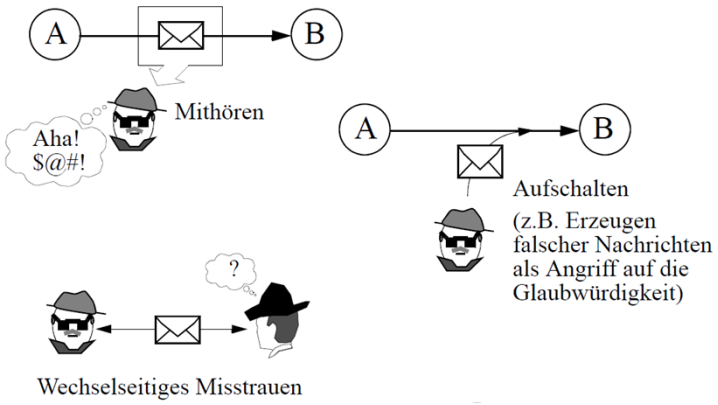
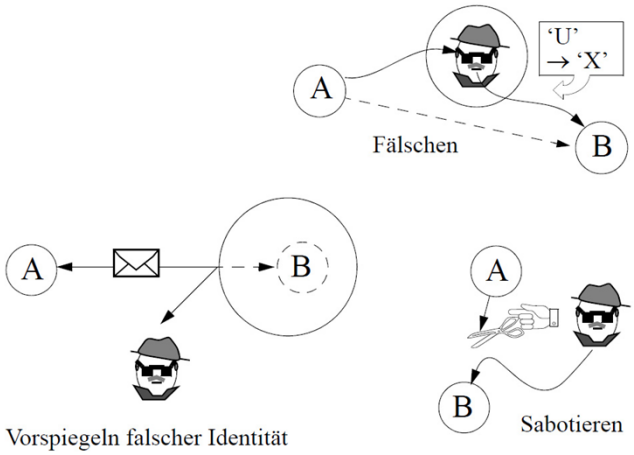
Sicherheit

Sicherheit in verteilten Systemen

Problemaspekte und Angriffszenarien:



Sicherheit in verteilten Systemen (2)



Sicherheit: Anforderungen

- Autorisierung / Zugriffsschutz
  - Einschränkung der Nutzung auf den Kreis der Berechtigten
- Vertraulichkeit
  - Daten / Nachrichteninhalt gegen Lesen Unberechtigter schützten
  - Kommunikationsverhalten (wer mit wem etc.) geheim halten
- Authentizität
  - Absender "stimmt" (z.B. Server ist der, für den er sich ausgibt)
  - Daten sind "echt" und aktuell (→ Integrität)
- Integrität
  - Wahrung der Unversehrtheit von Nachrichten, Programmen, Daten
- Verfügbarkeit der wichtigsten Dienste
  - keine Zugangsbehinderung ("denial of service") durch andere
  - kein provozierter Absturz ("Sabotage")

Weitergehende Anforderungen

Wie zum Beispiel:

- Nichtabstreitbarkeit
- Accountability
- Strafrechtliche Verfolgbarkeit
  - Protokollierung
  - „Key Escrow"
- Konformität zu rechtlich / politischen Vorgaben
- ...

Sicherheit: Verteilungsaspekte

- Offenheit in verteilten Systemen begünstigt Angriffe
  - grosse Systeme → vielfältige Angriffspunkte
  - standardisierte Kommunikationsprotokolle → Angriff einfach
  - räumliche Distanz → Ortung des Angreifers schwierig, Angriff sicher
  - breiter Einsatz, allgemeine Verwendung → Angriff reizvoller
  - physische Abschottung oft nicht durchsetzbar
  - technologische Gegebenheiten: z.B. Wireless LAN ("broadcast")
- Heterogenität
  - sorgt für zusätzliche Schwachstellen
  - erschwert Durchsetzung einer einheitlichen Schutzphilosophie
- Dezentralität
  - fehlende netzweite Sicherheitsautorität

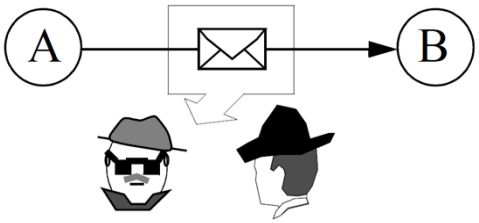
Sicherheit: Verteilungsaspekte

- → Gewährleistung der Sicherheit ist in verteilten Systemen wichtiger und schwieriger als in alleinstehenden Systemen
- Typische Techniken und "Sicherheitsdienste":
  - *Verschlüsselung*
  - *Autorisierung* ("der darf das!")
  - *Authentisierung* ("X ist wirklich X!")

Hierfür Kryptosysteme und Protokolle als "Security Service"

Angriffsformen – Passive Angriffe

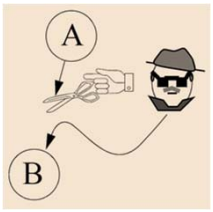
- Beobachten der Kommunikation
  - Nachrichteninhalt in Erfahrung bringen
  - Kommunikationsverhalten analysieren ("wer mit wem wie oft?")



→ Verschlüsselung
→ Anonymisierung

Aktive Angriffe: Eindringen, Vorsätzliche Täuschung etc.

- Einbruch (Zugangsschranken durchbrechen)
  - Diebstahl von Daten, Nachrichteninhalten,…
- Verändern des Nachrichtenstroms
  - Ändern, Vernichten, Erzeugen, Vertauschen, Verzögern, Wiederholen ("replay") von Nachrichten



- Falsche Identität vorspiegeln
  - Maskerade: Nachahmen anderer Prozesse oder Nutzung eines fremden Passwortes
- Missbräuchliche Nutzung von Diensten
- Denial of Service durch Sabotage oder Verhindern des Dienstzugangs, z.B. durch Überfluten mit Nachrichten

Authentizität

> Seid auf eurer Hut vor dem Wolf; wenn er hereinkommt, so frisst er euch alle mit Haut und Haar. Der Bösewicht verstellt sich oft, aber an seiner rauen Stimme und seinen schwarzen Füssen werdet ihr ihn gleich erkennen.
>
> (*Der Wolf und die sieben Geisslein*, grimmsche Märchen)

- Authentizität ist elementar und essentiell für die Sicherheit eines verteilten Systems
  - zu authentischen Nachrichten / Daten vgl. auch "Integrität"
- Authentizität eines Subjekts
  - Kommunikationspartner, Client, Server,…
  - ist der andere wirklich der, der er vorgibt zu sein?
  - z.B.: darf ich als Server daher ihm (?) den Zugriff gewähren?

Authentizität (2)

- Authentizität eines Dienstes
  - Bsp.: Handelt es sich wirklich um den Druckdienst oder um einen böswilligen Dienst, der die Datei ausserdem noch heimlich kopiert?
- Authentizität einer Nachricht
  - hat mein Kommunikationspartner dies wirklich so gesagt?
  - soll ich als Geldautomat wirklich so viel Geld ausgeben?
- Authentizität gespeicherter Daten
  - ist dies wirklich der Vertragstext, den wir gemeinsam elektronisch hinterlegt haben?
  - hat der Autor Casimir von Hinkelstein wirklich *das* geschrieben?
  - ist das Foto nicht eine Fälschung?
  - ist dieser elektronische Schlüssel wirklich echt?

Hilfsmittel zur Authentifizierung

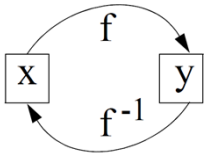
- Wahrung der Nachrichten-Authentizität
  - Verschlüsselung, so dass inhaltliche Änderungen auffallen (Signatur, kryptograph. Hashcode / Prüfsumme)
  - Beachte: Authentizität des Nachrichteninhalts garantiert nicht Authentizität der Nachricht als solche! (z.B. Replay-Attacke: Neuversenden einer früher abgehörten Nachricht)
  - Massnahmen gegen Replays: z.B. mitcodierte Sequenznummer
- Peer-Authentifizierung z.B. mit Frage-Antwort-Spiel
  - "challenge / response": Antworten sollte nur der echte Kommunikationspartner kennen
  - idealerweise stets neue Fragen verwenden (Replay-Attacken!)

