Vorlesung Sicherheit

Dennis Hofheinz

ITI, KIT

12.06.2017

Überblick

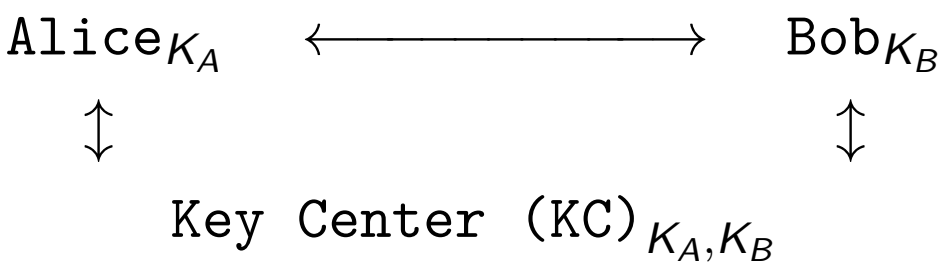
1 Schlüsselaustauschprotokolle
- Symmetrische Verfahren
- Asymmetrische Verfahren
- Transport Layer Security (TLS)

Überblick

1 Schlüsselaustauschprotokolle
- Symmetrische Verfahren
- Asymmetrische Verfahren
- Transport Layer Security (TLS)

Erinnerung

- Ziel: gemeinsamen geheimen Schlüssel K aushandeln
- Szenario: Secret-Key-Infrastruktur (mit Schlüsselzentrale):



- Vorschlag:
  1. $\texttt{Alice} \xrightarrow{(\texttt{Alice},\texttt{Bob})} \texttt{KC}$
  2. $\texttt{KC} \xrightarrow{\mathrm{Enc}(K_A,K),\quad \mathrm{Enc}(K_B,K)} \texttt{Alice}$
  3. $\texttt{Alice} \xrightarrow{\mathrm{Enc}(K_B,K)} \texttt{Bob}$

Kerberos

- Sehr gängig: Kerberos (im folgenden ursprüngliches Protokoll)

1. $\texttt{Alice} \xrightarrow{(\texttt{Alice},\texttt{Bob})} \texttt{KC}$
2. $\texttt{KC} \xrightarrow{\mathrm{Enc}(K_A,(T_{\texttt{KC}},L,K,\texttt{Bob})),\quad \mathrm{Enc}(K_B,(T_{\texttt{KC}},L,K,\texttt{Alice}))} \texttt{Alice}$

   Hier $T_{\texttt{KC}}$ Zeitstempel, L Gültigkeitsdauer, K frischer Schlüssel
3. $\texttt{Alice} \xrightarrow{\mathrm{Enc}(K,(\texttt{Alice},T_A)),\quad \mathrm{Enc}(K_B,(T_{\texttt{KC}},L,K,\texttt{Alice}))} \texttt{Bob}$

   Hier T_A Zeitstempel
4. $\texttt{Bob} \xrightarrow{\mathrm{Enc}(K,T_A+1)} \texttt{Alice}$

Eigenschaften von Kerberos

- Geschachtelte Struktur verhindert Man-in-the-Middle-Angriffe
- Zeitstempel/Gültigkeitsdauern verhindern Replay-Angriffe
- **Aber:** aktiv sichere Verschlüsselung nötig!
- Sicherheit von Kerberos nicht formal geklärt

Eigenschaften von Kerberos

- **Wichtig:** Kerberos authentifiziert Alice und Bob auch
  - Reales Kerberos hauptsächlich zur Authentifizierung genutzt
  - Kerberos-Standard/-Implementierung historisch zur Single-Sign-On-Lösung gewachsen

Überblick

1 Schlüsselaustauschprotokolle
- Symmetrische Verfahren
- Asymmetrische Verfahren
- Transport Layer Security (TLS)

Asymmetrische Verfahren

- **Szenario:** Public-Key-Infrastruktur (pk_A, pk_B öffentlich)

$$\texttt{Alice}_{sk_A} \quad \longleftrightarrow \quad \texttt{Bob}_{sk_B}$$

- Schlüsselgenerierung von Parteien selbst vorgenommen
- Schlüsselverteilung/-zertifizierung als gegeben angenommen
- **Frage:** Wie könnte man hier Schlüssel austauschen?

