Kryptographische Grundlagen

Algorithmen, Protokolle und Angriffe

Definition von Sicherheit

Unter “Sicherheit” soll hier die Garantie folgender Eigenschaften verstanden werden:

- **Vertraulichkeit**: Eine Nachricht soll nur für den/die Empfänger zugänglich sein.
- **Authentizität**: Es soll eindeutig feststellbar sein, ob eine empfangene Nachricht auch tatsächlich vom vorgeblichen Absender stammt.
- **Integrität**: Eine Veränderung der Nachricht auf dem Weg zwischen Sender und Empfänger soll erkannt werden können.
- **Non-Repudiation**: Der Sender einer Nachricht soll im Nachhinein nicht leugnen können, diese verfasst zu haben.

Grundwerkzeuge

- **Symmetrische Verschlüsselung**: Gleicher Schlüssel zum Ver- und Entschlüsseln.
- **Asymmetrische Verschlüsselung**: Verschlüsseln mit öffentlichem Schlüssel, Entschlüsseln mit privatem Schlüssel (NP-hartes Problem).
- **Digitale Signatur**: Verschlüsseln mit privatem, entschlüsseln mit dem öffentlichen Schlüssel; jeder kann verifizieren.
- **Hashfunktionen**: “Prüfsumme” einer Nachricht: Non-reversibel, kollisionsfrei, repräsentativ.

Rollen, Angriffstypen, Zertifizierung

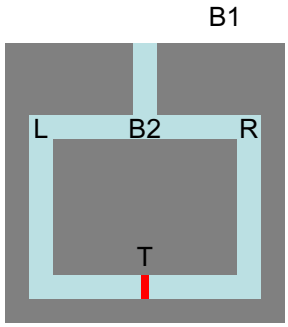
- **Rollen**:
  - **Alice**, **Bob**, **Carol** und **Dave** kommunizieren
  - **Eve** kann abhören
  - **Mallory** kann abhören und verfälschen
  - **Trent** ist ein “Notar”, dem alle vertrauen
- **Angriffstypen**:
  - Ciphertext-only
  - Known-Plaintext
  - Chosen-Plaintext
  - Chosen-Ciphertext
  - Man-in-the-middle
  - Playback
- **Zertifizierung**:
  - Notar bestätigt durch Unterschrift die Echtheit (z.B. eines Public Keys)

Bit-Commitment-Protokolle

- **Problem**: Alice muss sich zu einem bestimmten Zeitpunkt auf etwas festlegen (Commitment), ohne daß es Bob sofort erfährt.
  Danach soll Bob es erfahren, aber Alice soll ihre Aussage im Nachhinein nicht mehr ändern können.
- **Beispiel**: Alice ist Börsenmaklerin und Bob Spekulant.
  Sagt Alice ihm die Kurse im Voraus (als Beweis, das sie sich "auskennt") kauft Bob die Aktien selber und bezahlt Alice nicht.
  Sagt Alice Bob ihre Börsentips von letzter Woche, wird ihr Bob nicht glauben, daß sie sie schon damals wusste.
- **Lösung**: Bit-Commitment-Protokoll(e); beruhen allein auf für Hashfunktionen geforderten Eigenschaften.

Zero-Knowledge-Protokolle

- **Problem**: Alice muss Bob beweisen, das sie ein Geheimnis kennt, ohne daß Bob das Geheimnis selbst erfährt.
- **Beispiel**: Höhle (hellgrau) mit Tür T. Alice's Geheimnis ist es, diese Tür zu öffnen. Bob darf Öffnung nicht sehen.
- **Lösung**:
  - n mal: Bob befiehlt Alice, vom linken oder rechten Gang (L/R) zu ihm zu kommen (B2). Dazu muss Alice eventuell Tür öffnen.
  - Chance zu betrügen: 1/2n.
  - nur Interaktiv möglich.



Secret-Splitting-Protokolle

- **Problem**: Ein Geheimnis in mehrere Teile aufbrechen, die allein wertlos sind.
- **Beispiel**: Restaurantbesitzerin Alice kennt ein geheimes Rezept. Sie lebt in Sorge, dass einer ihre Köche dieses Rezept zur Konkurrenz trägt.
- **Lösung**: Alice zerlegt sie das Rezept und gibt jedem ein Fragment. Nur zusammen können die Köche das Rezept anwenden, jeder Teil allein ist völlig wertlos.
- **Beispiel**:
  - Secret S = 0xdeadbeef, Random R = 0x2463ae42. (hex)
  - Bob bekommt R.
  - Carol bekommt R xor S = 0xface10ad.
  - Carol und Bob zusammen:
    0x2463ae42 xor 0xface10ad = 0xdeadbeef = S

Blinde Signatur

- **Problem**: Jemand soll etwas unterschreiben, aber nicht erfahren können, was er unterschreibt.
- **Beispiel**: Trent ist Notar und signiert alles, was ihm vorgelegt wird; der Inhalt ist ihm dabei egal.
- **Lösung**: Ein **Blinding-Factor**:
- Sei D das zu unterschreibende Dokument, Blind() das Einfügen des Blinding Factors, und View() dessen Umkehrung. Sig() sei die Unterschrift.
- Da Sig() und View() **kommutativ** sein müssen gilt:
  View(Sig(Blind(D))) = Sig(View(Blind(D)))
  View(Blind(…)) heben sich gegenseitig auf, somit gilt:
  View(Sig(Blind(D))) = Sig(View(Blind(D))) = Sig(D)

Location Management mit Privatsphäre

MIX-Netzwerke und ihr Nutzen für die Mobilkommunikation

Motivation

Netzbetreiber ist fast “Big Brother”:

- Er weiß (technisch) **was** kommuniziert wird (GSM ist nicht Ende-zu-Ende verschlüsselt).
- Er weiß **wo** sich seine Kunden aufhalten (in Städten bis auf Häuserblocks genau).
- “stille SMS” etc: Spielplatz für Schnüffler

➔ **Idee**: Verschleierung des Standortes durch ein MIX-Netzwerk, eine Art “Remailing-Service”.

