Usern, Servern usw.
 - Challenge-Response (C-R) Protokolle: Eine „Frage“ in Form eines Randomwertes muss „beantwortet“ (z.B. signiert) werden.
- Key Establishment (Schlüssel erzeugen und verteilen).
 - Key Transport Protokolle wie Kerberos
 - Eine Partei erzeugt den Key und verteilt sie den anderen.
 - Key Agreement Protokolle
 - Alle Parteien tragen zum Geheimnis resp. Schlüssel bei.
 - Zu beiden Typen gibt es diverse Untertypen, die wir hier nicht näher betrachten.
- Weitere Protokolltypen (werden ansatzweise betrachtet)
 - Commitment Protokolle: Protokolle, wo die Teilnehmer eine gefasste Meinung oder Angebot unveränderbar und versteckt hinterlegen.
 - Beweisprotokolle (z.B. zero knowledge proof Protokolle).
 - Sichere Berechnungen, als Ersatz für eine vertrauenswürdige, unabhängige Instanz (SMPC).
- Zu allen Protokollarten gibt es div. Untertypen.

Einbettung der Protokolle in die Kryptologie

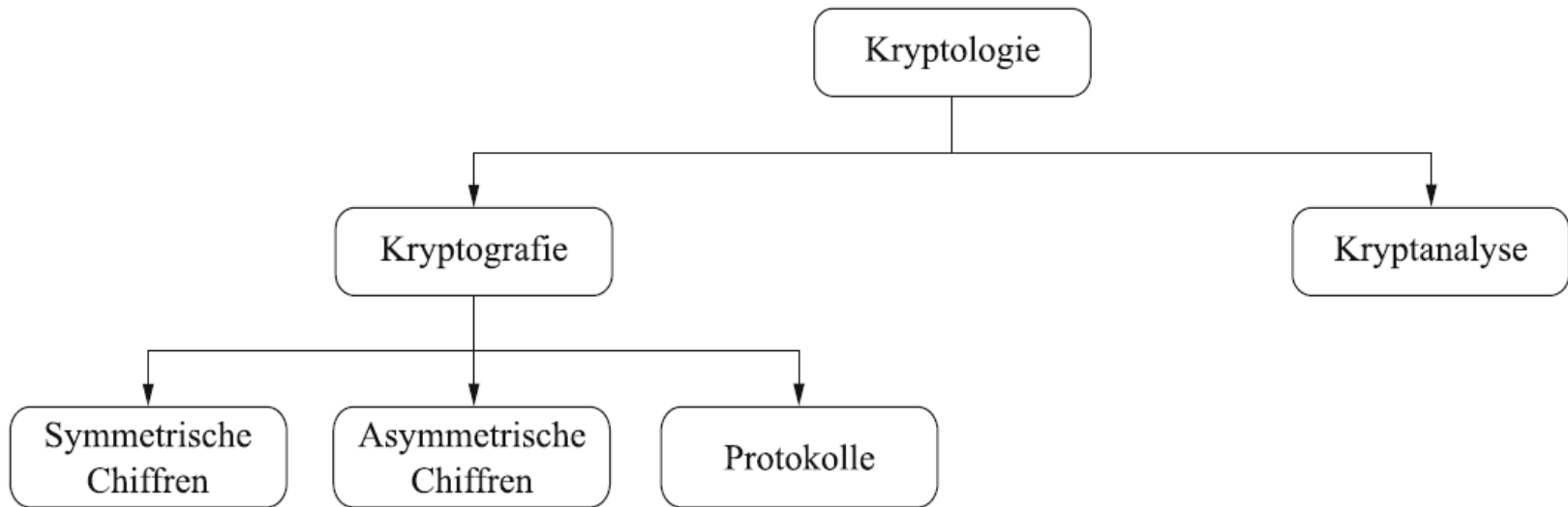


Abb. 1.3 Die Kryptologie und ihre Untergebiete

- Siehe Lehrbuch, C. Paar, S. 3.
- C. Paar versteht hier unter „Protokolle“ kryptographische Protokolle, schliesst aber Sicherheitsprotokolle wie TLS ebenfalls mit ein.
- **Wichtig** zu wissen ist, dass unter „Protokoll“ je nach Anwendungsgebiet nicht immer Identisches verstanden wird!! Die reinen Kryptologen-Innen verstehen etwas anderes als die Netzwerttechniker-Innen. In diesem Sinne soll man sich zuerst vergewissern, um was es nun bei einer Diskussion gehen soll. Ansonsten sind Missverständnisse vorprogrammiert.

Die Entwicklung von kryptograph. Protokollen

- Zu Beginn hat man (z.B.) Authentisier-Protokolle definiert. Man befand diese als ziemlich sicher, doch dann fand man immer nachträglich irgendwelche Schwächen.
- Diese Entwicklung versuche ich weiter hinten zu demonstrieren.
- Diesen Mangel hat man inzwischen soweit bereinigt, dass Eigenschaften und Ziele von Protokollen standardisiert wurden.
 - Viele neue Eigenschaften wurden in diesem Zusammenhang definiert.
 - Bei der Definition von einem Protokoll gibt man an, welche Eigenschaft dieses Protokoll erfüllen soll.
 - Erfüllt ein Protokoll eine Eigenschaft definitionsgemäss nicht, so kann man zwar i.d.R. Angriffe gegen diese Eigenschaft finden. Aber man beurteilt das nicht als „geknackt“, denn das Protokoll erfüllt per Definition diese Eigenschaft nicht.
 - Diese Themen müssten in einem Nachfolgemodul tiefer besprochen werden.

Beispiele von allg. Protokolleigenschaften

- In Commitment Protokollen sind **Binding** (eine einmal festgelegte Meinung oder festgelegtes Angebot kann später nicht mehr geändert werden) oder **Hiding** (die Meinung resp. das Angebot ist versteckt) solche neuen Eigenschaften.
- Bei E-Cash Protokollen wünscht man sich **Anonymität** (die Bank soll nicht wissen, von wem das Geld abgehoben wurde und für was es ausgegeben wurde) und **Unlinkbarkeit** (die Bank soll keinen Bezug zwischen verschiedenen Zahlungen machen können) → Siehe blinde Signaturen in PR 10.
- Bei Schlüsselaustauschprotokollen gibt es eine ganze Reihe von Eigenschaften (z.B. Key Freshness, Key Authentication, (Perfect) Forward Secrecy usw.) die man erfüllt haben möchte → siehe dazu später weiter hinten. Dabei werden wir nicht weit in die Tiefe gehen.

Angreifer und Angriffe auf Protokolle

- Angreifer und Bedrohungen
 - Alle Kanäle und damit alle Meldungen können abgehört werden (passive Angreifer).
 - Ein aktiver Angreifer kann Meldungen verändern, umleiten, löschen, neue Meldungen hinzufügen usw.; es sind „keine“ Grenzen gesetzt.
 - Ein Angreifer kann ein legitimer Teilnehmer oder ein Outsider sein.
 - Ein Angreifer ist z.B. fähig ...
 - ... den Session Key einer alten, vergangenen Session zu erhalten.
 - ... eine falsche Authentizität vorzutäuschen (Masquerade).
 - ... Angriffe auf die Schlüssel zu machen (Wiederverwenden von Schlüsseln; falsche Schlüssel einzuschleusen usw.), also die Authentizität der Schlüssel anzugreifen oder zu verändern.
- Neue und vielfältige Angriffe entstehen, resp. entstanden, z.B.:
 - Man in the middle Attack bei Diffie-Hellman
 - Parallel Session & Oracle Session Attacke bei Authentisierprotokollen.
 - Replay Attacken und viele weitere.

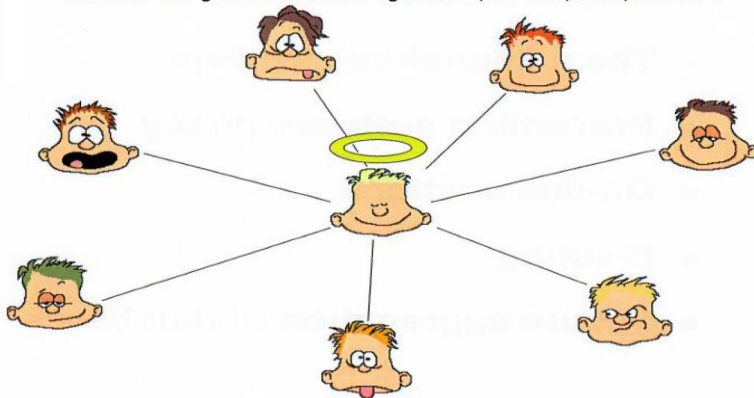
Secure Multi-Party Computation Protokoll

Grundsätzlich:

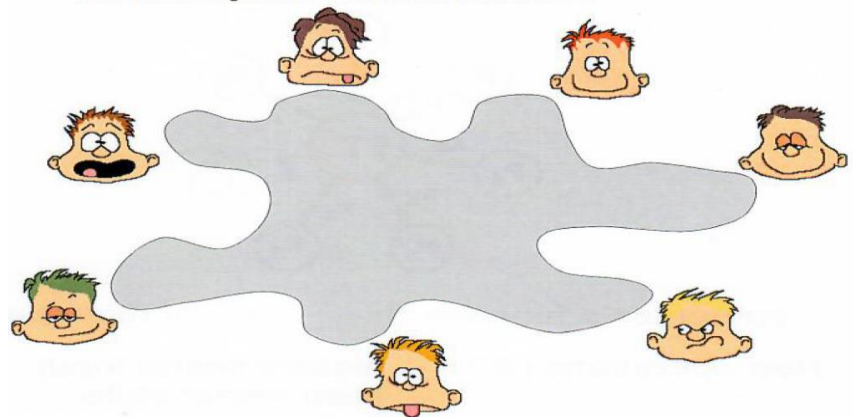
SMPC (Secure Multi-Party Computation) will ein Protokoll liefern, so dass mehrere Teilnehmer Berechnungen miteinander ausführen können, ohne sich gegenseitig zu vertrauen (d.h. ohne die eigenen Angaben bekannt zu geben).

Grundprinzip: Die vertrauenswürdige Instanz wird durch ein Protokoll ersetzt.

Klassische Lösung mit vertrauenswürdiger Instanz, z.B. Urne, Notar, Börse u.ä.



Vertrauenswürdige Instanz wird durch Protokoll ersetzt



Bekannte aktuelle Anwendungen, z.B.

- Anwendungen der Blockchain-Technologie (z.B. Bitcoin oder Aktienhandel zwischen zwei Personen ohne Bank dazwischen → möglich seit ca. Frühling 22).
- Web of trust (in PGP) als Alternative zu einer Certification Authority bei einer PKI.
- Elektronische Wahlen o. Auktionen (z.B. Ersatz der klassischen Urne/Börse).
- Secret Sharing, also Aufteilen von einem Geheimnis resp. Schlüssel.

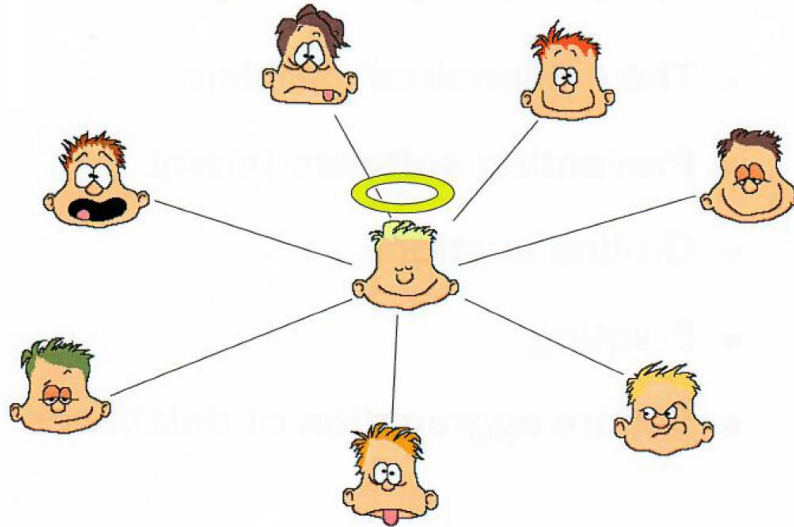
Heute klafft aber noch eine grosse Lücke zw. Forschung und Implementation!

Beispiel eines SMPC Protokolls, cf. Kap. 3.2.2

SICHERE MULTI-PARTY BERECHNUNGEN (Secure MPC)

- n Personen P_1, P_2, \dots, P_n wollen den Durchschnitt ihrer Löhne berechnen, ohne einander die jeweiligen Saläre mitzuteilen.
- **Klassische Lösung, mit Einsatz einer vertrauenswürdigen Instanz.**
 - Alle geben einer (ausenstehenden) Person (= vertrauenswürdige Instanz) ihr Salär bekannt. Diese Person berechnet den Durchschnittslohn.