Public-Key Transport

- **Szenario:** Public-Key-Infrastruktur (pk_A, pk_B öffentlich)

$$\texttt{Alice}_{sk_A} \quad \longleftrightarrow \quad \texttt{Bob}_{sk_B}$$

- Werkzeug: Public-Key-Verschlüsselung $(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$
- Schlüsselpaare mittels Gen generiert
- Basisverfahren überaus simpel:
  1. $\texttt{Alice} \xrightarrow{\mathrm{Enc}(pk_B, K)} \texttt{Bob}$ (mit K von Alice frisch gewählt)

Eigenschaften von Public-Key Transport

$$\texttt{Alice}_{sk_A} \xrightarrow{C:=\mathrm{Enc}(pk_B,K)} \texttt{Bob}_{sk_B}$$

- Erzielt so nur passive Sicherheit
  - Replay-Angriff möglich (wiederhole C)
    - (Ist das überhaupt ein Angriff?)
  - Nicht authentifiziert (wenn Kanal nicht authentifiziert)
- Allerdings passive Sicherheit erreicht
  - Passive Sicherheit $:\Leftrightarrow$ nur lauschender $\mathcal{A}$ (hier: der C erhält) kann K nicht von Zufall unterscheiden
    (genaues Modell lässt $\mathcal{A}$ mehrere Sitzungen beobachten)
  - PKE-Verfahren IND-CPA $\Leftrightarrow$ Schlüsselaustausch passiv sicher

Eigenschaften von Public-Key Transport

$$\texttt{Alice}_{sk_A} \xrightarrow{C := \mathrm{Enc}(pk_B, K)} \texttt{Bob}_{sk_B}$$

- Kann durch zusätzliche Signatur authentifiziert werden:

$$\texttt{Alice}_{\substack{sk_{\texttt{PKE},A},\\ sk_{\texttt{Sig},A}}} \xrightarrow{(C := \mathrm{Enc}(pk_{\texttt{PKE},B}, K),\ \sigma = \mathrm{Sig}(sk_{\texttt{Sig},A}, C))} \texttt{Bob}_{\substack{sk_{\texttt{PKE},B},\\ sk_{\texttt{Sig},B}}}$$

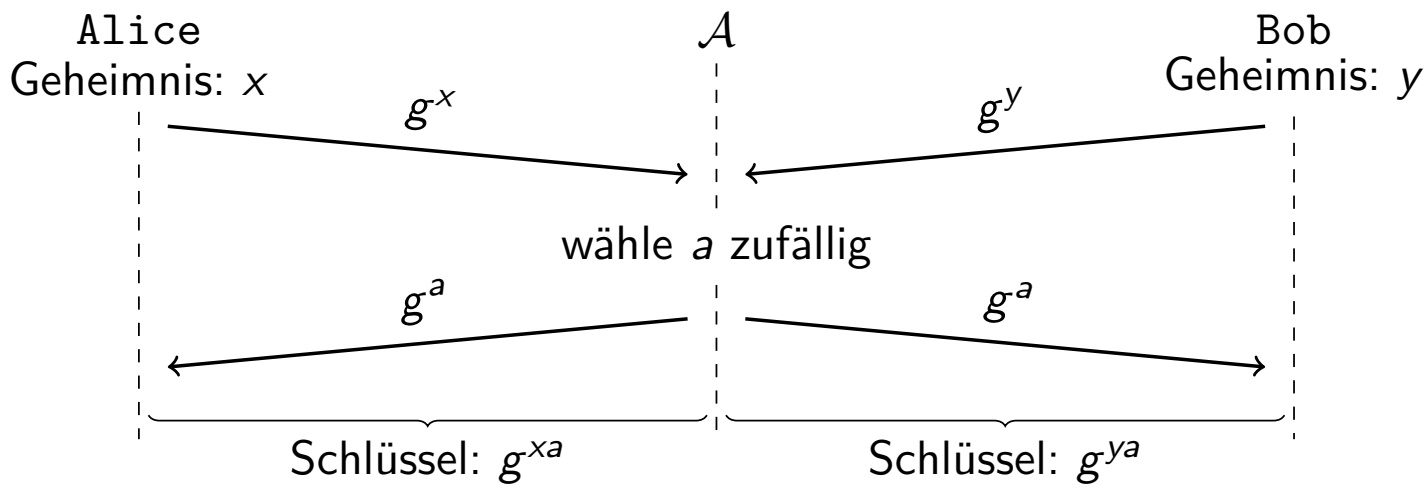
- Allerdings immer noch anfällig z.B. für Replay-Angriffe