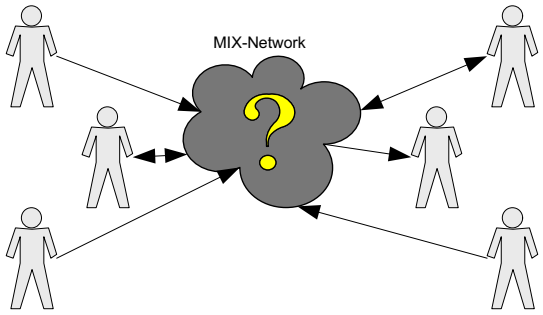
MIX-Netzwerke

Definition MIX-Netzwerk [Chaum 1981]:

Gerichteter Graph mit 2 Arten von Knoten: Benutzern und Mixen. Kanten beschreiben wer an wen etwas senden kann.

Aufgabe eines MIX-Netzwerks: Nachrichtenfluss kann nicht beobachtet werden, da Route (MIX-Kaskade) durch Mixe verschleiert wird.

- Mixe entsprechen Remailing-Postämtern: Brief auspacken, Inhalt (wieder ein Brief) weiterschicken



MIX-Netzwerke

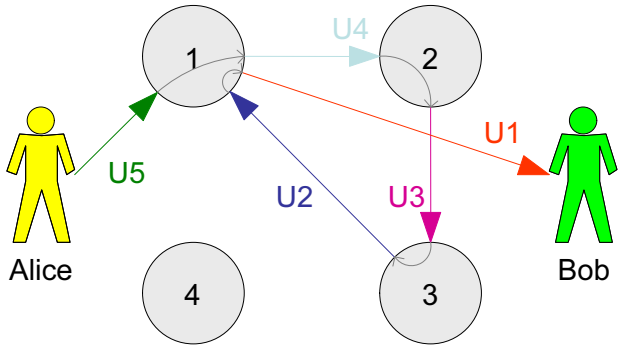
Anforderungen an die Mixe:

- **Padding/Splitting**: Nachrichten werden zu Blockgrössen aufgefüllt/zerlegt.
- **Dummy-Traffic**: Mixe müssen ständig Nachrichten oder Pseudo-Nachrichten senden.
- **Repetition-Ignoring**: Mixe müssen mehrmaliges Senden gleicher Nachrichten verweigern (Playback-Angriff)

MIX-Netzwerke

Beispiel für ein MIX-Netzwerk:

- Benutzer: Alice (A), Bob (B) sowie diverse andere.
- Mixe: M1, M2, M3, M4.
- Alice schreibt Nachricht M an Bob. Sie wählt als Mix-Kaskade:
  A → M1 → M2 → M3 → M1 → B
- $U_5 = M1,Enc_{pM1}(M2,Enc_{pM2}(M3,Enc_{pM3}(M1,Enc_{pM1}(B,Enc_{pB}(M)))))$

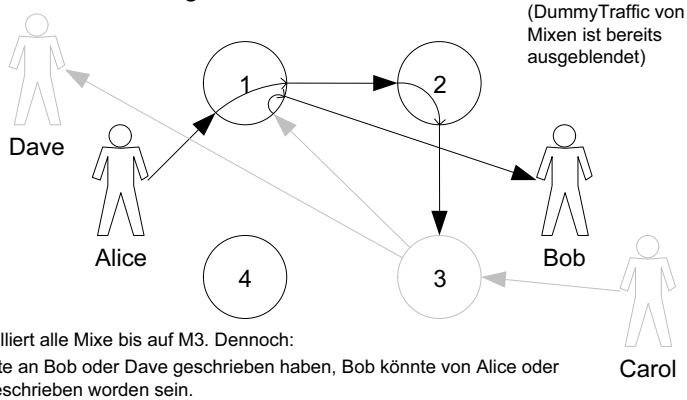


MIX-Netzwerke

Angriffe gegen MIX-Netzwerke:

MIX-Netwerk ist sicher, solange *mindestens ein* Mix der Kaskade sicher ist. Durch dessen Dummy-Traffic ist keine In/Out-Zuordnung der Nachrichten möglich.

(DummyTraffic von Eve's Mixen ist bereits ausgeblendet)



Eve kontrolliert alle Mixe bis auf M3. Dennoch:
Alice könnte an Bob oder Dave geschrieben haben, Bob könnte von Alice oder Carol angeschrieben worden sein.

Location Management mit Mixen

MIX-Netzwerke lassen sich nicht nur einsetzen, um die *Identität* des Senders geheim zu halten, sondern auch dessen *Standort*.

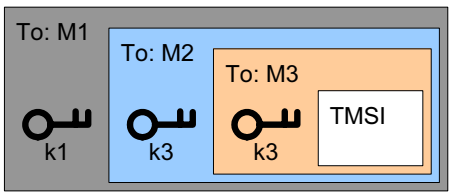
Anwendungsbeispiel für die Mobilkommunikation:

- Der Teilnehmer gibt seinen Standort (Location Area Identifier, LAI) nur in verschleierter Form, einer "untraceable Return Address" {LAI} gegenüber dem Home Location Register HLR des Netzbetreibers preis.
- Die Mix-Kaskade liegt zwischen dem HLR und der Location Area des Teilnehmers. An diesem Ende wird der Teilnehmer über seine Temporary Mobile Subscriber Identity TMSI ("vergängliches Pseudonym") referenziert.