Passwörter zur Authentifizierung

- Authentifizierung mit Passwort
  - typischerweise zur Authentifizierung eines Benutzers ("Client") zum Schutz des Dienstes vor unbefugter Benutzung (Autorisierung)
  - Kenntnis des Passworts gilt als Identitätsbeweis (gerechtfertigt?)
- Potentielle Schwächen von Passwörtern
  - Geheimhaltung (Benutzer kann Passwörter "verleihen" etc.)
    - Angriffsformen: Raten bzw. systematische Suche ("dictionary attack")
    - Gegenmassnahmen: Zurückweisung zu "simpler" Passwörter,
    - Zeitverzögerung nach jedem Fehlversuch und Maximalzahl von Fehlversuchen,
    - security logs
  - Abhörgefahr (möglichst kein Übermittlung im Klartext; Speicherung nur in codierter Form, so dass Invertierung prakt. unmöglich)
  - Replay-Attacke (Gegenmassnahme: Einmalpasswörter)

Einwegfunktionen

- Bilden die Basis vieler kryptographischer Verfahren
- Prinzip: **y = f(x)** einfach aus x berechenbar, aber **x = f^{-1}(y)** ist extrem schwierig aus y zu ermitteln
  - f^{-1} typw. exponentielle Zeitkomplexität
  - zeitaufwändig (→ praktisch undurchführbar)



z.B. f = O(n), O(n log n),...
aber f^{-1} = $O(2^n)$

- Es gibt (noch) keinen mathematischen Beweis, dass Einwegfunktionen überhaupt existieren
  - aber einige Funktionen sind es *allem Anschein nach*

Einwegfunktionen (2)

- Einwegfunktionen mögen zunächst sinnlos erscheinen :

  Ein zu y = f(x) verschlüsselter Text x kann nie wieder entschlüsselt werden!

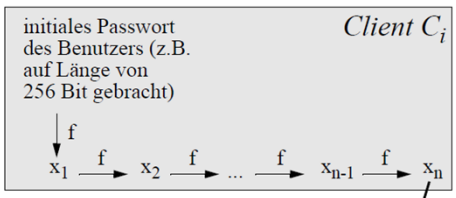
- Aber: Einwegfunktionen mit "trap-door"
  (ein Geheimnis, das es erlaubt, f^{-1} effizient zu berechnen)
  - Idee: Nur der "Besitzer" oder "Erfinder" von f kennt dieses
  - Beispiel Briefkasten: Einfach etwas hineinzutun; schwierig etwas herauszuholen; mit Schlüssel (= Geheimnis) ist das aber einfach!
  - Anwendung z.B. Public-Key-Verschlüsselung oder Einmalpasswörter

Prinzipien typischer (vermuteter) Einwegfunktionen

- Das Multiplizieren zweier (grosser) Primzahlen p, q ist effizient; das Zerlegen einer Zahl (z.B. n = pq) in ihre Primfaktoren i.Allg. schwierig (d.h., sehr aufwändig)

- In einem Restklassenring (mod m) ist die Bildung der Potenz a^k einfach; die k-te Wurzel oder den (diskreten) Logarithmus zu berechnen, ist i.Allg. schwierig.
  - aber: k-te Wurzel einfach, wenn Primzerlegung von m = pq bekannt → trap-door!

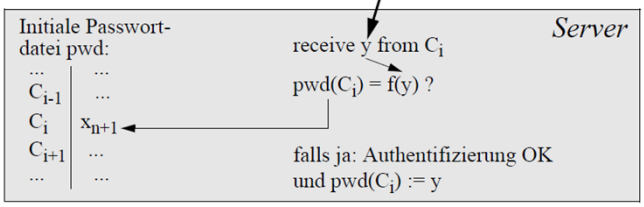
Einmalpasswörter mit Einwegfunktionen



Durch iterierte Anwendung einer (bijektiven) Einwegfunktion f wird (quasi auf Vorrat) eine Liste von n Einmalpasswörtern x_1 ... x_n erzeugt

Kommunikation über das Netz ist unsicher!

Zunächst wird x_n verwendet, beim nächsten Mal x_{n-1}, dann x_{n-2} etc.

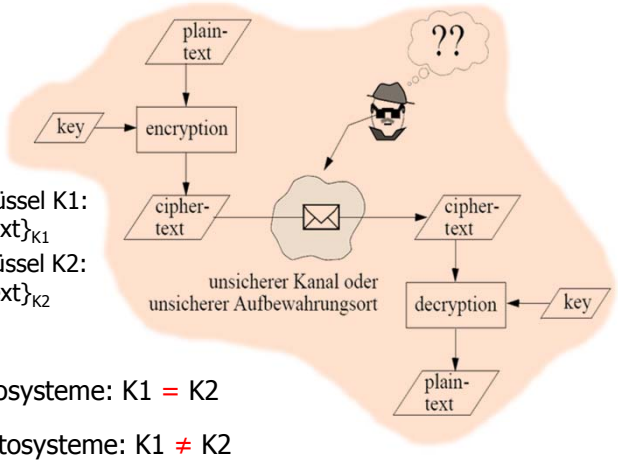


Einmalpasswörter mit Einwegfunktionen: Eigenschaften

- Ein abgehörtes Passwort x_i nützt nicht viel
  - Berechnung von x_{i-1} aus x_i ist (praktisch) nicht möglich
- Ein Lesen der Passwortdatei des Servers ist nutzlos
  - dort ist nur das *vergangene* Passwort vermerkt
- Einwegfunktion f muss nicht geheimgehalten werden

Verfahren ist z.B. im S/KEY-System (RFC 1760) realisiert

Kryptosysteme



Schreibweisen:

- Verschlüsseln mit Schlüssel K1:
  Schlüsseltext = $\{Klartext\}_{K1}$
- Entschlüsseln mit Schlüssel K2:
  Klartext = $\{Schlüsseltext\}_{K2}$

- Symmetrische Kryptosysteme: K1 = K2
- Asymmetrische Kryptosysteme: K1 ≠ K2

Kryptosysteme (2)

- Forderung an das Verfahren:
  Verschlüsselung ist ohne Kenntnis der Schlüssel höchstens mit unverhältnismässig hohem Rechenaufwand umkehrbar

- Geheimhalten eines konkreten Verschlüsselungsverfahrens stellt i.Allg. keinen Sicherheitsgewinn dar
  - organisatorisch oft nicht lange durchhaltbar
  - kein öffentliches Feedback über erkannte Schwächen des Verfahrens
  - Verfahren, die Geheimhaltung nötig hätten, erscheinen "verdächtig"

Symmetrische Schlüssel

- Nachteile symmetrischer Schlüssel:
  - Schlüssel muss geheimgehalten werden (da Verfahren i.Allg. bekannt)
  - mit allen Kommunikationspartnern separaten Schlüssel vereinbaren
  - hohe Komplexität der Schlüsselverwaltung bei vielen Teilnehmern
  - Problem des geheimen Schlüsselaustausches
- Vorteile symmetrischer Schlüssel:
  - ca. 100 bis 1000 Mal schneller als typ. asymmetrische Verfahren
- Beispiele für symmetrische Verfahren:
  - IDEA (International Data Encryption Algorithm): 128-Bit Schlüssel, Einsatz in PGP
  - DES (Data Encryption Standard) bzw. „Triple DES" (TDES)
  - AES (Advanced Encryption Standard) als Nachfolger von DES

One-Time Pads

- "Perfektes" symmetrisches Kryptosystem
  - Denkübung: Wieso und unter welchen Voraussetzungen?