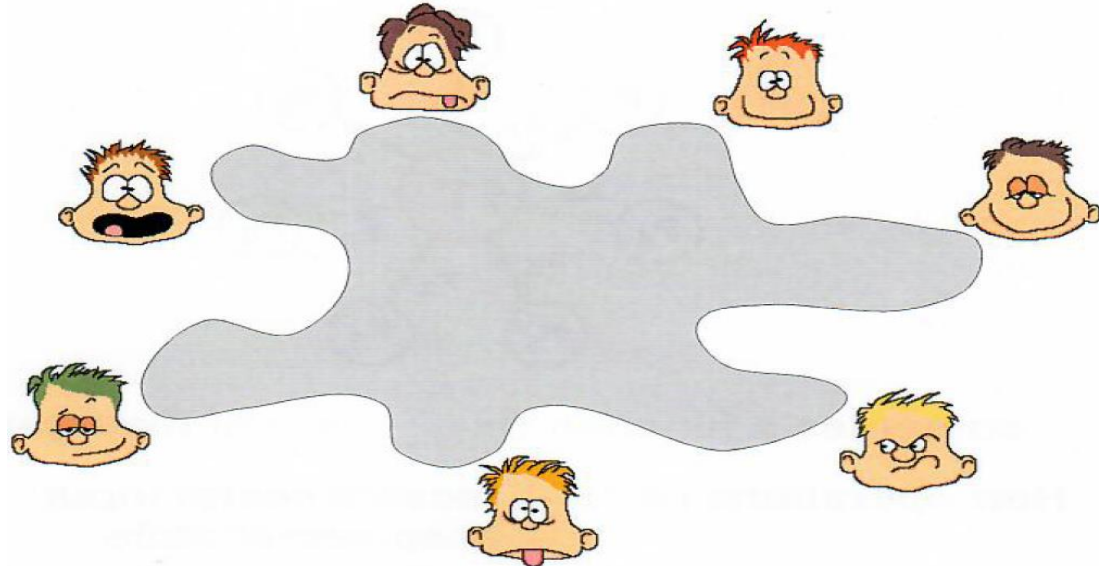
Klassische Lösung mit vertrauenswürdiger Instanz, z.B. Urne, Notar, Börse u.ä.



Beispiel eines SMPC Protokolls, Fort.

- **Secure MPC, ohne vertrauenswürdige Instanz**

Vertrauenswürdige Instanz wird durch Protokoll ersetzt



- P_1 wählt eine grosse Zufallszahl und addiert sein Salär dazu.
- Die addierte Zahl reicht P_1 an P_2 weiter, dieser addiert ebenfalls sein Salär dazu.
- Das geht so weiter bis zum P_n . Dieser addiert sein Salär und reicht die Summe P_1 weiter.
- P_1 subtrahiert die ursprüngliche Zufallszahl und dividiert durch n .
- **Kurzanalyse und absolut notwendige Voraussetzung**
 - Das Protokoll soll mit seiner Einfachheit das Prinzip zeigen.
 - In diesem Protokoll können P_i und P_{i+2} zusammenspannen um den Lohn von P_{i+1} zu bestimmen.
 - **Notw. Voraussetzung:** Ehrliche & korrekte Angabe der Höhe des Salärs.

Zero-Knowledge Protokolle

Zunächst ist es erfreulich, dass ich nun eine Frau als Protagonistin erwähnen darf.



Shafi Goldwasser (*1958)

Silvio Micali (*1954)



Charles Rackoff (*1948)

Um was geht es?

Es geht darum, dass ich beweisen kann, dass ich etwas weiss, ohne dass ich Wissen abgeben muss. Das tönt paradox! Aber wenn ich Wissen preisgeben muss, dann könnte dieses Wissen ja genutzt werden.

Dazu gibt es im Prinzip zwei Beweistypen:

- Beweise von Aussagen (proofs of statements).
- Beweise von Wissen (proofs of knowledge).

Zero-Knowledge Protokolle: Beispiele & Nutzen

Beispiele zu Beweis von Aussagen:

- Eine gegebene Zahl n hat genau 3 Primfaktoren.
 - Ein möglicher Beweis für diese Aussage besteht darin, die Primfaktoren anzugeben → aber damit gebe ich Wissen ab.
- Es gibt unendlich viele Primzahlpaare – z.B. (3; 5), (5; 7), (11; 13), (17; 19)
 - Es ist noch nicht bekannt, ob die Aussage wahr oder falsch ist.
- Zwei gegebene Graphen sind isomorph.
- Ein gegebener Graph G enthält einen Hamiltonschen Kreis.

Beispiele zu Beweis von Wissen:

- Ich kenne die Faktorisierung einer 500-stelligen Zahl.
- Es sei ein Generator g und ein Moduls p sowie $y = g^x \bmod p$ gegeben. Nun behaupte ich, ich kenne das x , so dass $y = g^x \bmod p$. Usw.
- Ich weiss, ob es unendlich viele Primzahlpaare gibt oder nicht, und ich kann das beweisen, dass ich es weiss, ohne zu sagen ob es unendliche viele Primzahlpaare gibt oder nicht.

Einsatz:

- Einsatz bei Benutzeridentifikationen.
- Einsatz bei Secure MPC.

Auf weitere Details – wie konkrete Protokolle – verzichten wir!

Bit-Commitment Protokoll

Grundsätzlich:

Es geht beim *Commitment* (Festlegung) darum, dass eine Person – z.B. Alice – eine Nachricht so hinterlegt, dass sie

- von niemandem gelesen werden kann und
- von niemandem – insbesondere auch von Alice – **nicht** verändert werden kann.

Mechanische Analogon und Beispiel:

Bei einer öffentlichen Ausschreibung legt jeder Anbieter sich auf sein Angebot fest, gibt dieses aber zunächst noch nicht preis (indem er es in einem verschlossenen Umschlag abgibt). Erst nach Ende der Ausschreibungsfrist wird die Festlegung offenbart (der Umschlag wird – z.B. von einem Notar – geöffnet). So wird gewährleistet, dass während der Ausschreibung die Konkurrenten nichts über die Höhe der Angebote der anderen erfahren, und dass die Angebote verbindlich sind, d.h. nachträglich nicht mehr geändert werden können.

Einsatz bei...

... Online Poker, ... Benutzeridentifikationen, ... Festlegen eines Angebots.

Repetition Sicherheitsmechanismen

Zwei oder drei Sicherheitsanforderungen?

Siehe Aufgabe 3.1 in Präsenz 0.

Anwendung	Geheimhaltung	Authentizität/ Integrität
A) Digitales öffentliches Archiv		X
B) Pay-TV (*)	X	
C) Elektronische Unterschrift		X

(*) Beim Pay-TV sprechen wir nur die Datenintegrität, nicht aber die Benutzerauthentizität (= Berechtigungszugang zum Pay-TV) an.

Wichtig:

Der (*) gibt uns den klaren Hinweis, dass wir auch die Benutzerauthentizität prüfen müssen → Authentisier-Mechanismen!

Beispiel zur Wichtigkeit von Auth.-Mech.

(Haus-)Aufgabe 2:

Dem Umstand, dass wir zwischen der Daten- und der Benutzerauthentizität unterscheiden möchten, wollen wir nun Rechnung tragen.

Schauen Sie das Video

<https://youtu.be/XCPu16cvxkA> an, und füllen Sie dann die Tabelle aus.



Anwendung	Geheim.	Datenauth.	Benutzerauth. (**)
D) Pay-TV inkl. Zugangsberechtigung	x		x
E) Berechtigung zur Ampelsteuerung			x
F) Bargeldbezug beim Bancomat	x	x	x
G) E-Banking	x	x	x

(**) Wird in [BSI1], Kap. 6 auch Instanzauthentisierung bezeichnet.

Authentisier-Mechanismen

User Authentication, Übersicht



"On the Internet, nobody knows you're a dog."

- Username / Password
Dictionary Attacks
- One-Time Passwords
Token: SecureID, etc.
- Symmetric Algorithms
C-R-Protocols
- Public Key Algorithms
Smartcards, Certificates,
Public Key Infrastructure,
C-R-Protocols
- Biometrical Methods
Fingerprint, Iris-Scan,
Voice, Face, Hand, etc.

Pro memoria und erste Betrachtungen

- 1) Im Modul ISF haben Sie diese Themen schon intensiv betrachtet.
- 2) Im Modul KRYPTO, cf. Kap. 4.1 „Angriffe und Angreifer...“. Wir haben dort insbesondere festgehalten, dass die Masquerade **nicht** mit einem klassischen Kryptoalgorithmus verhindert werden kann. Es braucht ein Protokoll und zwar ein sogenanntes Challenge – Response oder C-R Protokoll.
- 3) Auf das Thema „Benutzer Authentisierung“ (User Authentication) wurde in der PR 4 mit der vorherigen Folie schon kurz hingewiesen.
- 4) Sie kennen ja schon div. Authentisier-Verfahren
 - a) Reine PIN Eingabe (z.B. beim Bancomat)
 - b) Mit Username und Passwort: 1–Faktor Authentisierung
 - c) Mit Username, Passwort und einem zusätzlichen (z.B. per SMS) zugestelltem Code: 2–Faktor Authentisierung (oft wird ein solches 2–Faktor Verfahren als „starke Authentisierung“ bezeichnet). Dieser Begriff suggeriert aber – im Vergleich zu einem kryptographischen Authentisierprotokoll – eine Stärke, die es nicht wirklich hat. Kurz: „zwei schwache Verfahren ergeben kein starkes“. Wobei natürlich zu definieren ist, was „stark“ und was „schwach“ bedeutet.
 - d) One time password **OTP**, RSA SecurID (synchrones Laufen von Pseudo Random Generatoren).

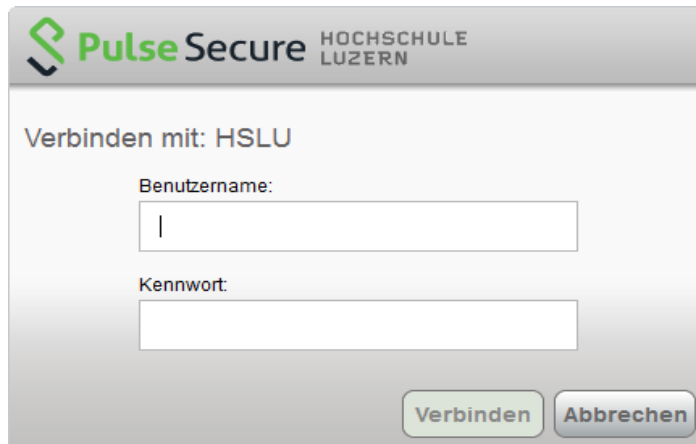
Erste Betrachtungen, Beispiel zu 4b) & 4c)

Zu 4b)



Aus Rolf Oppliger Vorlesung UNI ZH, 2020

Zu 4c) HSLU 2-Faktor Authentisierung



PulseSecure HOCHSCHULE LUZERN

Verbinden mit: HSLU

Benutzername:

Kennwort:



PulseSecure HOCHSCHULE LUZERN

Verbinden mit: HSLU

Geben Sie folgende Anmeldeinformationen an, um die Verbindung herzustellen.

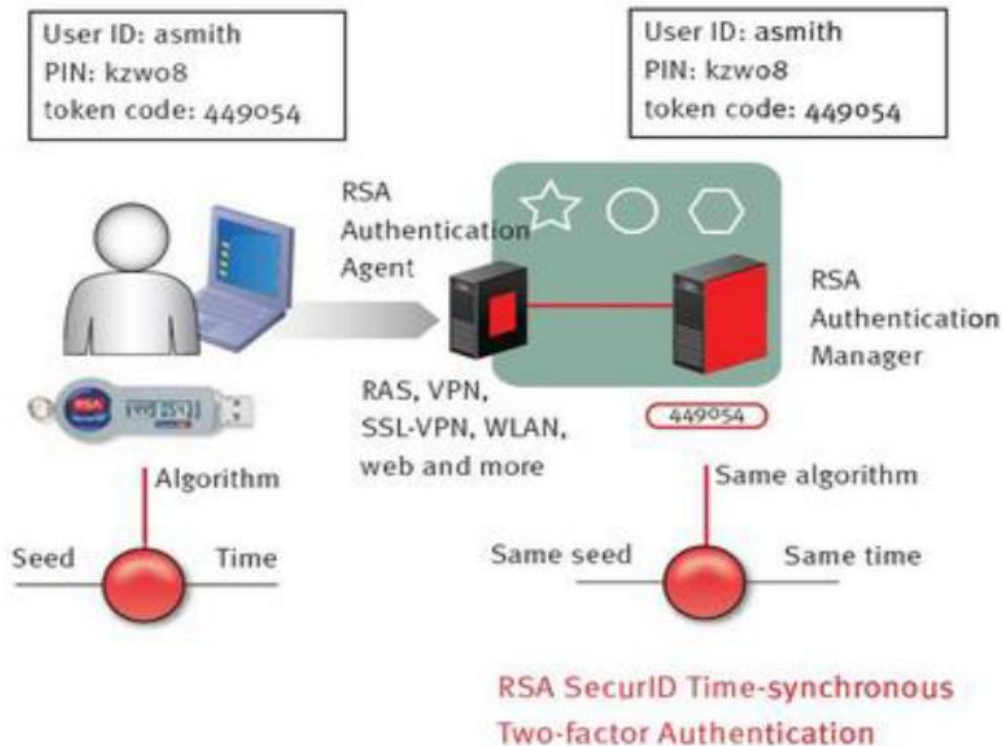
Meldung vom Server:

Geben Sie eine Antwort ein:

Erste Betrachtungen, Beispiel zu 4d)

Zu 4d) RSA SecurID

Aus Rolf Oppliger Vorlesung UNI ZH, 2020



Erste Betrachtungen, Fortsetzung

- 5) Biometrische Verfahren benutzen Sie wahrscheinlich auch, z.B. beim Handy das Fingerabdruckverfahren. Hier gibt es insbesondere zu betrachten, dass es eine «false accept rate, FAR» (Wsk., dass eine unberechtigte Person als berechtigt akzeptiert wird) und eine «false reject rate, FRR» (Wsk., dass eine berechtigte Person als unberechtigt taxiert wird). Beide Wsk. tief zu halten ist schwierig: Verringert man die eine Rate, erhöht sich i.d.R. die andere. Erschwerend kommt dazu, dass es einen grossen Unterschied macht, ob die Verifikation «one to one» oder «one to many» ist. In diesem Themenkomplex werden immer wieder Falschaussagen gemacht. In der folgenden Folie versuche ich die Problematik aufzuzeigen.