Location Management mit Mixen

Erzeugung einer untraceable Return Address {LAI}:

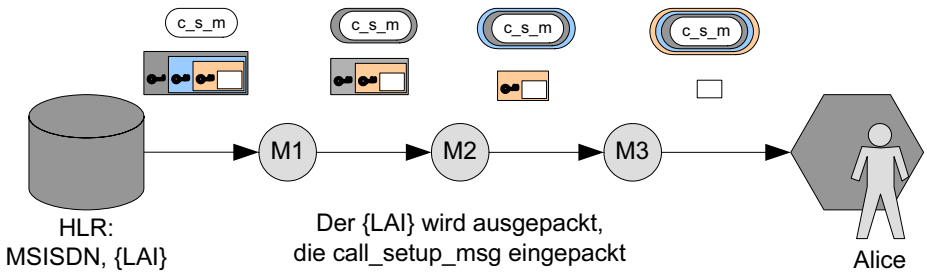
- Beispiel: Alice legt sich auf die Mix-Kaskade HLR → M1 → M2 → M3 → TMSI fest:
- An das HLR übergibt sie einen Umschlag mit Adresse von M1 und Schlüssel k1 für symmetrische Verschlüsselung.
- Im innersten Umschlag sind die TMSI sowie ein Schlüssel k3.
  - $U_1 = Enc_{PubM3}(k_3, TMSI)$
  - $U_2 = Enc_{PubM2}(k_2, M3, U_1)$
  - $U_3 = Enc_{PubM1}(k_1, M2, U_2)$
    $= Enc_{PubM1}(k_1, M2, Enc_{PubM2}(k_2, M3, Enc_{PubM3}(k_3, TMSI)))$
  - {LAI} = (M1, U_3)



Location Management mit Mixen

Mobile-originated-Call-Setup:

- Der {LAI} und die c_s_m auf dem Weg vom Netwerk zum Teilnehmer: Der {LAI} wird auf der Route ausgepackt und die c_s_m eingepackt.
- Bei Alice angekommen muss die verpackte Nachricht
  $Enc_{k3}(Enc_{k2}(Enc_{k1}(c\_s\_m)))$
  in umgekehrter Reihenfolge entschlüsselt werden.



Location Management mit Mixen

Mobile-originated-Call-Setup:

- Deutlich einfacher; erfolgt wie ein Nachrichtenversand im allgemeinen MIX-Netzwerk.

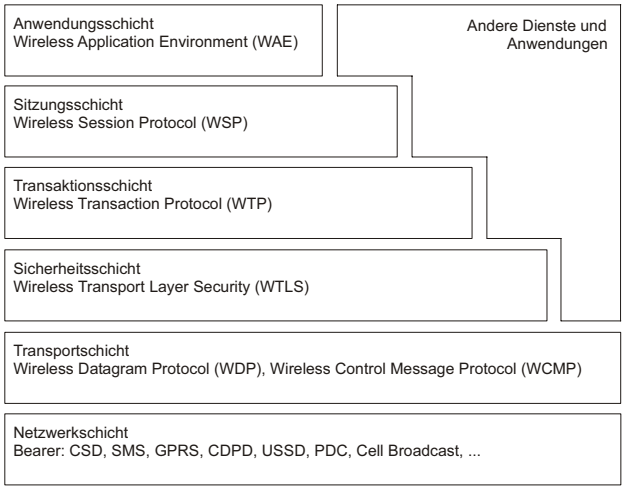
Fazit:

- Grundidee gut, aber…
- hoher Rechenaufwand:
  - Jeder {LAI} benötigt mehrfache public-Key-Verschlüsselung,
  - ist dabei aber nur einmal verwendbar (MIXE: Repetition-ignoring)!
- Unklar ist, wer das MIX-Netzwerk betreibt (der Netzbetreiber?)

Sicherheitsmechanismen im Wireless Application Protocol

Wireless Transport Layer Security

WTLS: Einbindung in den Protokollstack



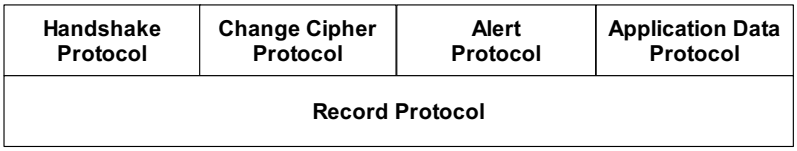
Wireless Transport Layer Security (WTLS)

- Orientierung an der im Internet üblichen Sicherheitsschicht **Transport Layer Security** (TLS bzw. SSL) mit zusätzlicher Optimierung für drahtlose Netze durch Berücksichtigung von:
  - langen round-trip-delay-Zeiten
  - niedrigen Bandbreiten
  - potentiell hohen Fehlerraten
  - eingeschränkter Rechenleistung und Speicherkapazität der Endgeräte
  - Im- und Exportbeschränkungen für Kryptographieverfahren
- **geforderte Eigenschaften**:
  - Datenintegrität
  - Vertraulichkeit
  - (gegenseitige) Authentifikation
  - Schutz gegen DoS-Angriffe
- selektive Nutzung der Sicherheitsfunktionen (abhängig von Anforderungen der Anwendung und Einschränkungen des Netzes)