Diffie-Hellman-Schlüsselaustausch

- Basisschema sehr ähnlich zu ElGamal ($\mathbb{G} = \langle g \rangle$ gegeben)
  1. $\texttt{Alice} \xrightarrow{g^x} \texttt{Bob}$
  2. $\texttt{Bob} \xrightarrow{g^y} \texttt{Alice}$
  3. Gemeinsamer Schlüssel: $K = g^{xy} = (g^y)^x = (g^x)^y$
- Diffie-Hellman (1976) lange vor ElGamal (1985), erstaunlich
- Passiv sicher unter geeigneter zahlentheoretischer Annahme
- Wird durch Signieren aller Nachrichten authentifiziert
  - Allerdings immer noch anfällig z.B. für Replay-Angriffe

Diffie-Hellman-Schlüsstelaustausch: Man-in-the-Middle (1)

- DH-Schlüsselaustausch anfällig für Man-in-the-Middle-Angriffe
- Angreifer $\mathcal{A}$ fängt Nachrichten von Alice & Bob ab



Diffie-Hellman-Schlüsselaustausch: Man-in-the-Middle (2)

- Alice denkt, der Schlüssel ist g^{xa}
- Bob denkt, der Schlüssel ist g^{ya}
- $\mathcal{A}$ kennt sowohl g^{xa}, als auch g^{ya}
- $\mathcal{A}$ kann Kommunikation zwischen Alice & Bob abhören, ohne aufzufallen

Diffie-Hellman-Schlüsselaustausch: Man-in-the-Middle (3)

Beispiel: Verschlüsselung

- Schlüssel wird für symm. Verschlüsselung verwendet
- Alice will $C = \mathrm{Enc}(g^{xa}, M)$ an Bob schicken
- $\mathcal{A}$ fängt C ab, entschlüsselt $M = \mathrm{Dec}(g^{xa}, C)$, sendet $C' = \mathrm{Enc}(g^{ya}, M)$ an Bob
- Bob schickt Chiffrat an Alice: Analog
- $\mathcal{A}$ kann mitlesen, fällt aber nicht auf

Lösung der Problematik: Signaturen

Geeignete formale Modelle

- Geeignetes Modell für Schlüsselaustausch nichttrivial
  - Komplexes Szenario: viele Parteien, viele Sitzungen pro Partei
  - Angreifer passiv oder aktiv?
  - Was „darf“ ein aktiver Angreifer genau?
  - In der Realität subtile Angriffe (?) möglich (später mehr)
  - Reales Netzwerk (TCP/IP) spezifisch und komplex
    - Wie können einzelne Sitzungen unterschieden werden (SIDs)?
  - Zahlreiche verschiedene theoretische Modelle
- Deshalb hier kein abstraktes Sicherheitsmodell

Überblick

1 Schlüsselaustauschprotokolle
- Symmetrische Verfahren
- Asymmetrische Verfahren
- Transport Layer Security (TLS)

Motivation für TLS

- **Problem:** `http`, `imap`, `smtp`, `ftp`, `...` ungeschützt
  - Interneteinkauf, Mailabholung/-transport, Dateitransfer
- **Mögliche Lösung:** auf höherer Protokollebene sichern
  - E-Mail verschlüsseln/signieren
  - Aufpassen: *alles* verschlüsseln/signieren!