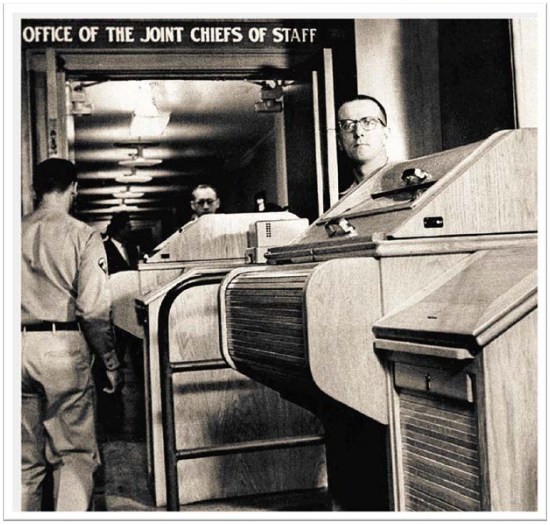
Klartext	V	E	R	T	E	I	L	T	E		S	Y	S	T	E	M	E
in ASCII	56	45	52	54	45	49	4C	54	45	20	53	59	53	54	45	4D	45
								XOR									
Schlüssel	4C	93	EF	20	B7	55	92	7C	DA	69	23	F8	BB	72	0E	81	00
= Chiffre	1A	D6	BD	74	F2	1C	DE	28	9F	49	70	A1	E8	26	4B	CC	45

- Prinzip: Wähle zufällige Sequenz von Schlüsselbits
  - Verschlüsselung: Schlüsseltext = Klartext XOR Schlüsselbitsequenz
  - Entschlüsselung: Klartext = Schlüsseltext XOR Schlüsselbitsequenz
  - Begründung: (a XOR b) XOR b = a (für alle Bitbelegungen von a, b)

One-Time Pads (2)

- Anforderungen an Schlüsselbitsequenz:
  - keine Wiederholung von Bitmustern (→ Schlüssellänge = Klartextlänge)
  - Schlüsselbitsequenz ohne Bildungsgesetz ("echte" Zufallsfolge)
  - Schlüsselbitsequenz wirklich "one-time" (keine Mehrfachverwendung!)
- Kryptoanalyse ohne Schlüsselkenntnis unmöglich
- Nachteile von One-Time Pads:
  - Verwendung unhandlich (hoher Bedarf an frischen Schlüsselbits, dadurch aufwändiger Schlüsselaustausch)
  - Synchronisationsproblem bei Übertragungsstörungen (wenn Empfang ausser Takt gerät, ist Folgetext verloren)
  - → nur für hohe Sicherheitsanforderungen üblich (z.B. "rotes Telefon")

Rotes „Telefon"



Der „heisse Draht" zwischen Washington und Moskau wird am 26. August 1963 installiert; es handelt sich um Fernschreibeinrichtungen (Telex) mit Kabel via London und Helsinki.

Ständige Prüfnachricht war „the quick brown fox jumps over the lazy dog 1234567890".

Ab 1971 auch Kommunikation via Satellit, seit 2008 auch Glaserfaserkabel.

Seit 1980 wird Fax benutzt, seit 2010 auch E-Mail.

Rotes „Telefon“

Der „heisse Draht“ zwischen Washington und Moskau wird am 26. August 1963 installiert; es handelt sich um Fernschreibeinrichtungen (Telex) mit Kabel via London und Helsinki.

Ständige Prüfnachricht war „the quick brown fox jumps over the lazy dog 1234567890“.

Ab 1971 auch Kommunikation via Satellit, seit 2008 auch Glaserfaserkabel.

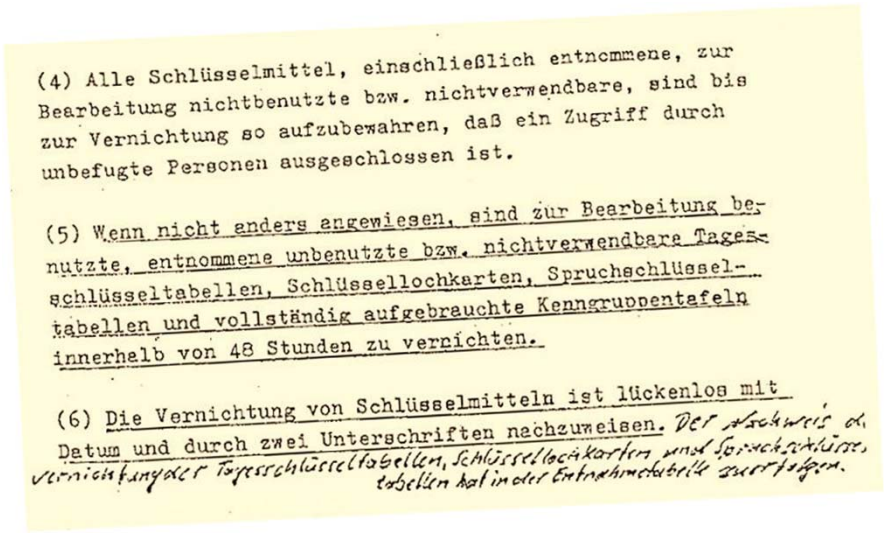
Seit 1980 wird Fax benutzt, seit 2010 auch E-Mail.

Die Kenngruppen dürfen nur einmal zur Bildung eines Spruchschlüssels verwendet werden.

(1) Der Tagesschlüssel ist ein Zeitschlüssel und hat jeweils 24 Stunden Gültigkeit. Der Schlüsselwechsel erfolgt 00.01 Uhr.

(2) Der Spruchschlüssel hat nur Gültigkeit für jeweils einen Spruch.

(3) Tagesschlüsseltabellen und Schlüssellochkarten, die auf Grund von Beschädigungen oder Kompromittierung nicht benutzt wurden, sind der Leitstelle (für die Chiffrierverbindung verantwortliche Chiffrierstelle) des Schlüsselbereiches chiffriert oder mittels Kurier zu melden. Der neue Tagesschlüssel (bzw. die neue Folge der Tagesschlüssel) wird durch die Leitstelle chiffriert oder mittels Kurier angewiesen.



(4) Alle Schlüsselmittel, einschließlich entnommene, zur Bearbeitung nichtbenutzte bzw. nichtverwendbare, sind bis zur Vernichtung so aufzubewahren, daß ein Zugriff durch unbefugte Personen ausgeschlossen ist.

(5) Wenn nicht anders angewiesen, sind zur Bearbeitung benutzte, entnommene unbenutzte bzw. nichtverwendbare Tagesschlüsseltabellen, Schlüssellochkarten, Spruchschlüsseltabellen und vollständig aufgebrauchte Kenngruppentafeln innerhalb von 48 Stunden zu vernichten.

(6) Die Vernichtung von Schlüsselmitteln ist lückenlos mit Datum und durch zwei Unterschriften nachzuweisen. Der Nachweis d. Vernichtung der Tagesschlüsseltabellen, Schlüssellochkarten und Spruchschlüsseltabellen hat in der Entnahmestelle zu erfolgen.

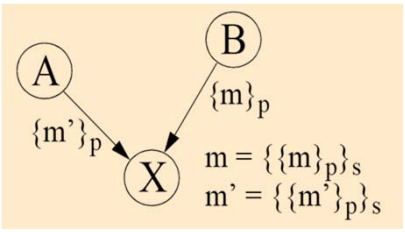
Asymmetrische Kryptosysteme

Schlimm sind die Schlüssel,
die nur schliessen auf, nicht zu;
Mit solchem Schlüsselbund im Haus
verarmest du.
Friedrich Rückert, Die Weisheit des Brahmanen

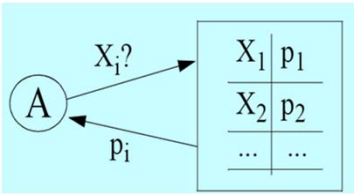
- Schlüssel zum Ver- / Entschlüsseln sind verschieden
  - z.B. RSA-Verfahren (Rivest, Shamir, Adleman, 1978), beruht auf der Schwierigkeit von Faktorisierung
  - andere Verfahren beruhen z.B. auf diskreten Logarithmen
- Für jeden Prozess X existiert ein Paar (p,s)
  - p = *public key* (zum *Verschlüsseln* von Nachrichten an X)
  - s = *secret key* (oder: private key) (zum *Entschlüsseln* von mit p verschlüsselten Nachrichten)

Asymmetrische Kryptosysteme

- Jeder Prozess, der an X sendet, kennt p
- Nur X selbst kennt s

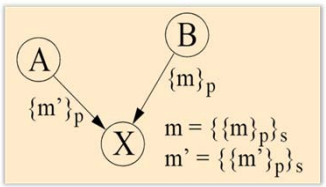


- Public-Key-Server
  - Welchen Schlüssel hat Prozess X_i?
  - Server muss vertrauenswürdig sein
  - Kommunikation zum Server darf nicht manipuliert sein



Asymmetrische Kryptosysteme

Gewünschte Eigenschaften:



- 1) m lässt sich nicht allein aus $\{m\}_p$ ermitteln
- 2) s lässt sich aus p oder einer verschlüsselten, bekannten Nachricht nicht (mit vertretbarem Aufwand) ableiten
- 3) $m = \{\{m\}_p\}_s$
- 4) Evtl. zusätzlich: $m = \{\{m\}_s\}_p$
  (Rolle von Verschlüsselung und Entschlüsselung austauschbar)
- Resistenz gegen „Chosen-Plaintext"-Angriff (beliebige Nachrichten M und deren Verschlüsselung $\{M\}_p$ sind jederzeit generierbar)

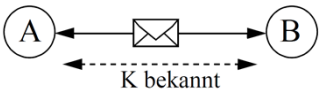
Vorteil asymmetrischer Verfahren gegenüber symmetrischen

- Vereinfachter Schlüsselaustausch
  - jeder darf den übermittelten public key p mithören
  - secret key s braucht grundsätzlich nie Dritten mitgeteilt zu werden
  - bei n Teilnehmern genügen 2n Schlüssel (statt $O(n^2)$ bei sym. Verf.)
- Kenntnis von s authentifiziert zugleich den Besitzer
  - "wer $\{M\}_{pA}$ entschlüsseln kann, der ist wirklich A" (wirklich?)

sA bzw. pA secret bzw. public key von A

- Digitale Unterschrift
  - "wenn (zu M) ein $\{M\}_{sA}$ existiert mit $\{M\}_{sA}\}_{pA} = M$, dann muss dies (M bzw. $\{M\}_{sA}$) von A erzeugt worden sein" (wieso?)

Authentifizierung mit symmetrischen Schlüsseln



Sei K der zwischen A und B vereinbarte (und geheimzuhaltende!) Schlüssel

- Problem: B soll die Authentizität von A feststellen
  - *Idee* (Geheimdienstprinzip): "Wenn X *das* weiss und kann, dann muss X wirklich X sein, denn sonst weiss und kann das niemand"
  - *Bemerkung:* Oft ist eine gegenseitige Authentifizierung nötig
- 1. Verfahren:

A: m := "Ich bin A"
m' := $\{m\}_K$

A→B: m', m

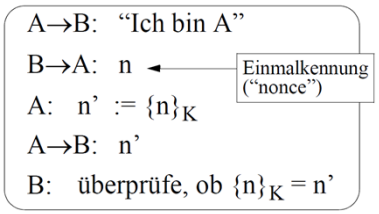
B: überprüfe, ob $\{m\}_K$ = m'

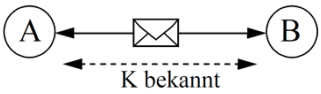
Damit B den richtigen Schlüssel (für A) wählt

- *Idee*: Überprüfe die Fähigkeit, Nachrichten mit einem geheimen Schlüssel zu kodieren
- *Nachteil*: Möglichkeit von replays durch Abhören

Authentifizierung mit symmetrischen Schlüsseln

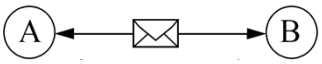
- 2. Verfahren:



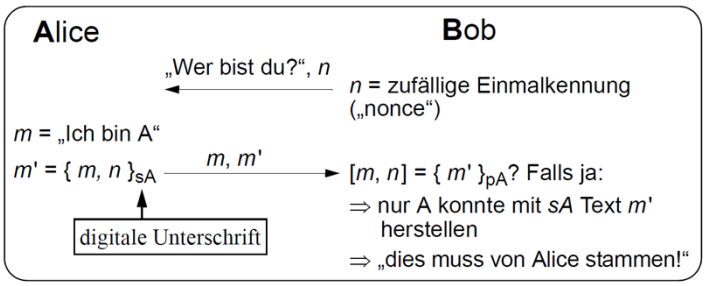


Genereller Nachteil bei symmetrischen Schlüsseln: Viele individuelle Schlüsselpaare für jede Client / Server-Beziehung

Authentifizierung mit asymmetrischen Schlüsseln



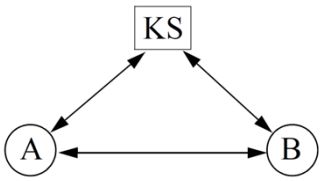
Notation: sX = secret key von X;
pX = public key von X



- Geschützt gegen Replays (wieso?)
- Vorsicht: "Man in the middle"-Angriff möglich (wie?)

Einbezug eines Schlüsselservers

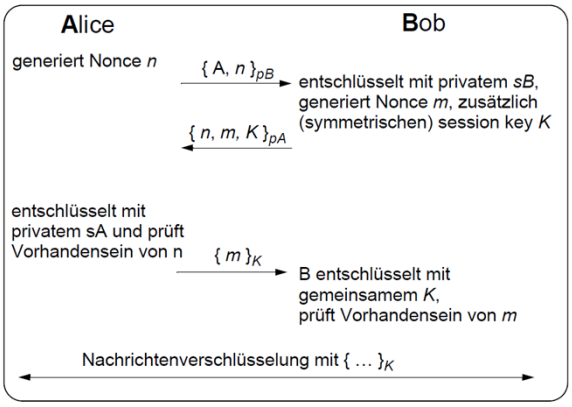
- Nachteil obiger Lösung: B muss viele public keys speichern
- Alternativ mit Key Server KS: dieser kennt alle public keys
  - B erfragt public key von A bei KS
  - KS signiert alle seine Nachrichten
  - jeder kennt public key von KS (um Unterschrift von KS zu verifizieren)



- Angriff auf den Schlüsselserver KS liefert keine Geheimnisse; erlaubt aber u.U., in dessen Rolle zu schlüpfen und falsche Auskünfte zu geben!
- KS ist evtl. repliziert oder verteilt

Gegenseitige Authentifizierung (mit Schlüsselvereinbarung)

- Im Prinzip wie oben beschrieben nacheinander in beide Richtungen möglich; effizienter beides zusammen erledigen



- Voraussetzung: A und B kennen die public keys pB bzw. pA des jeweiligen Partners
- Hier zusätzlich: Vereinbarung eines symmetrischen "session keys" K, der nach der Authentifizierung zur effizienten Verschlüsselung benutzt wird

Replays

- Generelles Problem: Auch wenn ein Angreifer eine Nachricht nicht entschlüsseln kann, kann er sie dennoch evtl. kopieren und später wieder einspielen
  - Autorisierungscodes für Geldautomaten,...

Mögliche Gegenmassnahmen:

1) Verwendung von Einmalkennungen, die vom Empfänger vorgegeben werden ("nonce")
  - etwas aufwändigeres Challenge-Response-Protokoll
  - → alle relevanten Nachrichten sind verschieden

Replays (2)

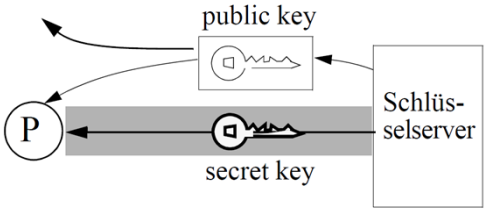
2) Verwendung von mitkodierten Sequenznummern
  - nur bei einer Nachrichtenfolge zwischen 2 Prozessen möglich

3) Mitverschlüsseln der Absendezeit

- Empfänger akzeptiert Nachricht nur, wenn seine Zeit max. Δt abweicht
  - globaler Zeitservice bzw. gut synchronisierte Uhren nötig
  - Angreifer darf Zeitservice nicht manipulieren können
- Zeitfenster Δt geschickt wählen
  - Nachrichtenlaufzeiten berücksichtigen
  - *zu gross* → unsicher durch mögliche Replays
  - *zu klein* → falscher Alarm

Schlüsselvergabe (public key)

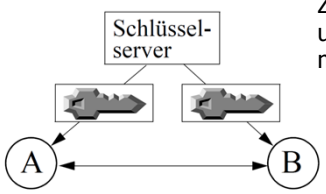
- Zur Erzeugung und Verwaltung von Schlüsseln existiert typischerweise ein eigener Dienst (mit Schlüsselserver)
- Zum Bsp. Vergabe eines Paares von public / secret keys:



  - Secret key muss auf sicherem Kanal zu P gelangen
  - Public key von P kann an beliebige Prozesse offen verteilt werden (jedoch i.Allg. "zertifiziert", dass der Schlüssel authentisch ist)

Schlüsselvergabe (session key)