Besonderheiten der WAP-Schichten

- **Transportschicht**
  - einheitliches Interface zur Nutzung verschiedener Trägertechnologien
  - verbindungsloser Datentransport im *Wireless Datagram Protocol*
- **Sicherheitsschicht**
  - Verwendung optional
  - erlaubt (im Ggs. zu TLS) Datagramme
- **Transaktionsschicht**
  - bietet u.a. verbindungsorientierten Datentransfer auf Grundlage von WDP
  - entspricht (teilweise) TCP
- **Sitzungsschicht**
  - WAP-Äquivalent zu HTTP 1.1
  - Session-Management sowohl für verbindungs- als auch für datagrammorientierten Betrieb
- **Anwendungsschicht**
  - Anwendungen und Dienste, z.B. Wireless Markup Language (WML)

WTLS: Architektur



- **Record Protocol**
  - Basis für Handshake-, Change Cipher- und Application Data Protocol
  - Verwaltung sämtlicher Informationen über gegenwärtigen und zukünftigen Zustand der Verbindung
  - Durchführung der eigentlichen kryptographischen Operationen
- **Handshake Protocol**
  - Vereinbarung der zu verwendenden Algorithmen
  - Austausch der dafür nötigen Parameter, Schlüssel und ggf. Zertifikate
- **Change Cipher Protocol**: Einleitung des verschlüsselten Datenaustauschs
- **Alert Protocol**: Meldung sicherheitskritischer Fehler an den Kommunikationspartner
- **Application Data Protocol**: Protokoll der übergeordneten Anwendungsschicht

WTLS: Grundbegriffe, unterstützte Algorithmen

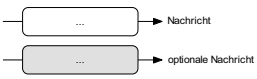
- **Sitzung (Session)**
  - zwischen den Verbindungsendpunkten wird *eine* sichere *Session* aufgebaut
- **Verbindung (Connection)**
  - eine Session dient als Umgebung für eine beliebige Anzahl von *Verbindungen*
- **unterstützte Algorithmen/Verfahren**:
  - *Bulk Encryption Algorithms* (symmetr. Verschlüsselung): RC5, DES, IDEA, 3DES
  - *Key Exchange Suites* (asymmetr. Verschlüsselung): DH, RSA, ECDH
  - *Keyed MAC Algorithms* (Hashfunktionen): SHA, MD5

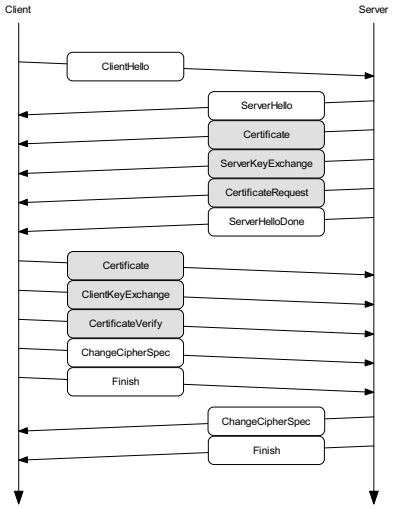
WTLS: Grundlegendes Prinzip

1. Vereinbarung der zu verwendenden Algorithmen, Austausch der nötigen Parameter
2. Bestimmung der Schlüssel für die Nutzdatenverschlüsselung mit Hilfe des vereinbarten (asymmetrischen) Schlüsselaustauschverfahrens
3. Austausch von (symmetrisch verschlüsselten) Nutzdaten
4. Im Fehlerfall oder ggf. nach Austausch einer bestimmten Datenmenge (d.h. Anzahl von Paketen): Neustart bei 1.

WTLS: Handshake Protocol (1)

- im Vergleich zu TLS möglichst „kompakt“ formuliert
- möglichst wenige Acknowledge-Meldungen u.ä., um auch in „langsamen“ mobilen Netzen (hoher round-trip-delay) schnellen Verbindungsaufbau zu erlauben
- „teure“ asymmetrische Verschlüsselung nur falls unbedingt notwendig
- optional verkürzte Verfahren (s.u.)





WTLS: Handshake Protocol (2)

1. **ClientHello**
   - nach Präferenz sortierte Liste der vom Client unterstützten Algorithmen
   - Zufallszahl (*Client Random*)
   - aktuelle Sitzungskennung falls bereits aktive Sitzung besteht (sonst 0)
   - weitere Daten
2. **ServerHello**
   - Auswahl der am besten geeigneten Verfahren aus der Liste des Clients
   - Zufallszahl (*Server Random*)
   - Sitzungskennung
   - weitere Daten
3. **Certificate, CertificateRequest, Server-/ClientKeyExchange**
   - ggf. Austausch von Zertifikaten und weiteren Schlüsselinformationen
   - *Optimized Full Handshake*: Server bezieht Zertifikat des Clients von einem Verzeichnisdienst oder aus einer eigenen Liste

WTLS: Handshake Protocol (3)

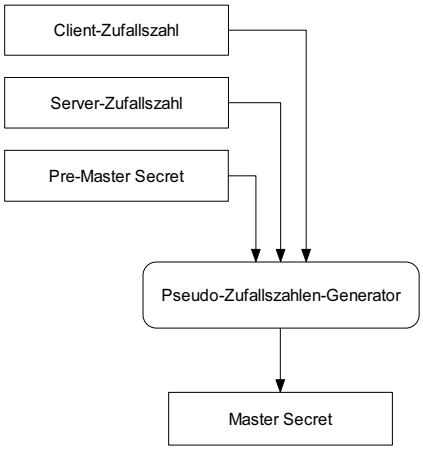
4. **Berechnung des Master Secret**
   - nach Austausch der Schlüsseldaten, Zufallszahlen und Zertifikate auf beiden Seiten alle nötigen Daten bekannt, um gemeinsames Geheimnis zu berechnen
   - konkretes Verfahren: s. u.
5. **ChangeCipherSpec**
   - erklärt die getroffenen Vereinbarungen für gültig
   - letzte Klartext-Nachricht; alle folgenden werden durch den vereinbarten Bulk Encryption Algorithm verschlüsselt
6. **Finish**
   - Beendigung des Handshake-Vorgangs
   - enthält Hashwert über alle bisherigen Nachrichten und das gemeinsame Geheimnis zur gegenseitigen Authentifizierung