Bild: Aus Rolf Oppliger
Vorlesung UNI ZH, 2020

Die Entscheidung, ein Vergleich

Reaktion des Bancomaten bei einer PIN-Eingabe & gültigem Bezugslimit.

	PIN richtig	PIN falsch
Bancomat gibt Geld	o.k.	Kommt zu „100%“ nicht vor; FAR = 0
Bancomat gibt kein Geld	Kommt zu „100%“ nicht vor; FRR = 0	o.k.

Im Gegensatz dazu:

An der Vereinzelungsschleuse ins Rechenzentrum mit Iris-Scan:

	Berechtigter	Unberechtigter
Drehtüre öffnet sich	o.k.	False Acceptance FAR > 0
Drehtüre öffnet sich nicht	False Reject FRR > 0	o.k.

- FAR und FRR «bekämpfen» sich, d.h. man kann nicht ohne Weiteres die FAR runterschrauben ohne damit die FRR zu vergrössern, und umgekehrt.
- Welche Rate zu optimieren ist, ist i.d.R. zu definieren.
- Plakativer Extremfall 1: FAR = 0, wenn ich keinen reinlasse.
- Plakativer Extremfall 2: FRR = 0, wenn ich jeden reinlasse.

One to one = Verifikation (Anmeldung im Handy)

Verifikation = Überprüfung ob ich derjenige bin, den ich vorgebe zu sein.

Verifikation bedeutet "Bestätigung der Identität." Die Personenverifikation entscheidet die Frage, ob es sich bei einer Person um diejenige handelt, für die sie sich ausgibt. In der Biometrie werden bei der Verifikation die aktuellen biometrischen Daten einer Person erfasst und mit den im Vorfeld erfassten biometrischen Referenzdaten desjenigen Individuums verglichen, als das sich die Person ausgibt (1:1 – Vergleich). Es findet nur ein Vergleich zweier Datensätze statt. Stimmen die beiden Datensätze innerhalb der gewählten Toleranzgrenzen miteinander überein, so wird bestätigt, dass es sich bei der Person um diejenige handelt, für die sie sich ausgibt.

One to many = Identifikation (Tätersuche)

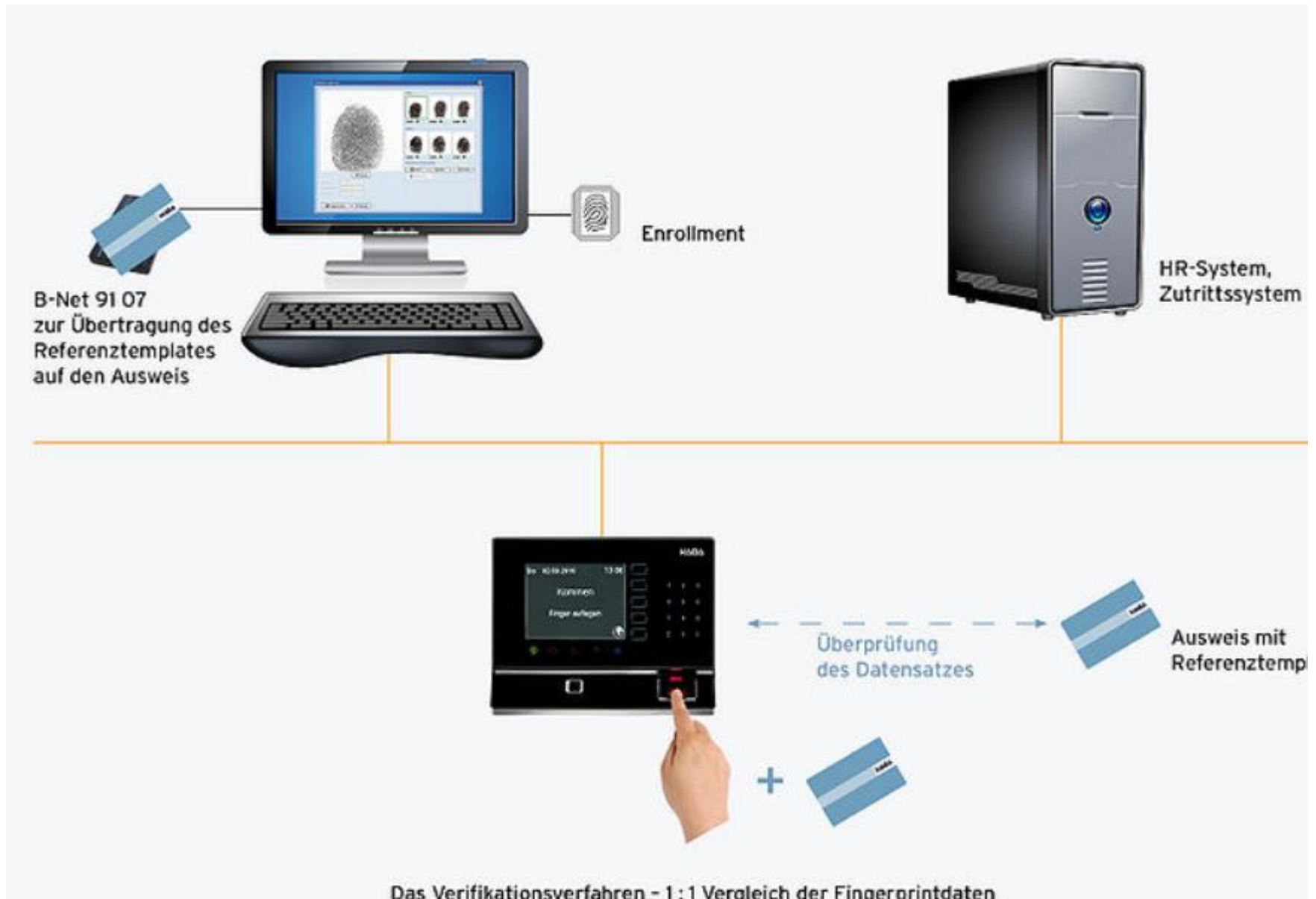
Identifikation = Sag du mir, wer ich bin.

Identifikation bedeutet "Feststellung der Identität". Bei der Personenidentifikation wird festgelegt, um welche Person es sich handelt. In der Biometrie werden bei der Identifikation die aktuellen biometrischen Daten einer Person erfasst und mit den im Vorfeld erfassten biometrischen Referenzdaten einer Vielzahl von Individuen verglichen (1:n – Vergleich). Diese Referenzdaten sind beispielsweise in einer Datenbank gespeichert. Es findet somit eine Vielzahl von Vergleichen statt. Die Person wird als dasjenige Individuum identifiziert, dessen biometrischer Referenzdatensatz mit dem aktuellen biometrischen Datensatz der Person innerhalb der gewählten Toleranzgrenzen übereinstimmt. Dabei ist nicht auszuschliessen, dass es auch mehrere solcher Personen geben kann.

Quelle: https://www.secupedia.info/wiki/Biometrische_Verfahren#ixzz6g33rCQ5l

Lizenziert unter CC-BY-SA 3.0 Germany (<http://secupedia.info/wiki/SecuPedia:Lizenz>)

One to one = 1 : 1 = Verifikation



One to many = 1 : n = Identifikation



Erste Betrachtungen: Eine Frage

- 6) Haben Sie sich auch schon bei einem Bargeldbezug am Bancomaten gefragt, ob das Gerät echt ist?

Die Frage ist berechtigt.

→ Skimming Attacke:



Lesen der Daten vom Magnetstreifen oder von der Chipkarte oder die Karte nach der PIN-Eingabe im Fake Aufbau einziehen, mit Kamera die PIN-Eingabe filmen. Danach mit der Karte oder mit einem Duplikat Bezüge tätigen. **Trick bei Bancomatbezügen, z.B. im Ausland: Zunächst eine falsche PIN eingeben. Falls nun die Karte eingezogen wird → dann passiert «nichts». Natürlich nicht machen, wenn Ihr Retry schon auf 1 gesetzt ist.**

Wir kommen nun zum Kern des Themas: Die gegenseitige Authentisierung resp. mutual Authentication. Was offensichtlich beim Bancomat nicht passiert. Der Bancomat authentisiert sich nicht!!!

Bemerkung: Der Betrug mit Steckenbleiben der Karte im Aufsatz funktioniert, weil im Aufsatz eine gesperrte Karte steckt. Sie wird erst nach der Rückantwort vom Host eingezogen. D.h. die Aufforderung zur PIN-Eingabe erfolgt, da der Bancomat zu diesem Zeitpunkt noch nicht weiss, dass die Karte gesperrt ist.

Authentisier-Protokolle



2 Grundprobleme in offenen Netzwerken

- Wie erkennen sich zwei Partner gegenseitig? (Mutual Authentication)
 - Wie kann man auf sichere Art und Weise Schlüssel austauschen? (Key Distribution)
- (1) Man kann das Eine problemlos lösen, wenn das Andere schon ausgeführt wurde. (D.h. nach einer Mutual Authentication kann problemlos eine Schlüsselverteilung sicher gemacht werden. Resp. umgekehrt: nach einer Schlüsselverteilung kann zu einem späteren Zeitpunkt eine Mutual Authentication gemacht werden.)
 - (2) Es können darauf aufbauend alle weiteren Sicherheitsdienste (z.B. Vertraulichkeit, Meldungsauthentizität usw.) aufgebaut werden.
 - (3) Nur kann man nicht, das eine mit dem anderen und gleichzeitig das andere mit dem einen lösen.
 - (4) Das Grundproblem ist zunächst die erstmalige (symmetrische) Schlüsselverteilung.
 - (5) Nachdem (4) gelöst ist, werden Authentisierungs- und Schlüsselverteilschritte ineinander verschachtelt durchgeführt.

Aus didaktischen Gründen behandeln wir in diesem Kapitel nur das Problem der gegenseitigen Authentifikation (mutual Authentication). Die Verteilung der Schlüssel wird im nächsten Kapitel betrachtet.

Wichtige Annahmen

Im Folgenden wollen wir gewisse Netzwerkattacken durchführen. Damit das Grundlegende dieser Attacken gezeigt werden kann, müssen wir gewisse Annahmen vorgeben:

- i. **Alice und Bob haben vorgängig Schlüssel ausgetauscht.**
- ii. **Alice und Bob sind Computer „ohne Gedächtnis“.** D.h. sie wissen nicht mehr, was eine Session vorher passiert ist.
- iii. **Eve kann eine authentifizierte Session nicht übernehmen.**

Frage:

Warum diese Annahmen?

Antwort:

- Das Problem mit dem Grundschlüssel K haben wir so gelöst.
- Z.B. mit applikatorischen Massnahmen – wie Einbezug einer Sequenznummer in die verschlüsselten Daten – könnten gewisse Attacken eingedämmt werden.
- In einem konkreten Protokoll werden Schlüssel zur Sicherung des weiteren Meldungsflusses mitgeschickt, cf. (5) in der vorherigen Folie.

Ohne diese Annahmen i. – iii. würden wir Gefahr laufen, dass wir uns in (unnötige) Diskussionen verlieren würden und so unser Ziel, das Grundlegende der Attacken kennenzulernen, aus dem Auge verlieren!

Wir wollen auch nicht z.B. applikatorische Massnahmen als Lösungen propagieren, sondern ausschliesslich kryptographische betrachten.

Diskussion: «Ist dieses Protokoll sicher?»

Beispiel 1: Was sagen Sie zur Sicherheit von diesem Protokoll?



Alice

Hallo, ich bin Alice, beweise mir, dass du B bist.



Hier hast du mein X.509 Zertifikat, damit beweise ich dir, dass ich Bob bin.



Bob

Bemerkung:

Anstatt «Zertifikat schicken» (hat ja zumindest einen kryptographischen Touch), könnte man auch «Passwort schicken» einfügen, o.ä.

Beispiel 2: Ist/war ein Standard zur gegenseitigen Authentisierung bei der Chipkartenherstellung zw. Chipkarte und einer Spezial Hardware

- Alice = Chipkarte, resp. eindeutige Chipkartennummer = $\text{CHIP_ID} = A$
- Bob = Spezial Hardware, resp. deren ID
- K = der gemeinsame symmetrische Schlüssel, wurde in einem Vorgängerprozess in den Chip eingebrannt, Bob kann ihn aus der CHIP_ID herleiten.
- R_A = Zufallswert von Alice, «R» steht für Random.
- $\text{ENC}(M, K)$ = (Block-)Verschlüsselung der Meldung M mit dem symmetrischen Schlüssel K .

Alice mit sym. Schlüssel K



Bob mit sym. Schlüssel K



„Ich bin Alice“, R_A

$C_A = \text{ENC}(R_A, K)$

?

Alice entschlüsselt C_A und überprüft $R_A \stackrel{?}{=} \text{DEC}(C_A, K)$

Kritik & Bemerkung:

- Das Protokoll ist sicher nicht gegenseitig, es authentisiert sich nur Bob gegenüber Alice. Bob weiss nicht, ob sich wirklich Alice gemeldet hat.
- Um eine (spätere) Übernahme des Meldungsflusses durch Eve zu verhindern, werden die Authentisierungsschritte mit Schlüsselverteilschritten kombiniert.

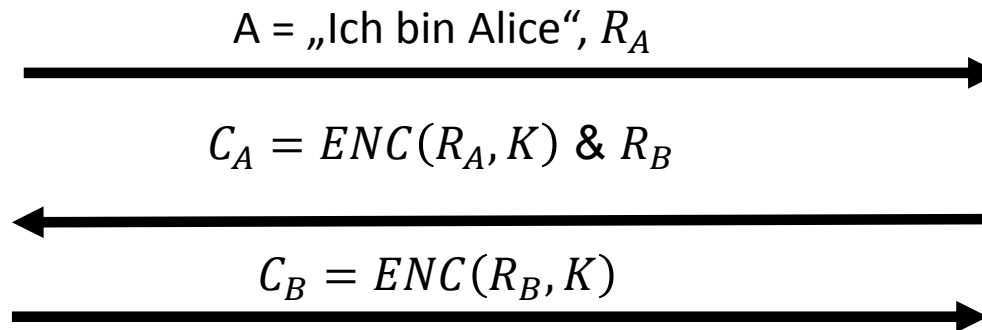
Beispiel 2, Verbesserung 1: Bob schickt auch einen Zufallswert, Alice muss diesen ebenfalls verschlüsseln.