- Zur Generierung von temporären symmetrischen Schlüsseln ("session key"):



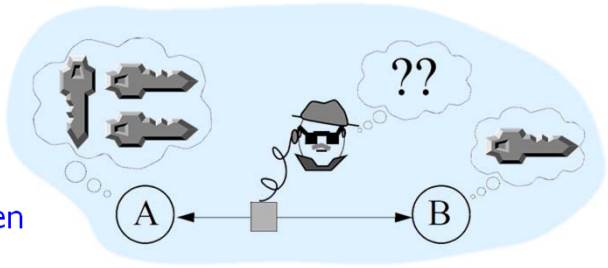
Z.B. bittet A den Schlüsselserver, ihm und B einen session key für die Kommunikation zwischen ihnen zu schicken

Session keys werden sicher und authentisch, z.B. mit einem Public-Key-Verfahren, an die Kommunikationspartner übertragen

  - Schlüsselserver kann session key nach Übertragung bei sich löschen
  - Aufwändiges Public-Key-Verfahren nur ein einziges Mal pro "Session", tatsächliche Nachrichtenverschlüsselung auf dem Kanal zwischen A und B dann effizient per symmetrischem Schlüssel

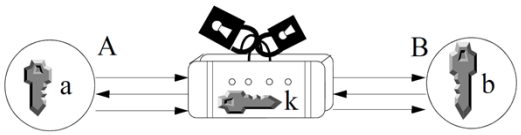
Direkte Schlüsselvereinbarung?

- Problem: A und B wollen sich ohne Schlüsselserver über einen unsicheren Kanal auf einen (geheimen) gemeinsamen Schlüssel einigen



- Sinnvoll z.B. bei dynamisch gegründeten Prozessen, die vorher noch nie kommuniziert haben
  - z.B. wenn keine public keys vorhanden bzw. nicht bekannt
- Wie geht dies?
  - wir erinnern uns an die "Schatzkiste mit zwei Vorhängeschlössern"

Direkte Schlüsselvereinbarung?



1. A generiert einen Sitzungsschlüssel k
2. A verschlüsselt k mit einem geheimen Schlüssel a
3. A→B: $\{k\}_a$ — a und b sind "lokal erfunden"
4. B verschlüsselt dies mit seinem Schlüssel b
5. B→A: $\{\{k\}_a\}_b$
6. A entschlüsselt mit seinem Schlüssel a:
   $\{\{\{k\}_a\}_b\}_{\bar{a}} = \{\{\{k\}_a\}_{\bar{a}}\}_b = \{k\}_b$
   Forderung! — Bezeichne $\bar{x}$ den zu x inversen Schlüssel (oft: $\bar{x} = x$)
7. A→B: $\{k\}_b$ — gemeinsames Geheimnis
8. B entschlüsselt mit seinem Schlüssel: $\{\{k\}_b\}_{\bar{b}} = k$

Beachte: k wird nie offen transportiert!

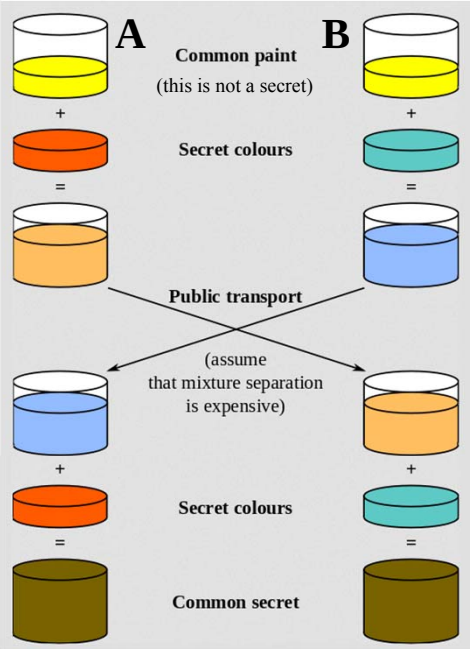
Frage: Geht hier xor mit "one-time pads" a, b ?
- xor erfüllt die Forderung (ist assoziativ und kommutativ)
- xor mit one-time pads ist sicher (wirklich?) und effizient

- Aber: Wenn Schritt 3 $\{k\}_a$ und Schritt 5 $\{\{k\}_a\}_b$ abgehört wird, dann kann daraus der Schlüssel b ermittelt werden, so dass in Schritt 7 aus dem abgehörten $\{k\}_b$ das geheime k ermittelt werden kann!

- Geht anstelle von xor etwas anderes, das sicher ist?
  → Idee des Diffie-Hellman-Verfahrens

Diffie-Hellman-Verfahren

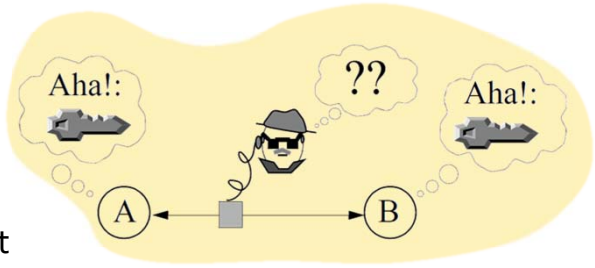
- Hier als Gleichnis in Form von Farbenmischen
- Ziel: A und B sollen sich über einen unsicheren Kanal auf ein gemeinsames "Geheimnis" (hier: eine Farbe) einigen, ohne dass ein Angreifer es erfährt
  - Auch wenn der Angreifer aus den beiden übermittelten Farben eine Probe mischt?



Bildquelle: Wikipedia

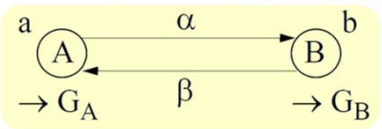
Schlüsselvereinbarung mit dem Diffie-Hellman-Verfahren

- Ziel: A und B sollen sich über einen unsicheren Kanal auf ein gemeinsames "Geheimnis" G einigen, ohne dass ein Angreifer es erfährt



- Nutzung einer Einwegfunktion: $f(x) = c^x \bmod p$
  (1 < c < p; wobei p eine grosse Primzahl ist)
  - in einem Restklassenring ist die Bestimmung diskreter Logarithmen (als Umkehrfunktion) viel schwieriger als die Bildung von Potenzen

Der Diffie-Hellman-Algorithmus



- wenig Nachrichten
- effizient

1. A wählt eine Zufallszahl a
2. A berechnet $\alpha = f(a)$
3. A → B: α
4. B wählt eine Zufallszahl b
5. B berechnet $\beta = f(b)$
6. B → A: β
7. A berechnet $G_A = \beta^a \bmod p$
8. B berechnet $G_B = \alpha^b \bmod p$

Bem.: a und b sind nur lokal bekannt und bleiben geheim

Behauptung: $G_A = G_B$
(gemeinsames Geheimnis!)

Beispiel (für c = 5 und unrealistisch kleines p = 7):
$f(x) = 5^x \bmod 7$
$a = 3 \rightarrow \alpha = 6$
$b = 4 \rightarrow \beta = 2$
$\rightarrow G_B = 6^4 \bmod 7 = 1$
$\rightarrow G_A = 2^3 \bmod 7 = 1$

$G_A = G_B$

Zu zeigen: $\beta^a \bmod p = \alpha^b \bmod p$, also:

$$(c^b \bmod p)^a \bmod p = (c^a \bmod p)^b \bmod p$$

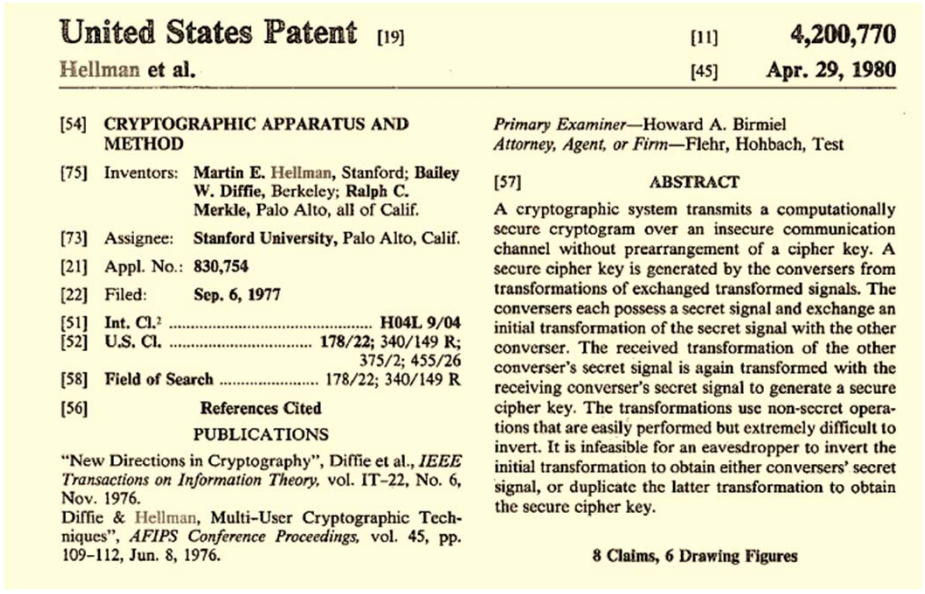
Lemma: $(k \bmod p)^n \bmod p = k^n \bmod p$ ← Restklassen-arithmetik...