WTLS: Berechnung des Master Secret

- **Pre-Master Secret** über Schlüsselaustauschverfahren vereinbart
- **Client- und Server-Zufallszahl** jeweils aus den entsprechenden Hello-Nachrichten bekannt
- **Pseudo-Zufallszahlen-Generator** erzeugt bei gleicher Eingabe immer dasselbe Resultat

➔ ***auf beiden Seiten unabhängig voneinander gleiches Ergebnis***

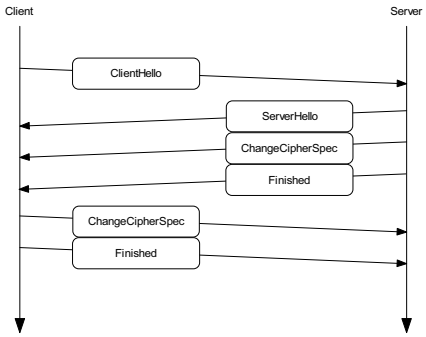
- **Shared Secret Handshake**: beidseitig fest implementiertes Shared Secret (z.B. in Hardware) dient als Pre-Master Secret



WTLS: Abbreviated Handshake

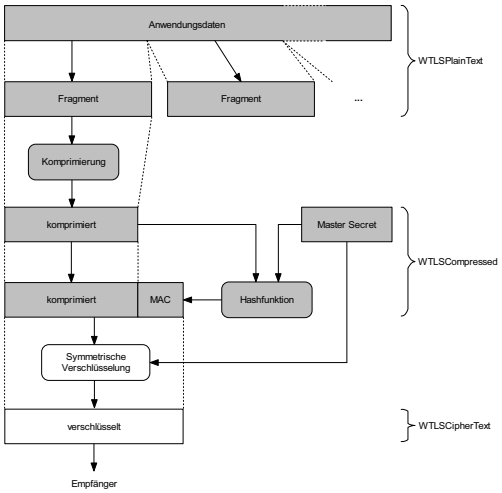
- **verkürzter Handshake**:
  - beim Aufbau von Verbindungen in bereits bestehender Sitzung
  - um eine unterbrochene Session wieder aufzunehmen
  - um im Verlauf einer Sitzung neue Schlüssel zu erzeugen
- Client- und ServerHello enthalten die aktive Session ID
- bisheriges Master Secret wird bei der Berechnung des neuen gemeinsamen Geheimnisses als Pre-Master Secret verwendet

➔ ***Informationsaustausch mit asymmetrischer Verschlüsselung entfällt***



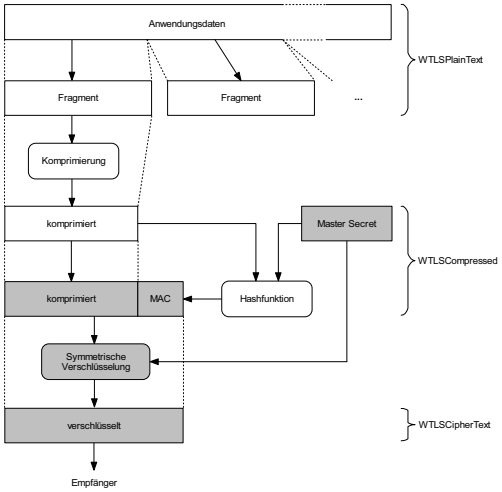
WTLS: Verarbeitung von Anwendungsdaten (1)

1. Aufteilung der Daten in Fragmente
2. Kompression der Fragmente (derzeit nur *Null-Komprimierung* spezifiziert)
3. Anhängen eines Nachrichtenauthentifizierungscodes (MAC), mittels der vereinbarten Hashfunktion aus dem komprimierten Fragment und dem Master Secret berechnet



WTLS: Verarbeitung von Anwendungsdaten (2)

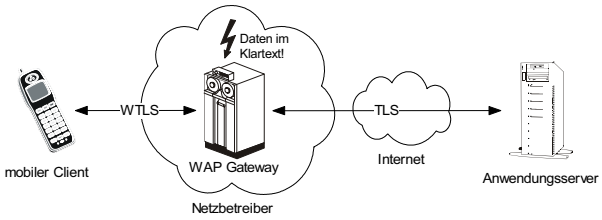
4. Verschlüsselung des so entstandenen Datenpakets nach dem vereinbarten symmetrischen Algorithmus mit dem gemeinsamen Geheimnis als Schlüssel
5. Versand des verschlüsselten Datenblocks an den Empfänger



WTLS: Probleme und Schwachstellen

- TLS als Grundlage für WTLS, dadurch Vermeidung vieler potentieller Probleme und bereits bekannter Sicherheitslücken im voraus
- einige Abweichungen von TLS Ursache für neue Risiken, insbesondere:
  - Unterstützung von Datagrammen
  - Unterschiede im Protokoll erfordern für die Kommunikation zwischen WTLS- und TLS-kompatiblen Geräten (typischer Fall: Mobiltelefon und Internet-Server) eine „Übersetzung“