- A = Identifikationsnummer von Alice
- B = Identifikationsnummer von Bob
- K = der gemeinsame symmetrische Schlüssel von Alice und Bob, er wurde (irgendwie) vorgängig ausgetauscht.
- R_A = Zufallswert von Alice, R_B = Zufallswert von Bob, «R» steht für Random.
- $ENC(M, K)$ = (Block-)Verschlüsselung der Meldung M mit dem symmetrischen Schlüssel K.

Alice mit sym. Schlüssel K



Bob mit sym. Schlüssel K



- Alice entschlüsselt C_A und überprüft $R_A \stackrel{?}{\cong} DEC(C_A, K)$
- Alice verschlüsselt R_B und schickt C_B

Bob entschlüsselt C_B und überprüft $R_B \stackrel{?}{\cong} DEC(C_B, K)$

«Mutual» ist nun gegeben, und es ist alles paletti? Oder doch nicht?

Beispiel 2, Parallel Session Attack:

- Eve hat keine Schlüssel.
- Eve tut so, wie wenn sie Alice wäre (Schritt 1)
- Eve tut so, wie wenn sie Alice wäre und mit Bob nochmals Kontakt aufnimmt, sie schickt dann R_B an Bob (Schritt 3).
- Eve beantwortet die Anfrage an Bob aus Schritt 2), indem sie die Antwort von Bob aus Schritt 4 an Bob zurückschickt. → Bob glaubt mit Alice zu kommunizieren.
- Eve bricht die zweite Anfrage (Schritt 3) & 4)) ab.
- Eve überprüft die Authentizität von Bob nicht.

Eve ohne Schlüssel

Bob mit sym. Schlüssel K



1. A = „Ich bin Alice“, R_A

2. $C_A = ENC(R_A, K)$ & R_B

5. $C_B = ENC(R_B, K)$



3. „Ich bin Alice“, R_B

4. $C_B = ENC(R_B, K)$ & R_{B-neu}



Beispiel 2, Weitere Verbesserungen 2 & 3:

- Verbesserung 2: Unterschiedliche Schlüssel pro Richtung verwenden $K_{AB} \neq K_{BA}$.

Alice mit sym. Schlüssel K_{AB} & K_{BA}



A = „Ich bin Alice“, R_A



$C_A = ENC(R_A, K_{BA})$ & R_B



$C_B = ENC(R_B, K_{AB})$



Bob mit sym. S. K_{AB} & K_{BA}



- Verbesserung 3: Die ID von Alice resp. Bob mitverschlüsseln, also neu $M = (B, R_A)$

Alice mit sym. Schlüssel K



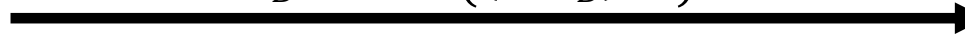
A = „Ich bin Alice“, R_A



$C_A = ENC((B, R_A), K)$ & R_B



$C_B = ENC((A, R_B), K)$



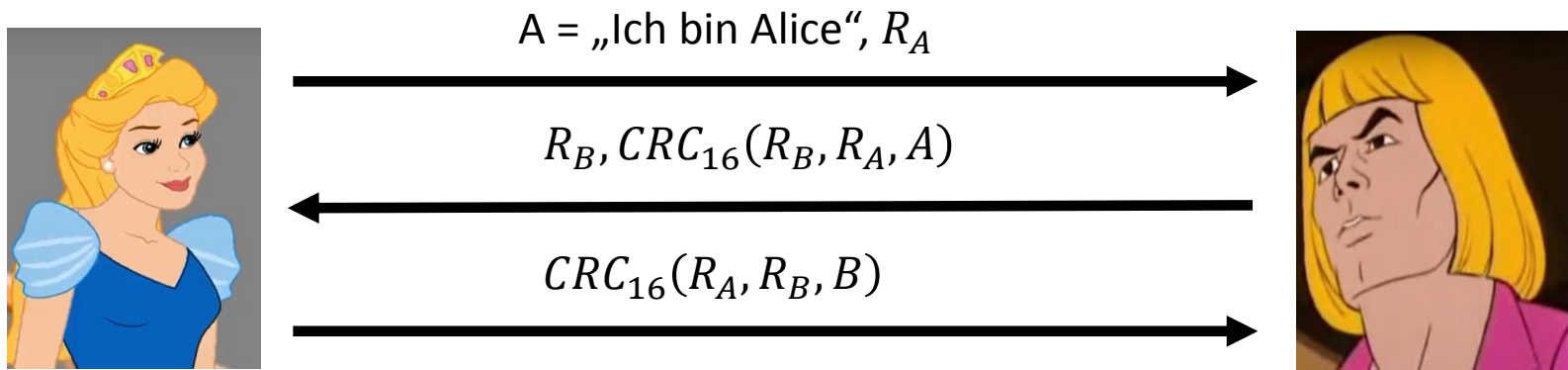
Bob mit sym. Schlüssel K



Doch leider erwiesen sich diese Verbesserungen als noch nicht genügend:

Beispiel 3: Dieses Authentisierungsprotokoll musste ich in meiner Industrietätigkeit auf seine Tauglichkeit überprüfen.

- A = Identifikationsnummer von Alice
- B = Identifikationsnummer von Bob
- R_A = Zufallswert von Alice, R_B = Zufallswert von Bob, «R» steht für Random.
- CRC_{16} = kryptographisch nicht sichere Hashfunktion ohne Schlüssel
- $CRC_{16}(M)$ = Berechnung des CRC_{16} Hashwertes (= Prüfziffer) über die Meldung M.
- Es werden keine (symmetrischen Schlüssel) verwendet.



Auf beiden Seiten wird die Überprüfung des $CRC_{16}(, ,)$ gemacht.

Kritik & Bemerkung:

- Absolut unsicher, alle Daten sind ersichtlich (was grundsätzlich noch kein Problem ist) und der $CRC_{16}(R_A, R_B, B)$ ist eine schlüssellose kryptographisch nicht sichere Hashfunktion (= Prüfziffer). D.h. Jedermann kann sich als Bob (oder Alice) gegenüber dem Anderen ausgeben.
- Aber tatsächlich ist der Protokollaufbau sehr gut und diente als erstes wirklich gutes Authentisierungsprotokoll, aber mit einer MAC-Berechnung (siehe weiter hinten).

Fazit aus den 3 Beispielen

1. Die drei Beispiele zeigen, dass beim Definieren und Einsetzen von kryptographischen Protokollen sehr vorsichtig vorgegangen werden muss.
2. Authentisier- und Schlüsselaustauschprotokolle müssen zusammenspielen (siehe später).
3. Es gibt ganz neue, raffinierte Attacken auf Protokolle. Brute-Force ist dazu im Vergleich wie „Keule“ zu „Judo ohne Krafteinsatz“.
4. Die Beispiele zeigen insbesondere:
 - a) Beispiel 1: Ein Protokoll ohne Stellen einer «Frage» (= Randomwert) kann **nicht** als Authentisierprotokoll funktionieren.
 - b) Beispiel 2: starker Algorithmus, schwaches Protokoll → es gibt Attacken.
 - c) Beispiel 3: schwacher Algorithmus, starkes Protokoll → ohne Aufwand geknackt.

Beispiel 4: „KryptoKnight“, das erste effiziente – kommt mit minimaler Anzahl Meldungsbits und Anzahl Meldungen aus – gegenseitiges Authentisierungsprotokoll (Im IBM Forschungslabor in Rüschlikon ZH ca. 1993 entworfen).

- A = Identifikationsnummer von Alice
- B = Identifikationsnummer von Bob
- R_A = Zufallswert von Alice, R_B = Zufallswert von Bob, „R“ steht für Random.
- $MAC(M, K)$ = CBC-MAC oder HMAC Berechnung mit Schlüssel K über die Meldung M .

Alice mit sym. Schlüssel K



Bob mit sym. Schlüssel K



$A = \text{„Ich bin Alice“}, R_A$

$R_B, MAC((R_A, R_B, B), K)$

$MAC((R_A, R_B), K)$

Wichtig: Auf beiden Seiten werden MAC's gemacht.

Bemerkungen:

- Es war das erste Protokoll, dass immun gegen die sog. Parallel Session und Oracle Session Attacken war.
- Grundsätzlich kann auch mit 2 Schlüsseln (je einer pro Richtung) gearbeitet werden.
- Es können auch Signaturen verwendet werden (dann braucht es zwei Key-Paare!!)
- Es dient als Grundlage für viele Authentisierungsprotokolle.
- Es kann sehr leicht zu einem Schlüsselaustauschprotokoll erweitert werden.

Parallel & Oracle Session Attacks

= Attacken auf Authentisier- Protokolle

Beispiel 2, Parallel Session Attacke:

- Wurde schon ein paar Folien weiter vorne gemacht.

Beispiel 4, Vorbereitung zur Parallel Session Attacke:

Als Vorbereitung zur Parallel Session Attacke von „KryptoKnight“ betrachten wir nun noch den Protokollablauf, wenn Bob mit dem Protokoll anfängt. Das erleichtert uns dann in der Attacke, die Sichtweise von Eve nachzuvollziehen.

Alice mit sym. Schlüssel K



Bob mit sym. Schlüssel K



B = „Ich bin Bob“, R_B

$R_A, \text{MAC}((R_B, R_A, A), K)$

$\text{MAC}((R_B, R_A), K)$

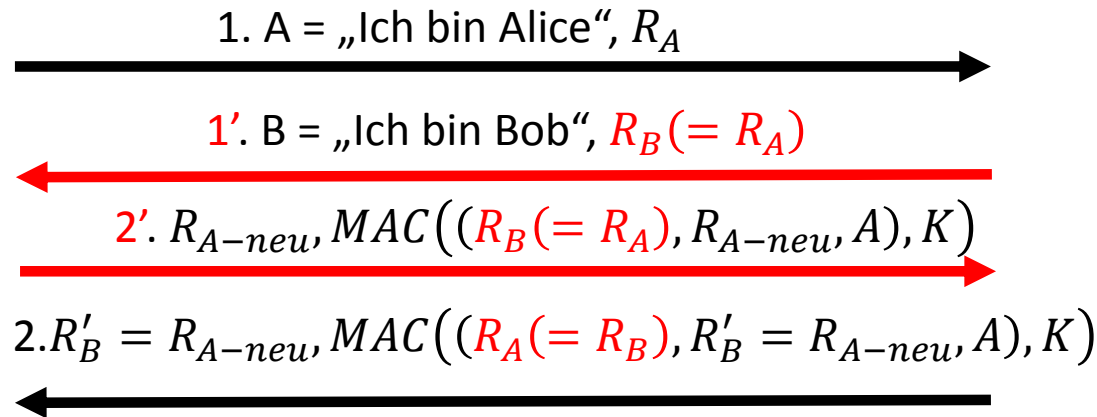
Beispiel 4, Parallel Session Attacke:

- Eve hat keine Schlüssel.
- Alice will sich mit Bob authentisieren, aber Eve will sich als Bob ausgeben.
- Eve tut so, wie wenn sie Bob wäre und auch gerade mit Alice Kontakt aufnehmen möchte, sie schickt dann den eben abgefangenen R_A an Alice zurück (Schritt 1').
- Alice beantwortet die Anfrage von Eve korrekt (Schritt 2').
- Eve schickt als Antwort zur ersten Anfrage (aus Schritt 1) die soeben von Alice erhaltene Antwort (aus Schritt 2') an Alice zurück (Schritt 2). Dabei tut Eve so, wie wenn $R_B = R_{A-neu}$

Alice mit sym. Schlüssel K



Eve ohne Schlüssel



- Aber Alice erwartet aber die Meldung $R'_B, MAC((R_A (= R_B), R'_B = R_{A-neu}, B), K)$ und bricht das Protokoll ab. Den geforderten $MAC((R_A (= R_B), R'_B = R_{A-neu}, B), K)$ kann Eve aber nicht berechnen, sie ist ja nicht im Besitz vom Schlüssel K.

Bemerkung:

Also diese Parallel Session Attacke funktioniert nicht \Rightarrow dass es keine anderen (Parallel Session) Attacken geben könnte!!

Beispiel 5: Ein (altes) ISO Protokoll für eine Mutual Authentication.

- A = Identifikationsnummer von Alice
- B = Identifikationsnummer von Bob
- R_A = Zufallswert von Alice, R_B = Zufallswert von Bob, «R» steht für Random.
- $E(M, K)$ = Verschlüsselung der Meldung M mit Schlüssel K.

Alice mit sym. Schlüssel K



Bob mit sym. Schlüssel K



„Ich bin Alice“, R_A

$E(R_A, K), E(R_B, K)$

R_B

Ablauf:

- A erzeugt eine Zufallszahl R_A und schickt sie B.
- B erzeugt ebenfalls eine Zufallszahl R_B , verschlüsselt beide und schickt beide A zurück.
- A entschlüsselt zuerst den eigenen, aber von B verschlüsselten Wert. Ist der entschlüsselte gleich ihrer Zufallszahl, weiss A, dass B die Meldung geschickt hat. Somit ist A sicher mit dem Teilnehmer B zu kommunizieren (könnte man meinen!!!).
- Nun entschlüsselt A den verschlüsselten Zufallswert von B und schickt diesen B zurück.
- B kontrolliert nun, ob der von A zurückgeschickte Wert auch tatsächlich derjenige ist, den er auch erzeugt hat. Falls ja, weiss B, dass A ihm geantwortet hat und B ist sicher, mit dem Teilnehmer A zu kommunizieren.