    (Nur Kreditkartendaten verschlüsseln könnte Manipulation erlauben)
- **Besser:** Universelle Lösung auf Transportebene*

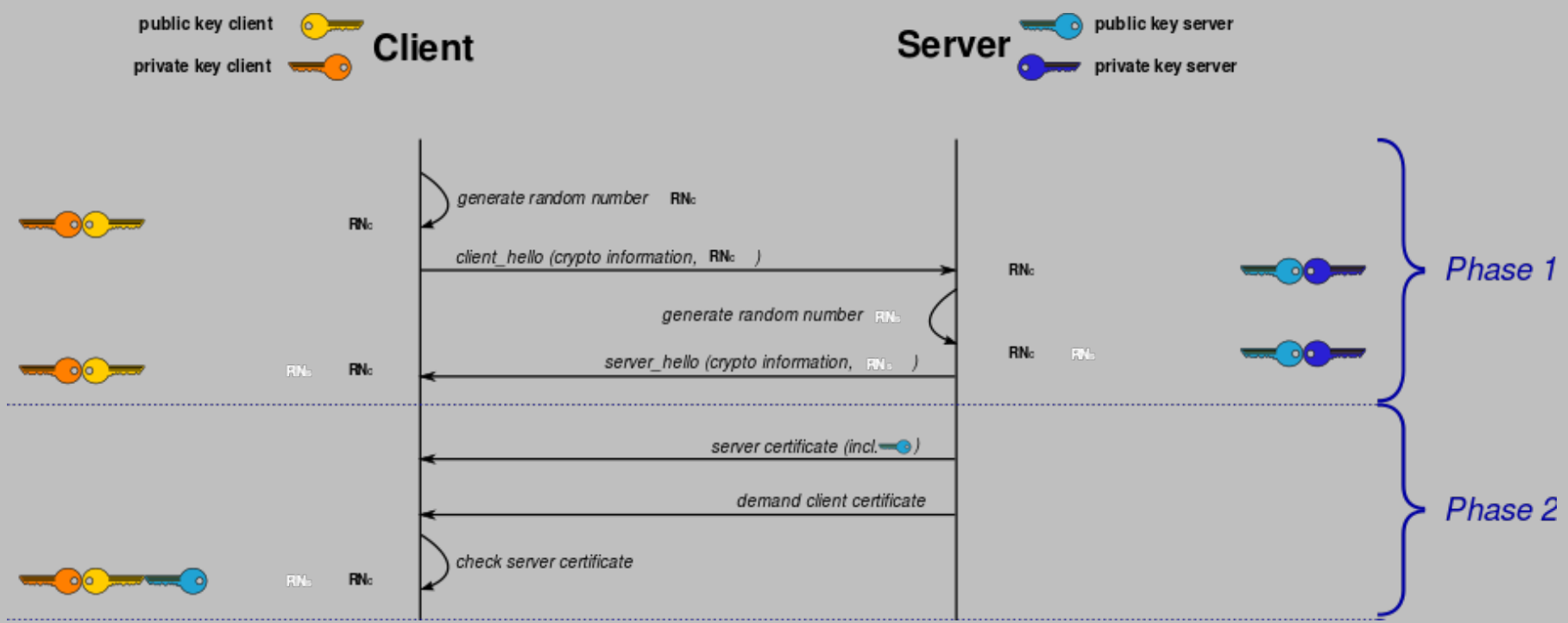
Transport Layer Security (TLS)

- TLS: Protokoll für den Aufbau und Betrieb sicherer Kanäle
  - Zuerst authentifizierter asymmetrischer Schlüsselaustausch...
  - ...danach symmetrische Verschlüsselung der Nutzdaten
  - Renegotiation („Neuaushandlung“ des Schlüssels) möglich
  - Viele Algorithmen möglich (RSA/DH, 3DES/AES/RC4, ...)
  - **Problem:** viele Kombinationen von Algorithmen/Varianten
- Aktuell Standard (OpenSSL/GnuTLS, `https`, `imaps`, ...)