$$\begin{aligned}(c^b \bmod p)^a \bmod p &= (c^b)^a \bmod p \quad \text{[Lemma]}\\ &= c^{(b\,a)} \bmod p\\ &= c^{(a\,b)} \bmod p\\ &= (c^a)^b \bmod p \quad \text{[Lemma]}\\ &= (c^a \bmod p)^b \bmod p\end{aligned}$$

Bemerkungen:
- Lässt sich auch auf k > 2 Benutzer verallgemeinern
- Der Algorithmus (entdeckt 1976) ist patentiert (U.S.-Patent Nummer 4200770, Sept. 1977)

Sweet Little Secret G

- A und B könnten $G = G_A = G_B$ nun als symmetrischen Schlüssel zur Kodierung ihrer Nachrichten verwenden
  - Besser: G nur als „master key" verwenden, um daraus (mittels Einwegfunktion) einen session key zu erzeugen
  - Motivation: G selbst so selten wie möglich benutzen
- Einzusehen bliebe noch, dass aus Kenntnis von α und β (sowie von c und p aus f) G von einem Mitlauscher (in effizienter Weise) nicht ermittelt werden kann
  - α und β sind unabhängig voneinander (wieso ist das ein Argument?)
  - Bem.: nicht jedes p ist „gut" und sollte auch einige 100 Bit lang sein



United States Patent [19]

Hellman et al.

[11] **4,200,770**

[45] **Apr. 29, 1980**

[54] **CRYPTOGRAPHIC APPARATUS AND METHOD**

[75] Inventors: **Martin E. Hellman**, Stanford; **Bailey W. Diffie**, Berkeley; **Ralph C. Merkle**, Palo Alto, all of Calif.

[73] Assignee: **Stanford University**, Palo Alto, Calif.

[21] Appl. No.: **830,754**

[22] Filed: **Sep. 6, 1977**

[51] **Int. Cl.**[2] **H04L 9/04**

[52] **U.S. Cl.** ..... **178/22**; 340/149 R; 375/2; 455/26

[58] **Field of Search** ..... 178/22; 340/149 R

[56] **References Cited**

PUBLICATIONS

"New Directions in Cryptography", Diffie et al., *IEEE Transactions on Information Theory*, vol. IT-22, No. 6, Nov. 1976.

Diffie & Hellman, Multi-User Cryptographic Techniques", *AFIPS Conference Proceedings*, vol. 45, pp. 109–112, Jun. 8, 1976.

Primary Examiner—Howard A. Birmiel
Attorney, Agent, or Firm—Flehr, Hohbach, Test

[57] **ABSTRACT**

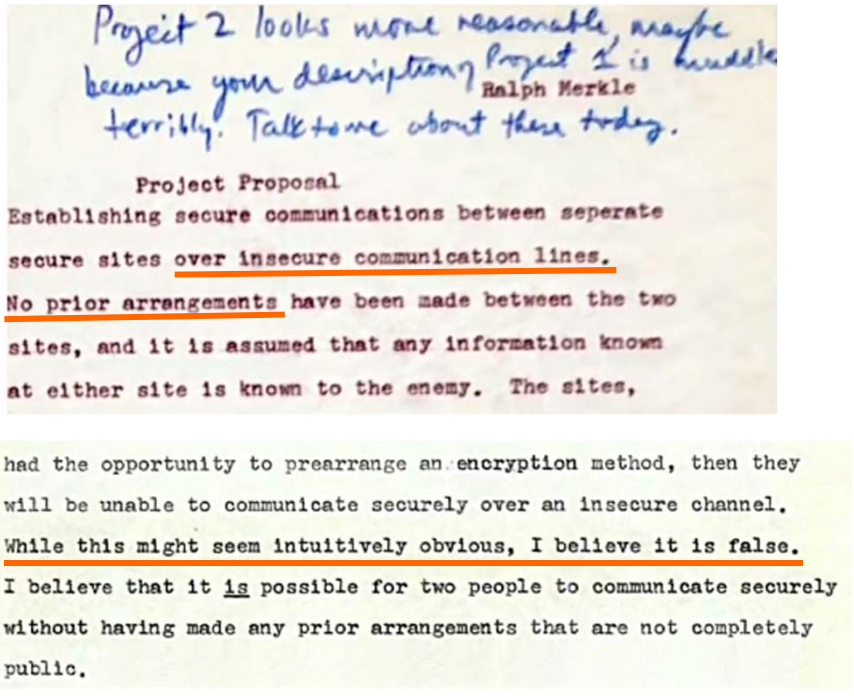
A cryptographic system transmits a computationally secure cryptogram over an insecure communication channel without prearrangement of a cipher key. A secure cipher key is generated by the conversers from transformations of exchanged transformed signals. The conversers each possess a secret signal and exchange an initial transformation of the secret signal with the other converser. The received transformation of the other converser's secret signal is again transformed with the receiving converser's secret signal to generate a secure cipher key. The transformations use non-secret operations that are easily performed but extremely difficult to invert. It is infeasible for an eavesdropper to invert the initial transformation to obtain either conversers' secret signal, or duplicate the latter transformation to obtain the secure cipher key.

8 Claims, 6 Drawing Figures

US4218582: Public key cryptographic apparatus and method
Issued/Filed Dates: Aug. 19, 1980 / Oct. 6, 1977

Abstract:
A cryptographic system transmits a computationally secure cryptogram that is generated from a publicly known transformation of the message sent by the transmitter; the cryptogram is again transformed by the authorized receiver using a secret reciprocal transformation to reproduce the message sent. The authorized receiver's transformation is known only by the authorized receiver and is used to generate the transmitter's transformation that is made publicly known. The publicly known transformation uses operations that are easily performed but extremely difficult to invert. It is infeasible for an unauthorized receiver to invert the publicly known transformation or duplicate the authorized receiver's secret transformation to obtain the message sent.

What is claimed is:
1. In a method of communicating securely over an insecure communication channel of the type which communicates a message from a transmitter to a receiver, the improvement characterized by: providing random numbers at the receiver; generating from said random numbers a public enciphering key at the receiver; generating from said random numbers a secret deciphering key at the receiver such that the secret deciphering key is directly related to and computationally infeasible to generate from the public enciphering key; communicating the public enciphering key from the receiver to the transmitter; processing the message and the public enciphering key at the transmitter and generating an enciphered message by an enciphering transformation, such that the enciphering transformation is easy to effect but computationally infeasible to invert without the secret deciphering key; transmitting the enciphered message from the transmitter to the receiver; and processing the enciphered message and the secret deciphering key at the receiver to transform the enciphered message with the secret deciphering key to generate the message.
2. ...

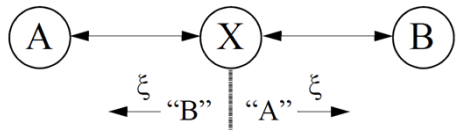


Project 2 looks more reasonable, maybe because your description of Project 1 is muddled terribly. Talk to me about these today.
Ralph Merkle

Project Proposal

Establishing secure communications between seperate secure sites over insecure communication lines. No prior arrangements have been made between the two sites, and it is assumed that any information known at either site is known to the enemy. The sites, had the opportunity to prearrange an encryption method, then they will be unable to communicate securely over an insecure channel. While this might seem intuitively obvious, I believe it is false. I believe that it is possible for two people to communicate securely without having made any prior arrangements that are not completely public.