Beispiel 5, Fortsetzung:

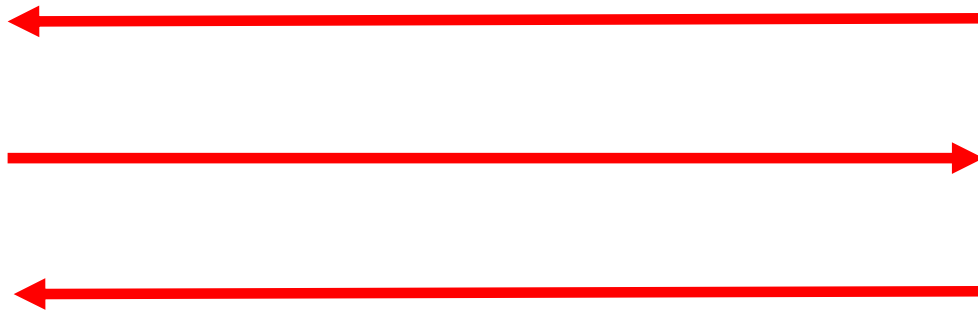
- A = Identifikationsnummer von Alice
- B = Identifikationsnummer von Bob
- R_A = Zufallswert von Alice, R_B = Zufallswert von Bob, «R» steht für Random.
- $E(M, K)$ = Verschlüsselung der Meldung M mit Schlüssel K .

Aufgabe 3:

Bestimmen Sie nun den Protokollablauf, wenn Bob mit der Anfrage beginnt. Benutzen Sie die vorgegebene Vorlage.

Alice mit sym. Schlüssel K

Bob mit sym. Schlüssel K



Beispiel 5, Fortsetzung:

Aufgabe 4:

Führen Sie die Parallel Session Attacke am Protokoll Beispiel 5: durch. Benutzen Sie die vorgegebene Vorlage.

Alice mit sym. Schlüssel K



Eve ohne Schlüssel



„Ich bin Alice“, R_A



$E(R_A, K), E(R_B, K)$



R_B



Beispiel 5: Verbesserung:

- Problemerkennung: Ein Problem ist sicher, dass eine richtungsunabhängige Verschlüsselung angewandt wird.
- Verbesserung: Eine Asymmetrie also z.B. richtungsabhängige Verschlüsselung einbauen.
 - Man definierte, dass Bob nicht R_A , sondern $R_A \oplus B$ (B die Id.nummer von Bob) verschlüsseln muss → **wir werden sehen, dass das nicht viel bringt!**
 - Besser wäre (wahrscheinlich), wenn mit $K_{AB} \neq K_{BA}$, also mit richtungsunabhängigen Schlüssel verschlüsselt würde. Ob es in diesem Fall nicht andere Attacken gibt, lassen wir an dieser Stelle stehen. **Denn wir werden dem Grundsatz folgen, dass solche Protokolle nicht auf Verschlüsselungen, sondern auf MAC-Berechnungen basieren sollen.**

Alice mit sym. Schlüssel K



Bob mit sym. Schlüssel K



„Ich bin Alice“, R_A

$E(R_A \oplus B, K), E(R_B, K)$

R_B

Alice erhält nach dem Entschlüsseln also $R_A \oplus B$. Sie muss demnach noch $(R_A \oplus B) \oplus B = R_A$ berechnen, um ihr R_A zu erhalten und zu kontrollieren

Beispiel 5: Verbesserung, Fortsetzung:

- Problemerkennung: Ein Problem ist sicher, dass eine richtungsunabhängige Verschlüsselung angewandt wird.
- Verbesserung: Eine Asymmetrie – also z.B. richtungsabhängige Verschlüsselung – einbauen.
 - Man definierte, dass Bob nicht R_A , sondern $R_A \oplus B$ (B die Id.nummer von Bob) verschlüsseln muss → **wir werden sehen, dass das nicht viel bringt!**
 - Besser wäre (wahrscheinlich), wenn mit $K_{AB} \neq K_{BA}$, also mit richtungsunabhängigen Schlüsseln verschlüsselt würde. Ob es in diesem Fall nicht andere Attacken gibt, lassen wir an dieser Stelle unbeantwortet. **Denn wir werden dem Grundsatz folgen, dass solche Protokolle nicht auf Verschlüsselungen, sondern auf MAC-Berechnungen basieren sollen.**

Alice mit sym. Schlüssel K



Bob mit sym. Schlüssel K



„Ich bin Alice“, R_A

$E(R_A \oplus B, K), E(R_B, K)$

R_B

Alice erhält nach dem Entschlüsseln also $R_A \oplus B$. Sie muss demnach noch $(R_A \oplus B) \oplus B = R_A$ berechnen, um ihr R_A zu erhalten und zu kontrollieren

Beispiel 5: Verbesserung, Analyse: Wiederum zunächst die Sicht von Bob.

Alice mit sym. Schlüssel K



Bob mit sym. Schlüssel K



„Ich bin Bob“, R_B

$E(R_B \oplus A, K), E(R_A, K)$

R_A

Alice mit sym. Schlüssel K



„Ich bin Alice“, R_A

Eve ohne Schlüssel



„Ich bin Bob“, $R_B = R_A \oplus A \oplus B$

$E(R_B \oplus A = R_A \oplus B, K), E(R_{A\text{-neu}}, K)$

$E(R_A \oplus B, K), E("R_B" = R_{A\text{-neu}}, K)$

Alice berechnet

$$(R_A \oplus A \oplus B) \oplus A \\ = (R_A \oplus B)$$

" R_B " (= $R_{A\text{-neu}}$)

$R_{A\text{-neu}}$

Alice ist schon nach dem zweiten schwarzen Meldungsschritt überzeugt, dass sie mit Bob kommuniziert. D.h. Wie bei Aufgabe 3, hätte die erste Variante schon gereicht.

Die Oracle Session Attacke, ev. Homework

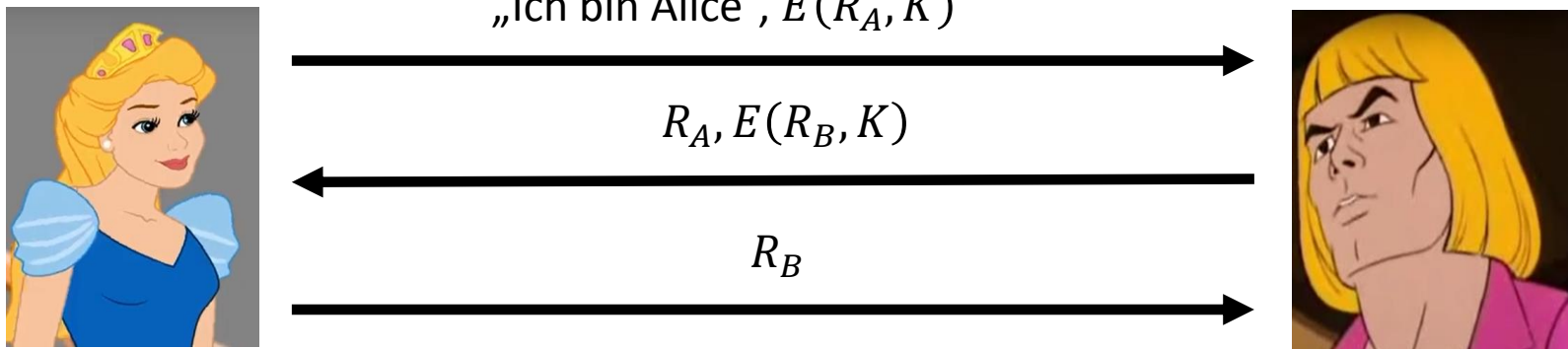
Man hat eine zweite Attacke – die Oracle Session Attacke – gefunden.

Beispiel 6: Das nachfolgende Authentisierprotokoll ist nicht immun gegen diese Attacke.

- R_A = Zufallswert von Alice, R_B = Zufallswert von Bob, «R» steht für Random.
- $E(M, K)$ = Verschlüsselung der Meldung M mit Schlüssel K.

Alice mit sym. Schlüssel K

Bob mit sym. Schlüssel K



1. A erzeugt eine Zufallszahl R_A und schickt sie mit dem Key K verschlüsselt an B.
2. B entschlüsselt zuerst den verschlüsselten Wert, erzeugt dann ebenfalls eine Zufallszahl R_B , verschlüsselt sie mit dem Schlüssel K und schickt beides A zurück.
3. A prüft zuerst, ob B die richtige Zufallszahl R_A zurückgeschickt hat, falls ja, weiss A, dass B die Meldung geschickt hat. Nun entschlüsselt A den Zufallswert von B und schickt ihn in klar an B zurück.
4. B kontrolliert nun, ob der von A zurückgeschickte Wert auch tatsächlich derjenige ist, den er auch erzeugt hat. Falls ja, weiss B, dass A ihm geantwortet hat, und B ist sicher mit dem Teilnehmer A zu kommunizieren (könnte man meinen!!!).

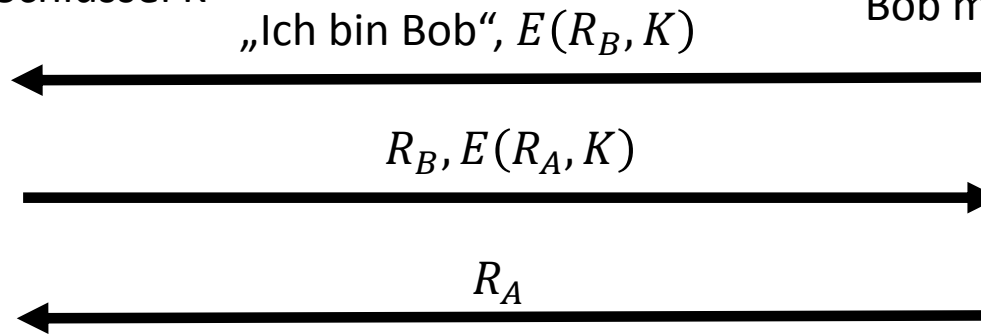
Beispiel 6:

Das gleiche Authentisierprotokoll dargestellt, wenn Bob beginnt.

- R_A = Zufallswert von Alice, R_B = Zufallswert von Bob, «R» steht für Random.
- $E(M, K)$ = Verschlüsselung der Meldung M mit Schlüssel K.

Alice mit sym. Schlüssel K

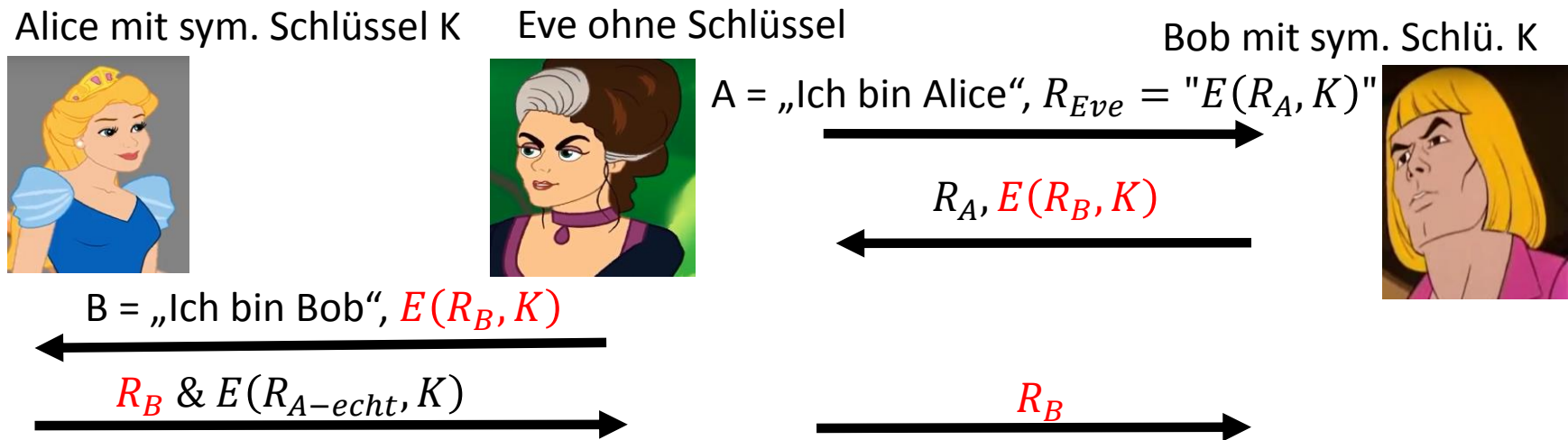
Bob mit sym. Schlüssel K



1. B erzeugt eine Zufallszahl R_B und schickt sie mit dem Key K verschlüsselt an A.
2. A entschlüsselt zuerst den verschlüsselten Wert, erzeugt dann ebenfalls eine Zufallszahl R_A , verschlüsselt sie mit dem Schlüssel K und schickt beides B zurück.
3. B prüft zuerst, ob A die richtige Zufallszahl R_B zurückgeschickt hat, falls ja, weiss B, dass A die Meldung geschickt hat. Nun entschlüsselt B den Zufallswert von A und schickt ihn in klar an A zurück.
4. A kontrolliert nun, ob der von B zurückgeschickte Wert auch tatsächlich derjenige ist, den er auch erzeugt hat. Falls ja, weiss A, dass B ihm geantwortet hat, und A ist sicher mit dem Teilnehmer B zu kommunizieren (könnte man meinen!!!).