WTLS: WAP Gap (1)



- Absicherung der Kommunikation zwischen Client und Gateway mit WTLS, zwischen Gateway und Server mit TLS
- Gateway „übersetzt“ Daten zwischen WTLS und TLS und enthält diese dabei zumindest zeitweise im Klartext

➔ ***keine end-to-end-Sicherheit.***

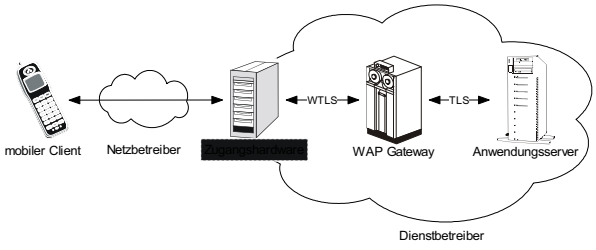
WTLS: WAP Gap (2)

Durch die Gateway-Architektur bedingte Probleme:

- Zugriff Dritter auf das Gateway-System unvermeidbar (zumindest durch Administrationspersonal)
- ein einzelner Knotenpunkt, über den viele sicherheitskritische Transaktionen abgewickelt werden, stellt ein ideales Angriffsziel dar
- potentielle Performance-Schwachstelle (*bottleneck*)
- keine end-to-end-Authentifizierung zwischen Anbieter und Benutzer möglich

WTLS: WAP Gap (3)

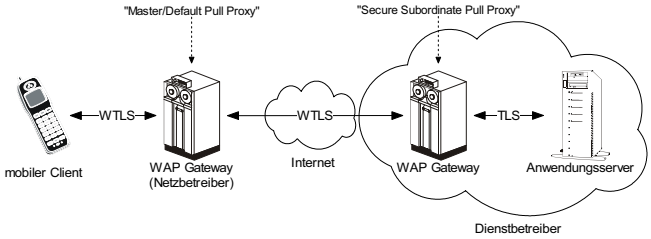
1. Verbesserungsansatz: dedizierter Gateway beim Dienstanbieter



- **Vorteil**:
  - Endpunkt der WTLS-geschützten Verbindung im direkten Einflußbereich des Anbieters; damit (im weiteren Sinne) end-to-end-Sicherheit
- **Nachteile**:
  - aufwendige Infrastruktur – jeder Anbieter muß Zugangspunkte für ggf. mehrere mobile Netze bereitstellen
  - Anwender muß verschiedene Account-Daten für sämtliche Dienste verwalten

WTLS: WAP Gap (4)

2. Verbesserungsansatz: *Transport Layer end-to-end-Security*



- **Vorteile gegenüber 1. Ansatz**:
  - keine Zugangshardware beim Dienstbetreiber notwendig
  - keine Dienst-spezifischen Einstellungen beim Benutzer
- **Nachteil**:
  - Kooperation mit Netzbetreiber nötig

WTLS: Schwache Algorithmen

- **DES-Verschlüsselung mit 40 Bit Schlüssellänge**
  - effektive Länge 35 Bit, da bei DES-Schlüsseln ein Paritätsbit pro Byte
  - kein ausreichender Schutz von Brute-Force-Angriffen

  ➔ in aktueller WTLS-Revision nur noch aus Kompatibilitätsgründen enthalten, von der Verwendung wird ausdrücklich abgeraten

- **Nachrichtenauthentifizierung mit SHA_XOR_40**
  - Erzeugung eines MAC durch Aufteilung der Nachricht in 5-Byte-Blöcke und anschließende XOR-Verknüpfung
  - Nachricht kann leicht verfälscht werden: Wird ein beliebiges Bit *n* gekippt, muß lediglich das ensprechende Bit (*n* mod 40) des MAC invertiert werden

  ➔ offensichtlich kein sinnvoller MAC, daher in der dritten WTLS-Revision aus der Liste der MAC-Algorithmen entfernt

WTLS: Nicht authentifizierte Alert-Nachrichten

- in der ersten WTLS-Version einige Alert-Nachrichten (meist *Warnings*) im Klartext und ohne Authentifizierung
- bei verbindungslosem Transport benötigen auch solche Meldungen eine Sequenznummer
- Angreifer kann beliebige verschlüsselte Pakete aus dem Datenstrom „entfernen“, indem er Fehlermeldungen mit den entsprechenden Sequenznummern verschickt (*Datagram Truncation Attack*)
- in aktueller WTLS-Revision: *alle* Alert-Nachrichten enthalten zur Authentifizierung eine 4 Byte lange Prüfsumme aus dem zuletzt empfangenen verschlüsselten Datenblock

WTLS: Alternativen

- **WAE-Sec**
  - vollständig TLS-kompatible Sicherheitsschicht innerhalb des *Wireless Application Environment* (Anwendungsschicht)
  - Kommunikation direkt mit der Transportschicht unter Umgehung von WSP und WTP
  - durch Kompatibilität zu TLS Vermeidung des WAP Gap-Problems
  - keine Angaben zur Performance
- **KSSL (Kilobyte SSL)**
  - vollständige Implementierung von TLS für mobile Endgeräte in Java (J2ME)
  - Performance-Untersuchung: vollständiger TLS-Handshake in einem Zeitrahmen von etwa 10-13 Sekunden, damit für sicherheitskritische Anwendungen (z.B. Online-Banking) vertretbar
- **LEAP (Lightweight and Efficient Application Protocol)**
  - komplettes Schichtenmodell auf Basis freier Protokolle

Ausblick: WAP 2.0

- Berücksichtigung der schnellen Entwicklung der Endgeräte (Rechen- und Speicherkapazität) und Mobilfunknetze (höhere Übertragungsraten)
- Orientierung hin zu etablierten Internet-Standards, insbesondere IP, TCP, HTTP und XHTML (Gateway wird überflüssig)
- Erhaltung des alten WAP-Stacks aus Kompatibilitätsgründen und für nicht IP-fähige Trägerdienste

Digitales Geld

Anforderungen und Implementierungen

Digitales Geld: Überblick

Motivation:

- Digitales Geld soll **Bargeld in digitaler Form** gut annähern.
- "**Technische Anonymität**" soll garantiert sein.