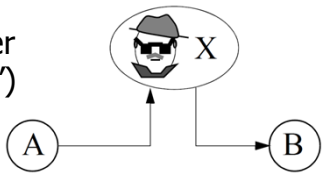
Ist das Diffie-Hellman-Prinzip gefeit gegen einen Eindringling?



- X ist ein sogen. „man in the middle"
  - mimt die Identität des jeweils anderen
- X kann unter Vortäuschung falscher Identitäten jeweils eigene Schlüssel für Teilstrecke AX bzw. XB vereinbaren!
  - „ein ξ für ein α bzw. β vormachen"

Aktive Angriffe: Man in the Middle

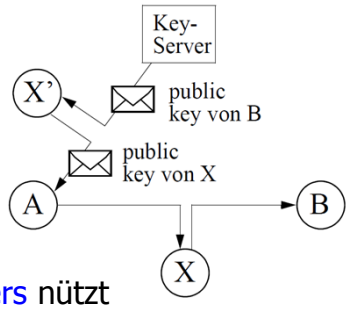
- Ein generelles Problem: X verhält sich gegenüber A wie B, gegenüber B wie A (→ X arbeitet "transparent")
  - z.B. eigene Schlüssel für die Teilstrecken vereinbaren



- Challenge-Response-Test nützt so nichts: X reicht Challenges einfach an den von ihm vorgetäuschten Partner weiter und mimt mit der abgefangenen Antwort die angenommene Identität

Aktive Angriffe: Schlüssel-fälschung beim Key-Server

- Kompromittierter Key-Server bzw. Verschwörung von X, X'
- X kann alle von A mit dem falschen Schlüssel verschlüsselten Nachrichten an B entziffern
  - X verschlüsselt danach die Nachricht neu mit dem richtigen Schlüssel für B
- Digitale Unterschrift des Key-Servers nützt nichts, wenn A den Prozess X' für den Key-Server hält und dessen Unterschrift akzeptiert

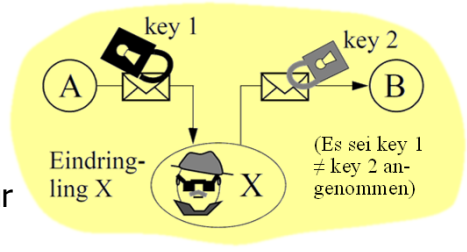


- Ist es überhaupt möglich, X in diesen Szenarien zu erkennen?
- Nützt die allgemeine Bekanntgabe des public keys des Key-Servers?

Erkennen von Eindringlingen?

Hier: Das sogen. „interlock protocol"

1) B stellt eine Anfrage, die nur A beantworten kann
2) A generiert die Antwort und verschlüsselt diese
3) A sendet zunächst aber nur die "Hälfte" davon zurück
   - z.B. nur jedes zweite Bit (also die "geraden" Bits)
   - B erwartet diese Hälfte der Antwort in weniger als t Zeiteinheiten
4) Ohne die andere Hälfte kann X dies nicht entschlüsseln und neu (mit key 2) verschlüsseln
5) Erst nach t Zeiteinheiten sendet A die andere Hälfte
   - B setzt Schlüsseltexthälften zusammen und überprüft Antwort



Das Dilemma des Eindringlings

- Gibt X die halbe Nachricht sofort unverändert weiter, kann B das Ganze nicht entschlüsseln → Fälschung erkannt
- Behält X die halbe Nachricht bis zum Eintreffen der anderen Hälfte, um alles zu entschlüsseln und neu mit key 2 zu verschlüsseln (und verzögert dann die Hälfte der ungeraden Bits um t Zeiteinheiten), dann arbeitet X nicht mehr zeittransparent → Eindringling vermutet
- Fragen: Wird in (in Schritt 1) nicht schon ein gemeinsames Geheimnis vorausgesetzt?
- Können (im Kontext des Diffie-Hellman-Verfahrens) A und B nicht dieses benutzen, um einen von X nicht ermittelbaren gemeinsamen Schlüssel zu finden? Oder genügt in 1) eine schwächere Eigenschaft ("originelle" Antwort; Fähigkeit, die nur A hat...)?

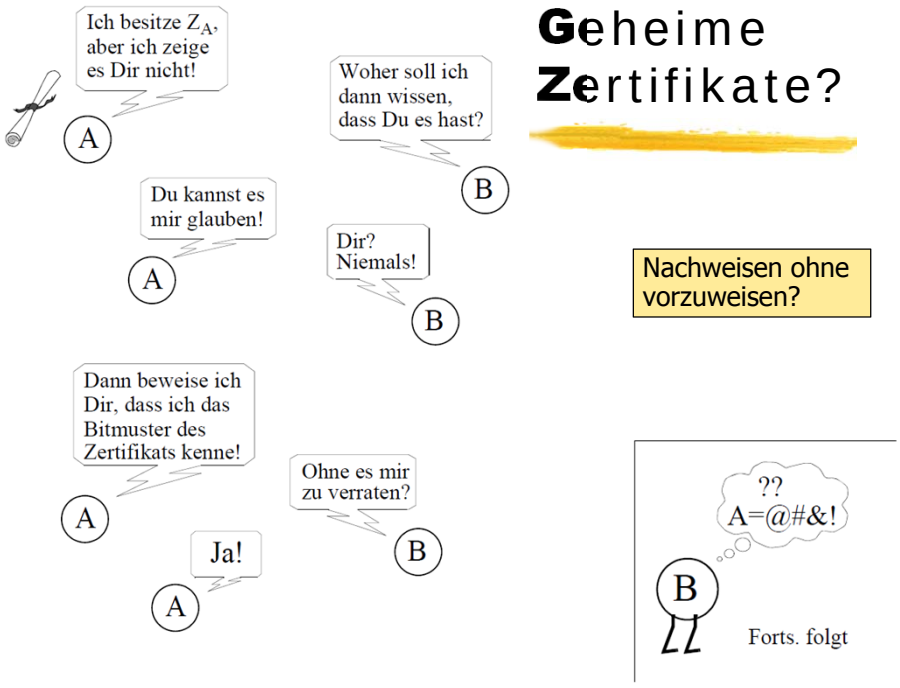
Zertifikate im gläsernen Tresor?

- Anschauen: ja
- Stehlen oder kopieren: nein

Authentifizierung mit geheimen Zertifikaten?

- A lässt sich von einer Autorität ein Zertifikat Z_A geben
  - Z_A soll von der Autorität signiert sein
  - Autorität gilt als vertrauenswürdig und hat A evtl. persönlich in Augenschein genommen (oder einem fremden Zertifikat vertraut)
- Wenn B an der Identität von A zweifelt, wird B von A auf sein Zertifikat Z_A hingewiesen
  - Besitz des Zertifikates = Authentifizierung
- Aber: A darf Z_A nie vorzeigen – sonst könnte B es sich kopieren und sich fortan als A ausgeben!
  - „Dokumente" (Bitfolgen) der digitalen Welt lassen sich perfekt kopieren
  - wie vermeidet man daher "raubkopierte Zertifikate"?
- Offenbar muss Z_A ein Geheimnis bleiben, das niemand ausser der Autorität und A kennt!
- Taugt ein solches Geheimnis als Zertifikat??
  - wie beweist man den Besitz eines Zertifikates, ohne es zu zeigen?

Geheime Zertifikate?

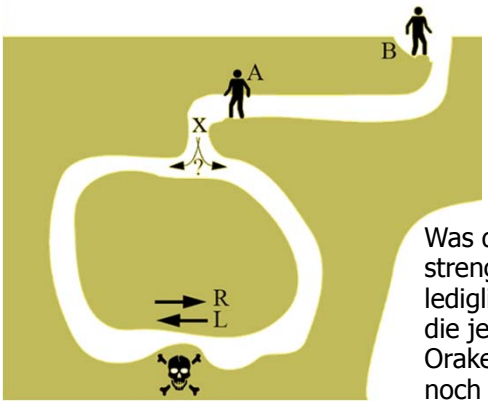


Nachweisen ohne vorzuweisen?

Geheime Zertifikate!