Oracle Session Attacke = Eine Man-in-the-middle Attacke:

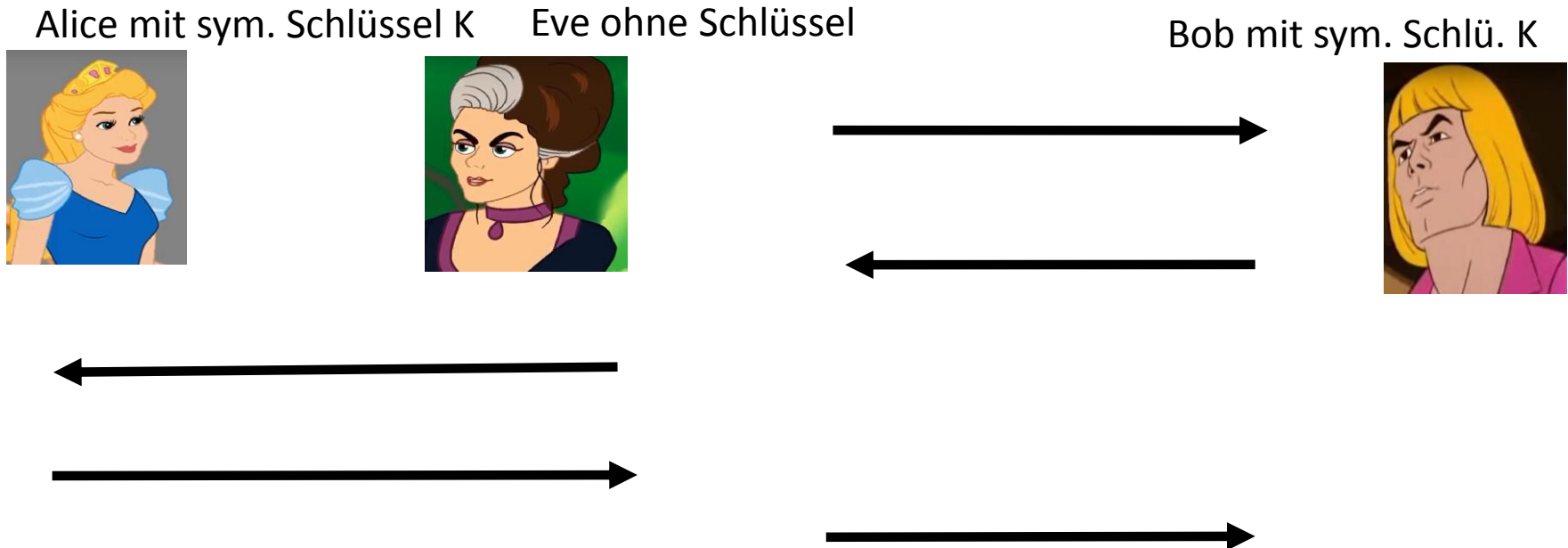
- Eve tut so, wie wenn sie A wäre und schickt B einen Zufallswert R_{Eve} , der für B wie $E(R_A, K)$ aussieht.
- In der Meinung mit A zu kommunizieren, entschlüsselt B den Wert und schickt diesen, sowie $E(R_B, K)$ an Eve zurück.
- Eve kann natürlich nicht kontrollieren, ob B das richtige R_A zurückgeschickt hat. Eve glaubt das einfach. Eve hat nun das Problem dem Teilnehmer B das richtige R_B zurückzuschicken. Um R_B zu bekommen, eröffnet Eve nun eine Session mit A, m.a.W. sie benutzt A zum Entschlüsseln des $E(R_B, K)$.
- A macht ihr natürlich den „Gefallen“.
- Eve kann nun B das korrekte R_B zurückschicken.
- B kontrolliert R_B , und er hat das Gefühl mit A zu kommunizieren.
- Das Protokoll mit A beendet Eve einfach, ohne es abzuschliessen; d.h. sie bricht einfach ab.



Bemerkung: Auch wenn (cf. Verbesserung von Beispiel 2)) von $A \rightarrow B$ und $B \rightarrow A$ unterschiedliche Schlüssel $K_{AB} \neq K_{BA}$ oder eine Public Key Verschlüsselung verwendet werden, ist hier diese Oracle Session Attacke erfolgreich!!

Aufgabe 5:

Führen Sie die Oracle Session Attacke an „KryptoKnight“ – Beispiel 4: – durch. Benutzen Sie die vorgegebene Vorlage.



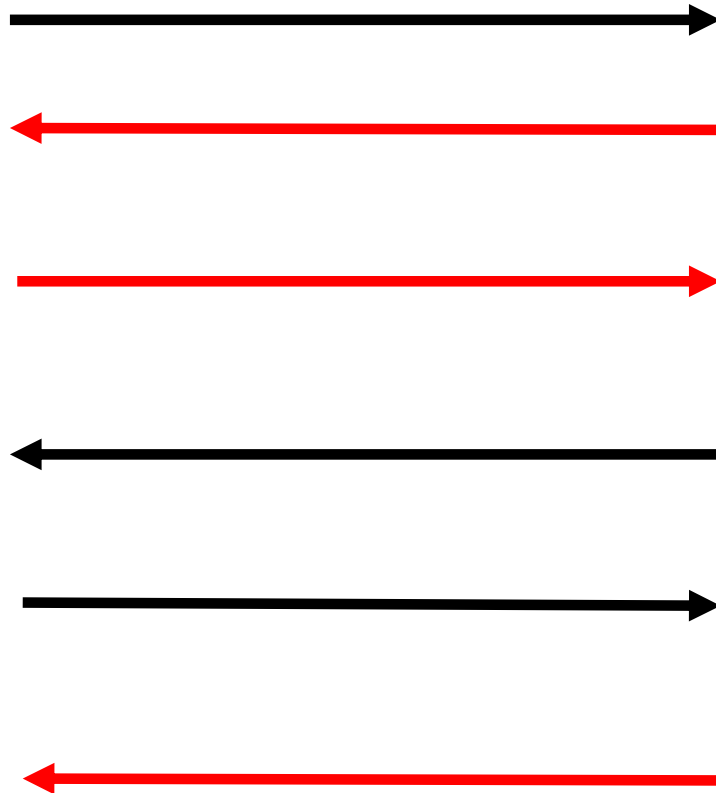
Bemerkungen:

1. Also bei „KryptoKnight“ funktionieren weder die Parallel noch die Oracle Session Attacke!
2. \nRightarrow dass es keine anderen Attacken gegen dieses Protokoll geben könnte!!
3. „KryptoKnight“ kann erweitert werden, so dass mit der Authentisierung auch ein Schlüssel ausgetauscht wird. Es ist dann ein 2PAKDP = 2Party Authenticated Key Distribution Protocol. Sie weiter hinten in „Key Establishment“.

Aufgabe 6:

Ist das Authentisierprotokoll aus Beispiel 6: wenigstens sicher in Bezug auf eine Parallel Session Attacke? Benutzen Sie die vorgegebene Vorlage. Wie sähe es aus, wenn von $A \rightarrow B$ und $B \rightarrow A$ unterschiedliche Schlüssel $K_{AB} \neq K_{BA}$ verwendet würden?

Alice mit sym. Schlüssel K



Eve ohne Schlüssel



Key Establishment Protokolle

Definition von Key Establishment Protokolle

- Das Verteilen und/oder Festlegen von kryptographischen Keys ist in verschiedenen Sicherheitsmechanismen zentral.
- Ein „Key Establishment“ ist ein Prozess oder Protokoll, indem geheime Schlüssel für zwei oder mehr Teilnehmer verfügbar werden. Diese (Session)-Schlüssel werden dann beim nachfolgenden Einsatz der Krypto-Primitiven direkt oder in abgeleiteter Form verwendet (in diesem Fall kommt dann eine sogenannte Key Derivation Function KDF zu Zuge).
- Typen von Key Establishment Protokollen:
 - Key Transport Protokoll: Die Schlüssel werden von einer Partei erzeugt, diese verschickt die Schlüssel dann an alle anderen Parteien.
 - Key Agreement Protokoll: Die Schlüsselinputs werden von allen Parteien erzeugt, d.h. jeder kann etwas zum Schlüssel beitragen.
 - Hybrid: Die Keys werden > 1 Partei, aber nicht von allen erzeugt, d.h. aus Sicht dieser Teilnehmer ist es ein Agreement Protokoll. Aus Sicht der restlichen Teilnehmer ist es ein Transport Protokoll.

Klassifizierung von Schlüsselverteilsprotokollen

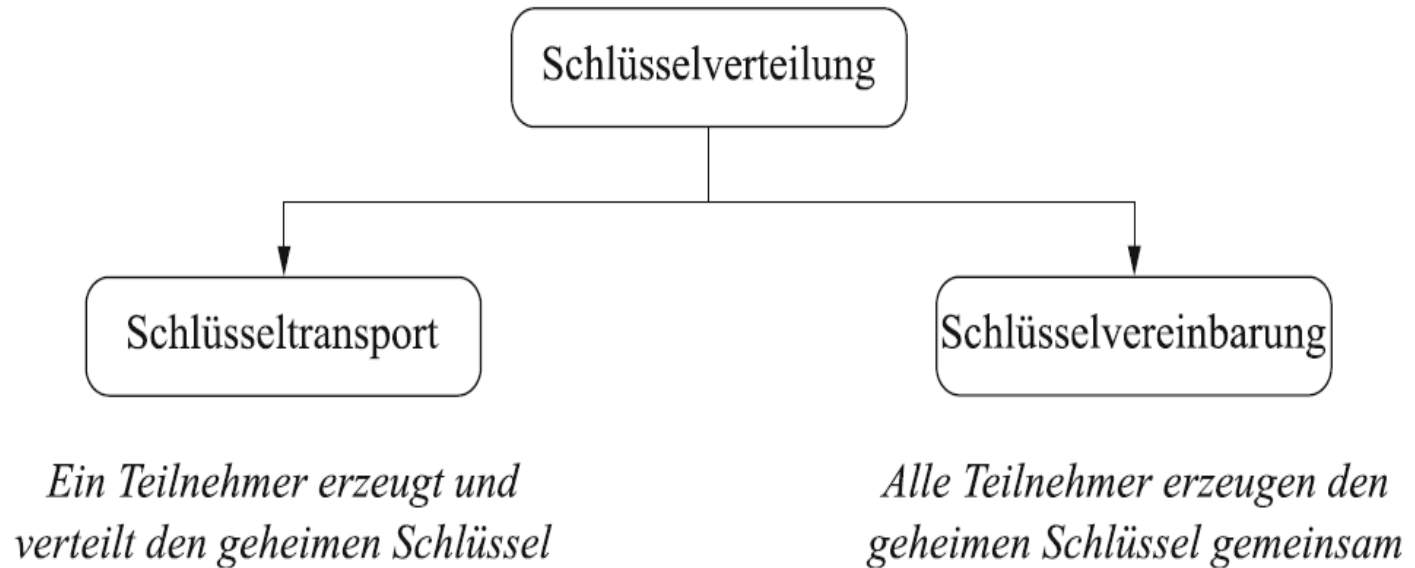
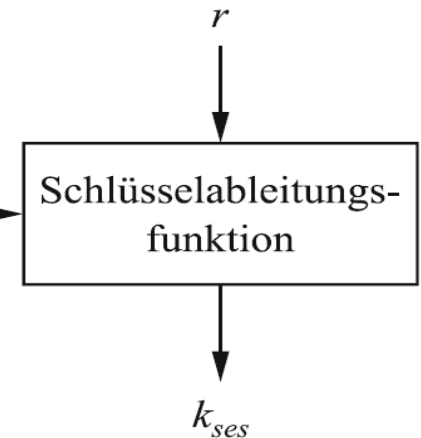


Abb. 13.1 Klassifizierung von Schlüsselverteilungsprotokollen

Aus [CP-D], S. 378. Hybride Verfahren werden nicht aufgezeigt resp. besprochen.

Key Derivation Function KDF = Schlüsselableitungsfkt.

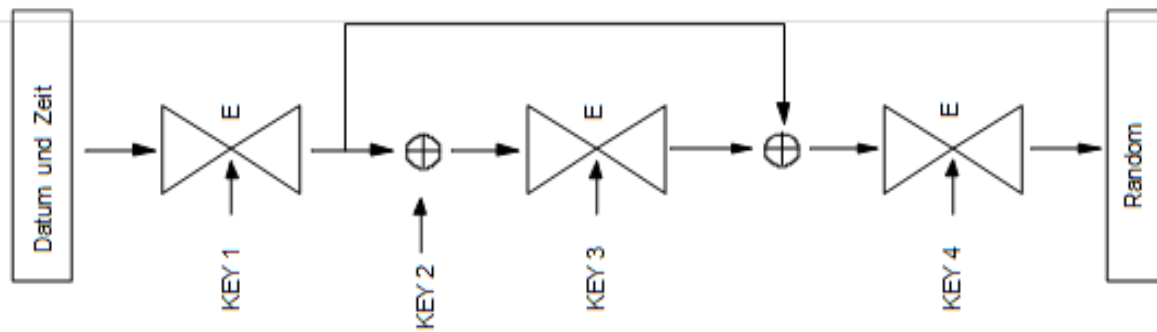
- Ein zufälliger (einmaliger) Input r (oft „Nonce“ genannt) wird mittels einer one-way function und eines Keys k_{AB} zu einem Session Key k_{ses} abgeleitet.
- k_{ses} = Sitzungsschlüssel = Sessionkey
- k_{ses} kann aber auch ein temporäre S. = ephemeral key = kek (key encryption key) sein. k_{AB} →



Aus [CP-D], S. 379,

Abb. 13.2 Das Prinzip der Schlüsselableitung

- **Beispiel 7:** $k_{ses} = HMAC_{k_{AB}}(r)$
- **Beispiel 8:** Cf. Kap. 8.4.1 in JS Skripte, r = „Datum und Zeit“, resp. irgendein anderer geeigneter Input; $k_{AB} = KEY_1, KEY_2, KEY_3, KEY_4$; $k_{ses} = \text{Random}$