Anforderungen:

- **Double-spending-Sicherheit**
- **Privatsphäre**: Geldfluss nicht nachvollzieh- / überwachbar.
- **Unabhängigkeit**: Sicherheit hängt nicht vom physischen Ort ab, über Computernetze übertragbar.
- **Transferierbarkeit**: Es ist zu anderen Benutzern übertragbar.
- **Offline-Bezahlung**: Bezahlung benötigt keine Online-Verbindung zur Bank.
- **Teilbarkeit**: Es kann (mit korrekter Summe) zerlegt werden.

Anonyme Schecks: Überblick

Anonyme Schecks garantieren:

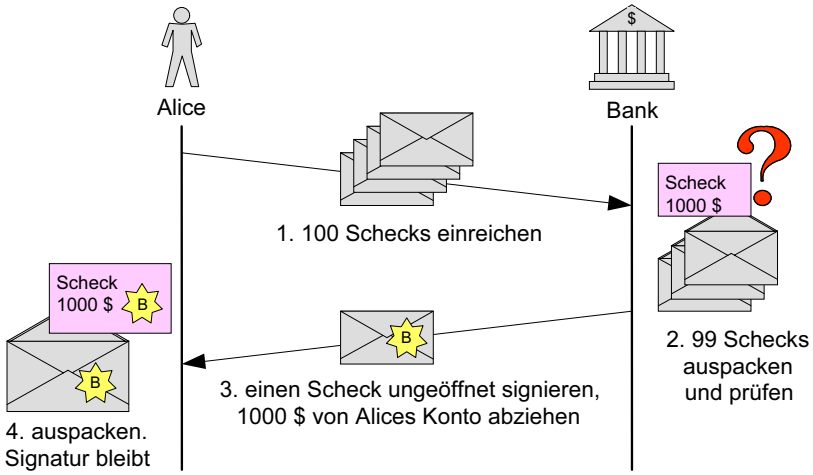
- Privatsphäre
- Double-Spending-Sicherheit
- Unabhängigkeit
- Offline-Bezahlung

Die Rolle der **Bank**:

- **Ausstellen** von Schecks: (+ Abziehen vom Konto)
- **Annehmen** von Schecks: (+ Gutschrift auf Konto)
- Betrüger *identifizieren* (*nur* bei Betrug)
- erfährt nicht, wo ein ausgestellter Scheck **hinkommt**,
- erfährt nicht, wo ein eingereichter Scheck (ursprünglich) **herkommt**.

Anon. Schecks: Protokoll 1 (1)

Erzeugung eines anonymen, von der Bank signierten Schecks:



Anon. Schecks: Protokoll 1 (2)

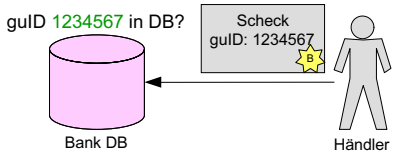
Diskussion

- Anonymität:
  - Bank weiss nicht, wohin Scheck für Alice geht,
  - Bank weiss nicht, woher Scheck des Händlers kommt.
- *Keine* Double-Spending-Sicherheit:
  - Alice kann denselben Scheck an mehrere Händler ausgeben und/oder
  - Händler kann denselben Scheck mehrfach zur Bank tragen.
  - Bank kann dies nicht erkennen, kann “ausgesaugt” werden.

Fazit: Protokoll funktioniert (noch) nicht!

Anon. Schecks: Protokoll 2 (1)

- **Idee**: Jeden Scheck vor Einpacken in Umschlag mit einer eigenen zeitlich/weltweit einmaligen **Eindeutigkeitsfolge** (guID) versehen.
- **Bezahlen** beim Händler:
  - Händler löst Scheck noch *während* des Bezahlens bei Bank ein.
- **Einlösen** des Schecks:
  - Bank überprüft ob *dieselbe* guID schon einmal eingereicht wurde.
  - falls nein: guID auf die Liste für verbrauchte setzen.
  - falls ja: Illegale Scheckkopie, Annahme verweigern!



Anon. Schecks: Protokoll 2 (2)

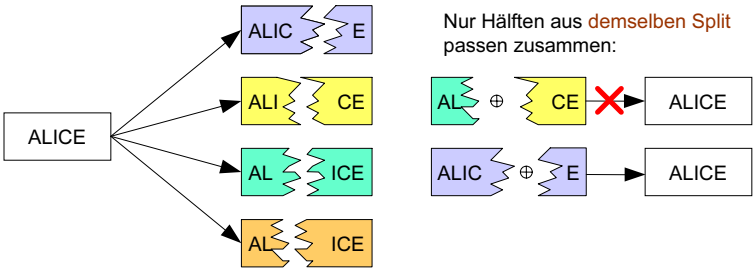
Diskussion:

- Double-Spending-Sicherheit:
  Falls Händler den Scheck erst später einlöst, kann Bank zwar Betrug erkennen, weiß aber nicht,
  - Ob Händler betrügt, indem er zum 2. Mal einlöst
  - Ob Alice den Händler betrogen hat.