- Im Prinzip wissen wir schon, dass das geht:
  Der secret key sA eines asymmetrischen Verfahrens stellt ein solches Zertifikat dar
  - braucht von A nicht verraten zu werden
  - B kann dennoch überprüfen, ob A das Zertifikat hat (z.B. indem sich B von A etwas mit sA verschlüsseln lässt und anschliessend durch Anwenden von pA prüft; bzw. indem B eine Challenge $\{M\}_{pA}$ an A schickt und sich dies von A mit sA entschlüsseln lässt)
- Eine andere Realisierung geht mit "zero knowledge"
  - beweist Kenntnis eines Geheimnisses G, ohne relevante Information preiszugeben

Ein Höhlengleichnis

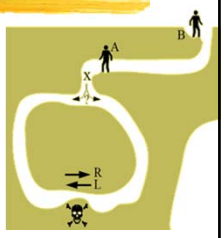


Der Höhlendämon lässt nur diejenigen die Engstelle lebendig passieren, die dort in der richtigen Richtung (L oder R) vorbeigehen.

Was die richtige Richtung ist, ist ein streng gehütetes Geheimnis; es ist lediglich bekannt, dass der Dämon die jeweilige Entscheidung einer Orakelbox entnimmt, die bisher noch niemand öffnen konnte.

- A sagt zu B: "Ich kenne das Geheimnis. Das beweise ich Dir, ohne das Geheimnis zu verraten!"

Ein Höhlengleichnis



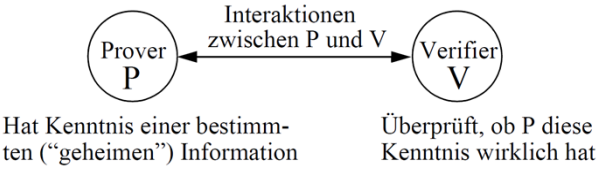
- A sagt zu B: "Ich kenne das Geheimnis. Das beweise ich Dir, ohne das Geheimnis zu verraten!"
- A begibt sich in die Höhle bis zur Engstelle; erst danach folgt B bis zur Stelle x
  - B weiss nicht, welche Richtung A dort eingeschlug
- B ruft A *entweder*
  - - "komm links heraus!" *oder*
  - - "komm rechts heraus!" zu
- A tut dies, indem A ggf. die Engstelle (in der richtigen Richtung) passiert
- A und B verlassen zusammen die Höhle
- Nachdem A das ganze n Mal überlebt hat, ist B überzeugt, dass A das Geheimnis (= Funktion der Orakelbox) kennt!

Die Irrtumswahrscheinlichkeit beträgt nur 2^{-n}

B hat in diesem "interaktiven Beweis" das Geheimnis nicht erfahren!
→ "Zero knowledge proof"

Zero-Knowledge-Beweis

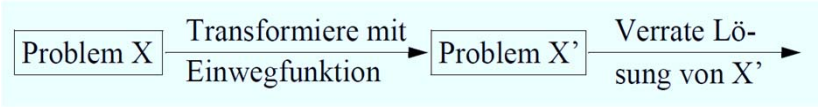
- "Beweis" = Nachweis, dass P eine bestimmte Folge von Bits (= Zahl, Algorithmus, Zertifikat,...) kennt



- P soll V (praktisch) nicht betrügen können: Wenn P die Information nicht hat, sollen seine Chancen, V zu überzeugen, verschwindend gering sein
  - V soll über die eigentliche Kenntnis von P nichts erfahren
  - V erfährt auch sonst nichts Relevantes von P, was V nicht auch alleine in Erfahrung bringen könnte

Zero-Knowledge-Beweis

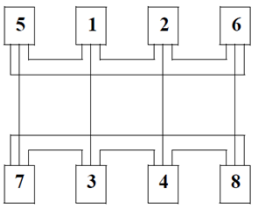
- Idee: geheime Information = Lösung eines schwierigen Problems X

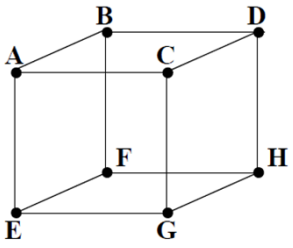


- Wobei die Lösung von X' die Lösung von X logisch impliziert, sie jedoch nicht effektiv-konstruktiv liefert

Beispiel: Isomorphie von Graphen

Bemerkung: Ob zwei grosse (z.B. in Form von Adjazenzmatrizen) gegebene Graphen G_1, G_2 topologisch isomorph ($G_1 \sim G_2$) sind (d.h. bis auf Umbenennung von Knoten und evtl. Kanten identisch sind), ist ein schwieriges Problem





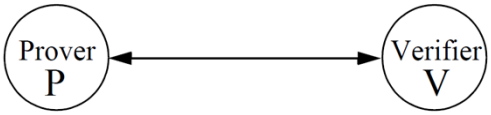
A = 7
B = 5
C = 8
D = 6
E = 3
F = 1
G = 4
H = 2

Hier nur ein kleines und daher einfaches (also unrealistisches) Beispiel

Überprüfung eines (durch eine Knotenzuordnung gegebenen) Isomorphismus ist allerdings "einfach"!

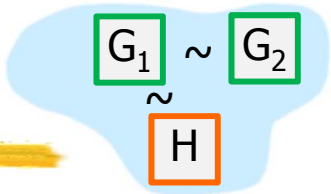
Zero-Knowledge bei Graphisomorphie

- P behauptet, einen Beweis zu haben, dass zwei gegebene Graphen G_1, G_2 isomorph sind, möchte den Beweis aber nicht verraten
  - etwa weil diese Isomorphie seinen „Identitätsausweis" darstellt

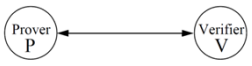


- Folgendes Protokoll überzeugt V davon, wobei der Isomorphismus selbst ein Geheimnis von P bleibt →

Zero-Knowledge bei Graphisomorphie



- P erzeugt durch zufällige Umbenennung der Knoten einen Graphen H mit $H \sim G_1$ (und damit $H \sim G_2$)
  - für P ist dies einfach
- P sendet H an V
- V bittet dann P, entweder $H \sim G_1$ nachzuweisen, oder $H \sim G_2$
- Da P den Graphen H konstruiert hat, kann P das Gewünschte einfach tun
  - Für Andere aber ist $H \sim G_1$ und $H \sim G_2$ genauso schwierig wie $G_1 \sim G_2$
  - P hütet sich allerdings davor, auch noch die andere, von V nicht gewünschte, Alternative nachzuweisen – wieso?
- V kann den gelieferten Isomorphiebeweis einfach verifizieren



Unvorher-sehbar für P

P und V wiederholen alles n Mal, wobei von P jedes Mal ein anderer "Zeuge" H konstruiert wird
→ Beweissicherheit = $1\text{-}2^{-n}$

Zero-Knowledge bei Graphisomorphie – Eigenschaften

- Falls P lügt, also keinen Isomorphismus zwischen G_1 und G_2 kennt, kann P keinen Graphen H konstruieren, der nachweislich isomorph zu *beiden* ist
  - verschiedene H_1, H_2 zu finden mit $H_1 \sim G_1$ und $H_2 \sim G_2$ ist einfach; mit 50% Wahrscheinlichkeit wird P so allerdings der Lüge überführt!
- V lernt nichts über die Isomorphie $G_1 \sim G_2$
  - glaubt aber schliesslich, dass P eine solche kennt
- Zur Minimierung der Interaktionen lassen sich die "Runden" parallelisieren: P sendet mehrere "isomorphe Zeugen" an V, und V sendet einen Bitvektor zurück, der die Einzelnachweise auswählt

Zero-Knowledge bei Graphisomorphie – Eigenschaften

- V kann einem Dritten W gegenüber nicht beweisen, dass P den Isomorphismus kennt: Selbst ein exaktes Protokoll der Kommunikationsvorgänge muss W nicht überzeugen: P und V könnten sich verschworen haben!
- Da V nichts Relevantes gelernt hat, kann V sich anderen gegenüber auch nicht mit der Kenntnis schmücken
  - sich also nicht für P ausgeben (Kenntnis ist Identitätsausweis von P)
- Grosse Graphen sind in der Praxis etwas unhandlich; es gibt praktischere Ausprägungen des Zero-Knowledge-Verfahrens, z.B. das Protokoll von Fiat und Shamir; dieses beruht auf der Schwierigkeit, die k-te Wurzel in einem Restklassenring zu berechnen