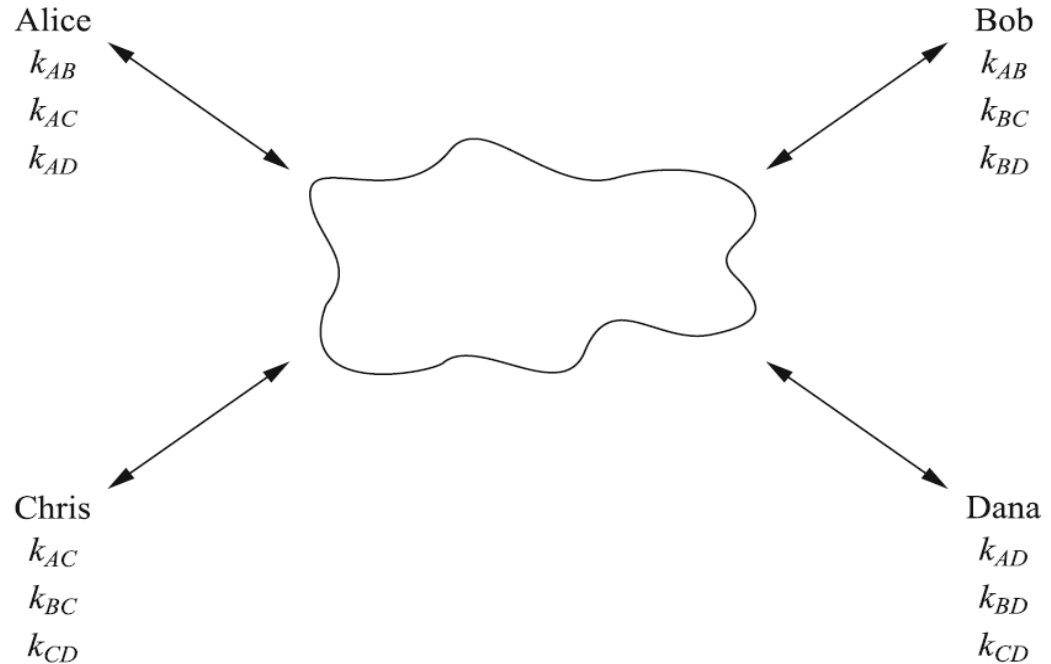
Eigenschaften von Key Establishment Protokolle

Im Wesentlichen wurden folgende Eigenschaften für Key Establishment Protokolle definiert:

- Verwenden von Session resp. ephemeral Keys: Kurzlebigkeit von aktuellen Schlüsseln.
- Key Freshness: Jeder Partner kann verifizieren, dass ein Session Key neu ist und keine Schlüssel von alten Sessions gebraucht wurden.
- Key Authentication: Man weiss von wem der Schlüssel ist.
- (Perfect) Forward Secrecy PFS: PFS (in [CP-D], Kap. 13.2.3 als (perfekte) Vorwärtssicherheit bezeichnet) ist dann gegeben, wenn die (aktiven) Sessionkeys nicht berechnet werden können, selbst wenn die long-term Keys bekannt geworden sind.
 - An dieser Stelle zwei Anmerkungen aus dem Buch END-TO-END ENCRYPTED MESSAGING von Rolf Oppliger.
 - Man kann zu recht monieren, dass es eher eine Rückwärtssicherheit ist. Denn i.d.R. werden ja keine weiteren Session Keys mehr erzeugt, wenn der long-term Key geknackt wurde (und man es merkte). D.h. wenn der long-term Key bekannt ist, dann können nur frühere (darum eben «rückwärts») Session Keys ggf. berechnet werden.
 - Das Wort «perfekt» hat in der Kryptologie eigentlich eine andere Bedeutung (cf. perfekte Sicherheit nach Claude Shannon), nämlich, dass selbst mit unbeschränkten Ressourcen man nicht besser dran ist, als mit raten.
- Und viele weitere an dieser Stelle nicht erwähnte Eigenschaften.

Das n^2 -Schlüsselverteilproblem

Abb. 13.3 Schlüssel in einem Netzwerk mit $n = 4$ Teilnehmern



- Jeder Teilnehmer speichert $n - 1$ Schlüssel.
- Insgesamt existieren im Netzwerk $\frac{n(n-1)}{2} \approx \frac{n^2}{2}$ Schlüssel.
- Siehe oben $\frac{4 \cdot 3}{2} = 6$ Schlüssel, nämlich AB, AC, AD, BC, BD, CD
- Je nach Design, ist die Anzahl um den Faktor 2 (z.B. von $A \rightarrow B$ und $B \rightarrow A$ unterschiedliche Schlüssel oder 2 Dienste), 4 oder mehr (mehrere Dienste) grösser (siehe PR 1, Kap. 6)
- Schlechte Skalierbarkeit: Wenn ein neuer Teilnehmer aufgenommen wird, müssen sichere Kanäle zu jedem Teilnehmer aufgebaut werden, um neue Schlüssel hochzuladen.

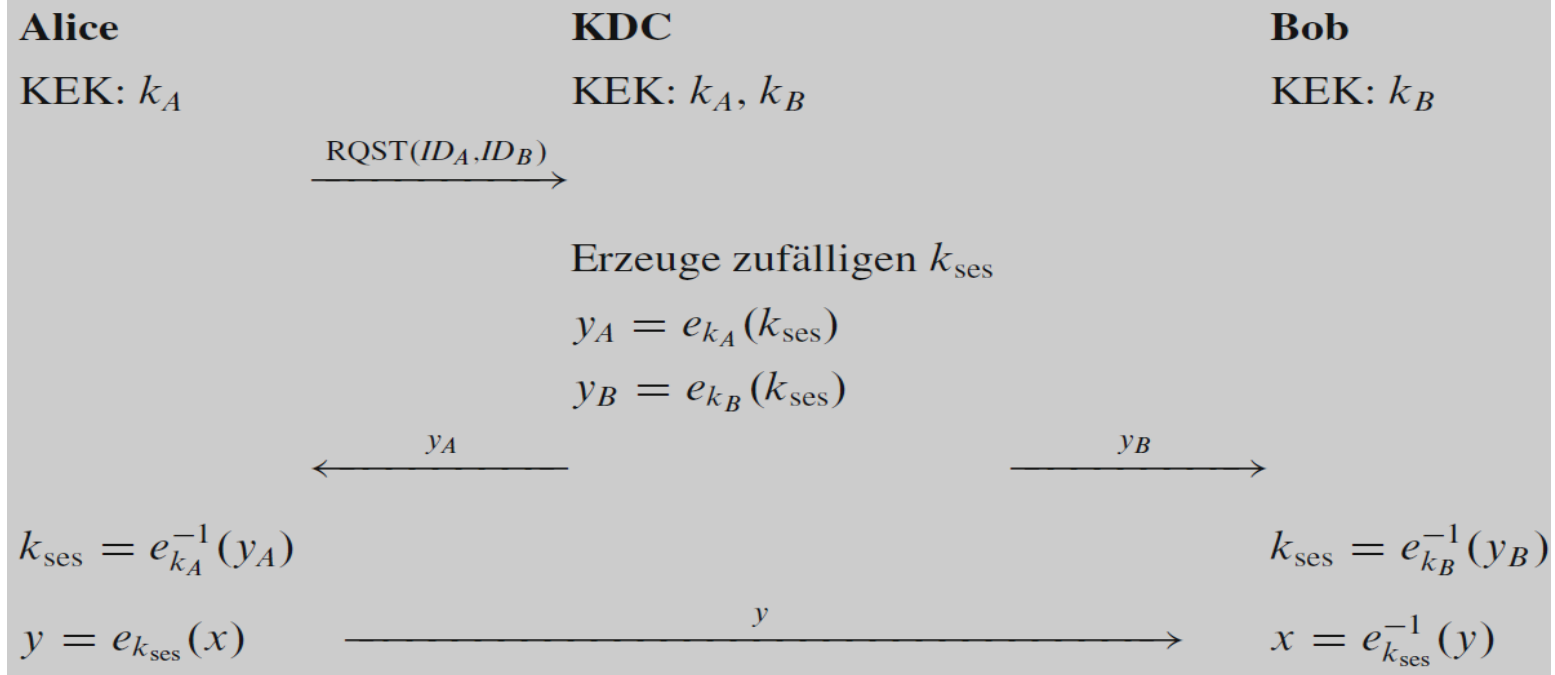
Schlüsselverteilung mit symmetrischer Kryptographie

- KDC = Key Distribution Center = Schlüsselservers, dem alle Teilnehmer vertrauen.
- KEK = Key encryption Key, sie dienen der sicheren Übertragung von Sitzungsschlüsseln. Jeder Benutzer hat jeweils einen solchen Key mit dem KDC.

Basisprotokoll (Details cf. [CP-D], 382 ff.

Eine Bedingung für diese Protokollfamilie ist, dass jeder Benutzer U einen geheimen KEK k_U mit dem Schlüsselservers, d.h. dem KDC, teilt. Dieser muss vorab über einen sicheren Kanal, z.B. manuell durch einen Systemadministrator, verteilt werden.

Schlüsselverteilung mit einem KDC



Key Words zu den Basic Protokollen S. 382 ff. in [CP-D]

- Vorteile eines KDC:
 - Nur n Langzeit symmetrische Schlüssel (sogenannte KEK = key encryption keys) sind nötig.
 - Diese symmetrischen Langzeit Schlüssel werden nur zwischen dem jeweiligen Teilnehmer und dem KDC ausgetauscht. Im Gegensatz dazu muss im Netzwerk jeder mit jedem den entsprechenden Schlüssel austauschen.
 - Ein neuer Teilnehmer muss nur im KDC – zumindest in Bezug auf die Schlüssel – erfasst werden → gute Skalierbarkeit.
- Weitere Key Words:
 - *Kerberos* ist ein bekanntes Produkt.
 - Die Verschlüsselung der Schlüssel (z.B. mit AES im CBC-Modus) und die Meldungsverschlüsselung (z.B. mit AES im CTR-Modus) kann unterschiedlich sein.
 - Warum ist ein langlebiger KEK k_A – im Gegensatz zu k_{ses} – kein Problem?
 - Der theoretische Hintergrund ist die sogenannte unicity distance von Shannon. Im Wesentlichen heisst das: Mit einem symmetrischen Schlüssel kann man im Prinzip beliebig viele Zufallsbitstrings (was ja ein Schlüssel schlussendlich ist) verschlüsseln. Bei einer Meldung mit Redundanz gilt das eben nicht. Verschlüsselt man Klartexte mit Redundanz (also Klartexte, die nicht aus Zufallswerten bestehen) mit einem symmetrischen Schlüssel, dann «nützt» sich dieser ab.
 - Diese Betrachtungen haben wir im Thema «Klassische Analyse von symmetrischen Verfahren», speziell in Kap. 10.6.2 «Was ist der Unterschied?» schon einmal betrachtet.

Key Words zu den Basic Protokollen S. 382 ff. in [CP-D]

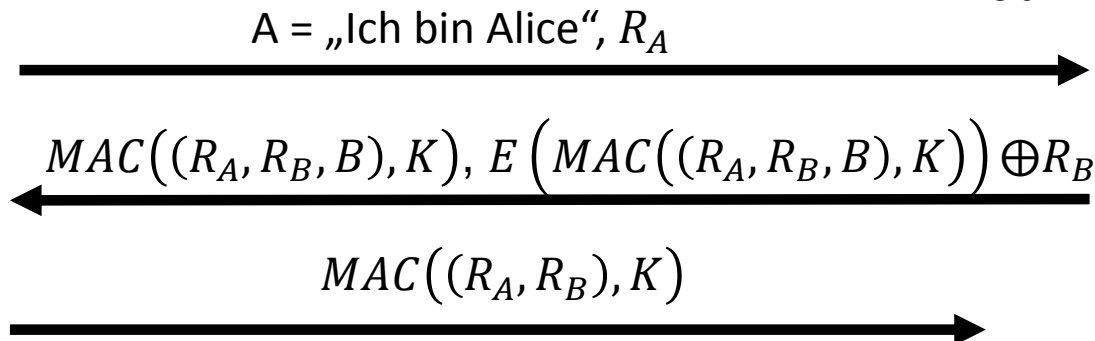
- Die Protokolle sind Grundprotokolle, die das Wesentliche aufzeigen sollen.
- Nachteile der vorgestellten Protokolle:
 - Replay Attacken sind möglich. D.h. Key Freshness fehlt, daher könnte ein alter – korumpierter – Schlüssel gebraucht werden.
 - Keine Key Confirmation: D.h. Alice und Bob wissen nicht, ob die Schlüssel auch wirklich für sie angefragt und bestimmt sind. Denn mit einer Key Confirmation Attacke kann ein legitimer aber betrügerischer Teilnehmer Oscar etwas Falsches vorgaukeln (siehe p. 386 in [CP-D]).
- Fazit:
 - Authentisierung ist bei einem Schlüsselaustausch Protokoll wichtig.
 - Authentisier- und Schlüsselaustauschprotokolle sind nicht trennbar.
 - Weitere Probleme bleiben (es gibt ja noch weitere Angriffe), z.B.
 - Verfügbarkeit: Ein KDC ist ein single point of failure.
 - Forward Secrecy ist nicht gewährleistet.

Beispiel 9: „KryptoKnight“, als ein 2PAKDP = 2Party Authenticated Key Distribution Protocol.