Fazit: Protokoll noch “unausgewogen”, da unvorteilhaft für Händler, Alice kann zu leicht entwischen.

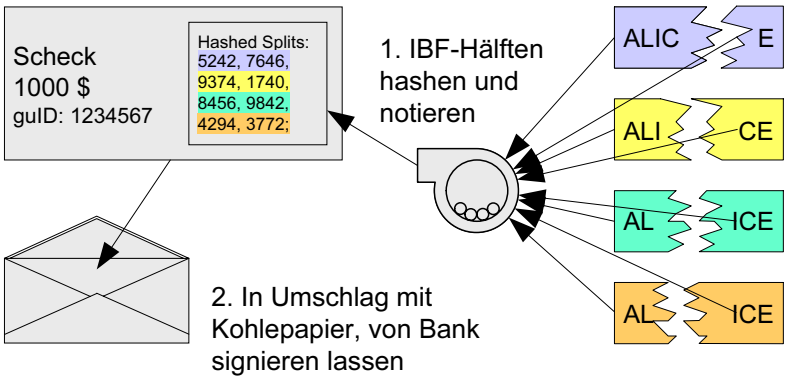
Anon. Schecks: Protokoll 3 (1)

- **Idee**: Alice muss in jedem Scheck “Bruchstücke” ihres Namens notieren (**IdentitätsBitFolge** IBF).
- **Erzeugung** des Schecks:
  - Für jeden Scheck zerlegt Alice ihre IBF n mal (Secret Splitting) in linke/rechte Hälfte I_{xL}, I_{xR}.



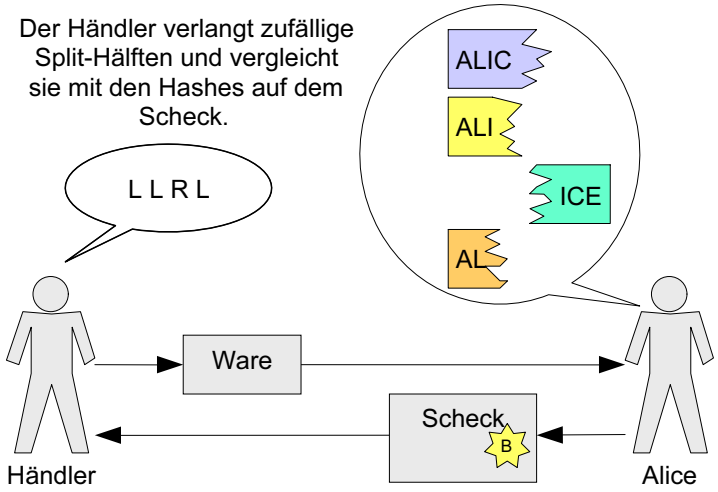
Anon. Schecks: Protokoll 3 (2)

- Hashes aller IBF-Splithälften auf Scheck notieren.
- Jeder Scheck kommt in einen Umschlag mit Kohlepapier.
- Bank verfiziert für n-1 Schecks alle IBF-Splithälften + deren Hashes.



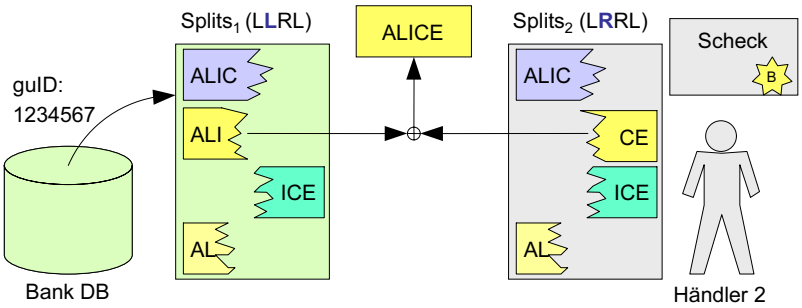
Anon. Schecks: Protokoll 3 (3)

- **Bezahlen** beim Händler:



Anon. Schecks: Protokoll 3 (4)

- "**Alice betrügt**": Dann existieren 2 unterschiedliche Auswahlstrings



- "**Händler betrügt**":
  - Er müsste Bank IBF-Splithälften zu anderen Auswahlstrings geben.
  - Er kann auch keine IBF-Splithälften "erfinden", da Hashes auf Scheck notiert.

Anon. Schecks: Protokoll 3 (5)

Diskussion:

- Die “Schwachstellen” (1/n Chance für Alice bei Umschlägen) müssen durch Strafen für Betrüger “unattraktiv” werden.
- Nicht zwischen Benutzern transferierbar:
  Falls Alice ihren Scheck an Mallory verkauft und dieser betrügt, wird Alice als Betrügerin identifiziert!

Anonyme Schecks

- **Gesamtfazit**: “Praxistauglich”
  - Scheckerzeugung ist zwar rechenintensiv, kann aber im Voraus erledigt werden.
  - Schecks können auf Alice’s Handy geladen und an Händler ohne Bankverbindung weitergegeben werden.
  - Händler muss nur die Unterschrift der Bank und Hashwerte sofort berechnen können (vertretbarer Aufwand).

- **Implementierungen**:
  - NetCash: Prototyp der University of Southern California
  - NetCheque: Ergänzung mit skalierbarem Anonymitätsgrad
  - PayMe: bietet zusätzlich Schecktransfer zwischen Benutzern
  - eCash, DigiCash: mittlerweile wieder eingestellt