- Anstatt R_B schickt Bob $E \left(MAC((R_A, R_B, B), K) \right) \oplus R_B$
- R_B können beide rechnen und R_B wird der neue, ausgetauschte Schlüssel, also $R_B = K_{neu}$

Alice mit sym. Schlüssel K

Bob mit sym. Schlüssel K



Bemerkungen:

1. Besser wäre natürlich, wenn für die MAC-Berechnung von $A \rightarrow B$ und $B \rightarrow A$ und die Verschlüsselung von $B \rightarrow A$ jeweils unterschiedliche Schlüssel (also total 3) verwendet würden.
2. Bob kann natürlich ein KDC, also ein Key Distribution Center sein.
3. Es gibt dann auch 3PAKDP = 3Party Authenticated Key Distribution Protocols. Das ist dann ein Schlüsselaustausch zw. A und B über ein KDC.

Aufgabe 7: Ist das obige 2PAKDP ein Schlüsseltransport oder –vereinbarungsprotokoll?

Schlüsselverteilung mit asymmetrischer Kryptographie

z.B. mit Diffie – Hellman – Schlüsselaustausch

- Das Diffie-Hellman Schlüsselaustauschverfahren ist quasi die Mutter aller Schlüsselaustauschprotokollen.
- Siehe auch z.B. Das Elgamal Verschlüsselungsverfahren.
- Das Grundproblem ist die fehlende Authentisierung der Teilnehmer und die damit verbunden Man-in-the-middle MITM Attacke.
- Lösungsansatz = Zertifikate cf. [CP-D], Kap. 13.3 → Nicht Prüfungsstoff
(**ausser natürlich 13.3.1 die MITM Attacke**)
- Grundsätzlich wird in sehr vielen Schlüsselaustauschprotokollen mit Diffie-Hellman gearbeitet.
- Das Grundproblem bleibt: Wie kriege ich den ersten Schlüssel zum Teilnehmer?
- «Lösung» z.B. bei Whats'App: «Die erste Meldung wird quasi durch den Benutzer gesichert, danach ist es ja sicher». Sie nennen das «OTR», «Out of the record».

Zu guter Letzt eine konkrete Anwendung

- Das Zeigen einer konkreten Anwendung ist nicht gerade einfach. Der Grund liegt darin, dass es sofort von Spezialbegriffen und Abkürzungen nur so wimmelt. Man käme nicht darum herum, sich (ev. recht mühsam) in die Materie (sprich: das konkrete Protokoll) einzulesen.
- In Ilias habe ich einen Artikel «Kryptographie im Mobiltelefon» hochgeladen. Dieser Artikel sollte in nützlicher Frist lesbar sein. (Nicht Prüfungsstoff)

Basis-Test PR 12

Aussage	Richtig oder falsch?	Begründung
Ein Protokoll in unserem Sinne ist eine streng geregelte Abfolge von Schritten, resp. Meldungen.	<input checked="" type="checkbox"/> richtig <input type="checkbox"/> falsch	
Bei den Protokollen gibt es allgemeine aber keine spezifischen Eigenschaften.	<input type="checkbox"/> richtig <input checked="" type="checkbox"/> falsch	
Sowohl im Bereich der Kryptologie wie im Bereich Netzwerk gibt es viele Typen von Protokollen.	<input checked="" type="checkbox"/> richtig <input type="checkbox"/> falsch	
In einem Biometrischen Zutrittsverfahren kann man sagen, dass $FAR = FRR = 0$ ist.	<input type="checkbox"/> richtig <input checked="" type="checkbox"/> falsch	Ist nie null.
Bei einem Biometrischen Verfahren ist „Verifikation“ und „Identifikation“ identisch.	<input checked="" type="checkbox"/> richtig <input type="checkbox"/> falsch	
Bei Authentisierprotokollen hat man neuartige Attacken gefunden.	<input checked="" type="checkbox"/> richtig <input type="checkbox"/> falsch	
Wenn nur eine Partei etwas zum Schlüssel beiträgt, nennt man das „Schlüsselvereinbarung“.	<input type="checkbox"/> richtig <input checked="" type="checkbox"/> falsch	Schlüsselvereinbarung sind beide involviert.
Diffie-Hellman ist eine „Schlüsselvereinbarung“.	<input type="checkbox"/> richtig <input type="checkbox"/> falsch	
Es ist vollkommen unklar, wieso Langzeitschlüssel lang leben dürfen, Kurzzeitschlüssel aber nicht.	<input type="checkbox"/> richtig <input checked="" type="checkbox"/> falsch	

Lösungen

Aufgabe 1:

- Non repudiation of receipt
- Mutual Authentication

Aufgabe 2:

Anwendung	Geheim.	Datenauth.	Benutzerauth. (**)
D) Pay-TV inkl. Zugangsberechtigung	X		X
E) Berechtigung zur Ampelsteuerung			X
F) Bargeldbezug beim Bancomat	X	X	X
G) E-Banking	X	X	X

Aufgabe 3:

Alice mit sym. Schlüssel K



Bob mit sym. Schlüssel K



„Ich bin Bob“, R_B

$E(R_B, K), E(R_A, K)$

R_A

Aufgabe 4:

Alice mit sym. Schlüssel K



„Ich bin Alice“, R_A

Eve ohne Schlüssel



„Ich bin Bob“, $R_B = R_A$

$E(R_B = R_A, K), E(R_{A-\text{neu}}, K)$

$E(R_A, K), \text{Ein bel. Wert} = E("R_B", K)$

Da Alice R_A erhält, ist sie überzeugt mit Bob zu kommunizieren.

Für Eve ist " R_B " nicht relevant.

" R_B "

Diese Meldung braucht es nicht mehr, da Alice schon glaubt mit Bob zu kommunizieren. Eve schickt diese Meldung nicht mehr, sie bricht ab.

Aufgabe 4: Ablauf in Worten

- A erzeugt eine Zufallszahl R_A und will sie B schicken, dieser Wert wird aber von Eve abgefangen.
- Eve schickt A dieselbe Zufallszahl R_A zurück (d.h. sie tut so, als ob sie als B exakt im gleichen Moment auch eine Session eröffnen will, diese Session eröffnet „B“ „per Zufall“ mit dem gleichen Zufallswert.
- A antwortet nun, wie wenn sie von B zur Authentisierung aufgerufen worden wäre und schickt Eve die entsprechenden Kryptogramme zurück.
- Eve antwortet nun mit den erhaltenen Werten auf die ursprüngliche Sessionanfrage von A. Eve muss nur die Reihenfolge der Kryptogramme vertauschen.
- A erhält nun korrekte Authentisierungswerte und hat (ohne weitere Kontrollen) das Gefühl, mit B kommuniziert zu haben.

Bemerkung:

Durch das Verwenden von verschiedenen Schlüsseln in den Meldungen von A zu B und B zu A, könnte diese Attacke verhindert werden.

Aufgabe 4: Alternative Lösung

Alice mit sym. Schlüssel K

Eve ohne Schlüssel



„Ich bin Alice“, R_A

„Ich bin Bob“, $R_B = R_A$

$E(R_B = R_A, K), E(R_{A-neu}, K)$

$E(R_A, K), E("R_B" = R_{A-neu}, K)$

" R_B " (= R_{A-neu})

R_{A-neu}

Nun ist Alice auch nach dem roten Meldungs austausch überzeugt, dass sie mit Bob kommuniziert.

Aufgabe 5:

Alice mit sym. Schlüssel K



Eve ohne Schlüssel



Bob mit sym. Schlü. K



„Ich bin Alice“, $R_{Eve} = R_A$



$R_B, MAC((R_A, R_B, B), K)$



„Ich bin Bob“, R_B



$R_A, MAC((R_B, R_A, A), K)$



??, denn Eve müsste $MAC((R_A, R_B), K)$ schicken!

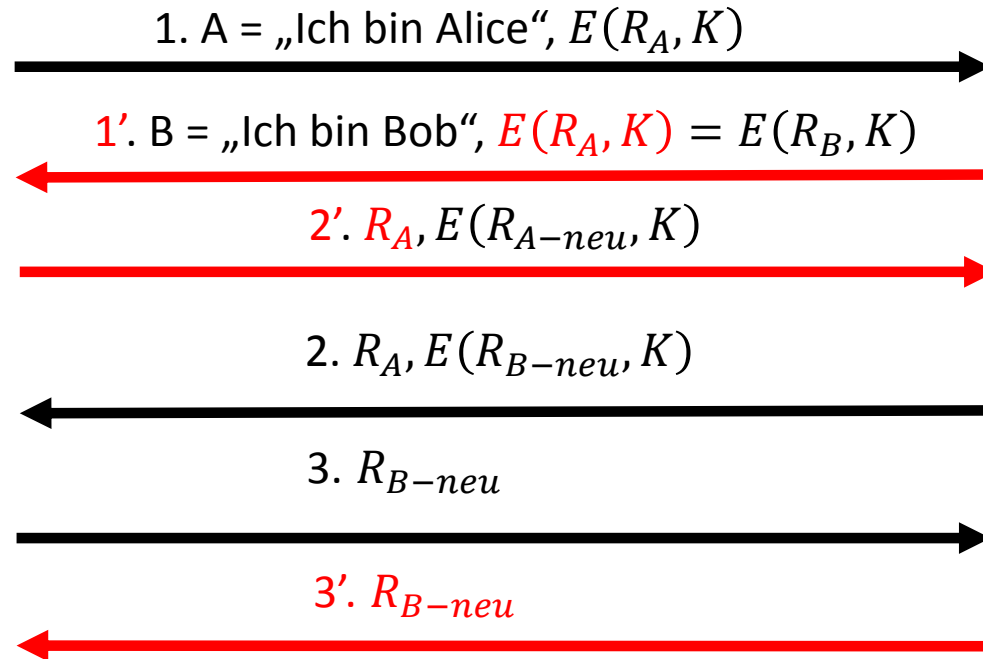


Aufgabe 6:

Alice mit sym. Schlüssel K



Eve ohne Schlüssel



Unterschiedliche Schlüssel $K_{AB} \neq K_{BA}$ würden den Angriff auch nicht verhindern!

Aufgabe 7: Es ist ein Schlüsseltransportprotokoll, da nur Bob etwas zum neuen Schlüssel beiträgt.

Basis-Test PR 12

Aussage	Richtig oder falsch?	Begründung
Ein Protokoll in unserem Sinne ist eine streng geregelte Abfolge von Schritten, resp. Meldungen.	<input checked="" type="checkbox"/> richtig <input type="checkbox"/> falsch	
Bei den Protokollen gibt es allgemeine aber keine spezifischen Eigenschaften.	<input type="checkbox"/> richtig <input checked="" type="checkbox"/> falsch	Es gibt viele spezifische Eigenschaften wie „Binding“ usw.
Sowohl im Bereich der Kryptologie wie im Bereich Netzwerk gibt es viele Typen von Protokollen.	<input checked="" type="checkbox"/> richtig <input type="checkbox"/> falsch	
In einem Biometrischen Zutrittsverfahren kann man sagen, dass $FAR = FRR = 0$ ist.	<input type="checkbox"/> richtig <input checked="" type="checkbox"/> falsch	Bei einer PIN-Eingabe am Bancomaten gilt das, bei einem Biometrischen Zutrittsverfahren nicht.
Bei einem Biometrischen Verfahren ist „Verifikation“ und „Identifikation“ identisch.	<input type="checkbox"/> richtig <input checked="" type="checkbox"/> falsch	„Verifikation“ = „one to one“ „Identifikation“ „one to many“
Bei Authentisierprotokollen hat man neuartige Attacken gefunden.	<input checked="" type="checkbox"/> richtig <input type="checkbox"/> falsch	Z.B. Parallel-Session und Oracle-Session und Man-in-the Middle Attacken.
Wenn nur eine Partei etwas zum Schlüssel beiträgt, nennt man das „Schlüsselvereinbarung“.	<input type="checkbox"/> richtig <input checked="" type="checkbox"/> falsch	„Schlüsseltransport“
Diffie-Hellman ist eine „Schlüsselvereinbarung“.	<input checked="" type="checkbox"/> richtig <input type="checkbox"/> falsch	Da beide etwas zum Schlüssel beitragen.
Es ist vollkommen unklar, wieso Langzeitschlüssel lang leben dürfen, Kurzzeitschlüssel aber nicht.	<input type="checkbox"/> richtig <input checked="" type="checkbox"/> falsch	Mit Langzeitschlüsseln werden nur Schlüssel (i.d.R. Kurzzeitschlüssel), also zufällige Werte verschlüsselt. Mit Kurzzeitschlüsseln werden Meldungen mit Redundanz verschlüsselt. Das ist ein gewaltiger Unterschied! Stichwort: Shannon's Unicity Distance. Dieser Begriff haben wir aber nicht weiter ausgeführt und erklärt!

Quellenangaben & Danksagung

- Das Titelbild und weitere Ideen und tw. Bilder sind aus Folien von Prof. Dr. Rolf Oppliger und seiner Vorlesung an der UNI ZH 2020, entnommen worden.
- Viele Ideen und einige Bilder (Secure Multi Party Computation) entstammen von Vorlesungen von Prof. Dr. Ueli Maurer, ETHZ.
- Einen besonderen Dank geht an Dr. Philippe A. Janson vom IBM Research Laboratory. Aufgrund seines Vortrages 1993 an der ETHZ zu KryptoKnigth, seinen Unterlagen und Papers wurde ich zum ersten Mal vertieft auf die zwei dargestellten Netzwerkattacken aufmerksam gemacht. Dieser Vortrag war eine Initialzündung zu eigenen (Seminar-)Vorträgen und vor allem auch zu kritischen Betrachtungen in meiner Arbeit als Kryptodesigner in den CH-Zahlungssystemen. Dadurch konnten wir viele Verbesserungen in realen Systemen umsetzen.
- Weitere Ideen entstammen aus den Vorlesungen von Prof. Dr. Mark Stamp, San Jose State University.
- Last but not least, sind – vornehmlich zum letzten Teil – stammen viele Ideen und Abbildungen von unserem Lehrbuch von Christoph Paar [CP-D].
- An dieser Stelle an Alle ein weiteres Mal ein herzliches Danke schön.