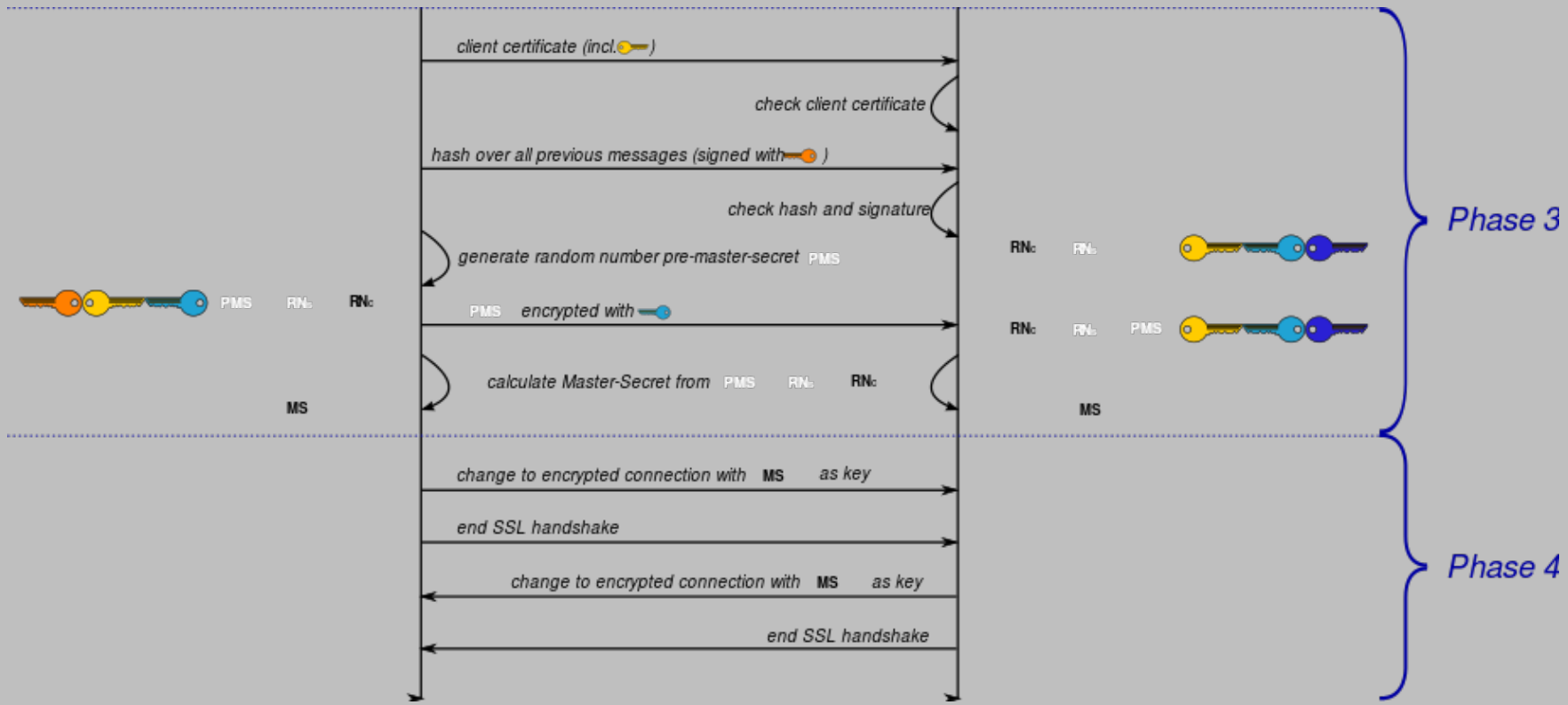
TLS Handshake (Schlüsselaustauschprotokoll)

(Quelle: `wikipedia.org`)

TLS Handshake (Schlüsselaustauschprotokoll)



TLS Handshake (Schlüsselaustauschprotokoll)



Geschichte von TLS

- Vorläufer SSL von TLS:
  - SSL 1.0 (1994) von Netscape für Mosaic-Browser
  - SSL 3.0 (1996) komplette Überarbeitung
- TLS aus SSL hervorgegangen:
  - TLS 1.0 (1999) $\approx$ SSL 3.0, erstmals in Windows 98 (SE)
  - TLS 1.1 (2006), TLS 1.2 (2008) gut unterstützt (außer IE)
  - TLS 1.3 aktuell IETF working draft
- Geschichtliche Einordnung (erfolgreiche Musik/Kinofilme):
  - 1994: „Hyper Hyper“ (Scooter), „Pulp Fiction“
  - 1996: „Wannabe“ (Spice Girls), „Independence Day“
  - 1999: „Baby One More Time“ (Britney Spears), „Star Wars I“
  - 2008: „Apologize“ (Timbaland), „Twilight“

Demo: Browserunterstützung

- **Demo:** `https://cc.dcsec.uni-hannover.de/`
  (Welche TLS-Version unterstützt mein Browser?)

Sicherheit von TLS

- Viele Patches (im Hinblick auf Angriffe) für konkrete Implementierungen
- Nach heutiger Kenntnis (in neuester Version, mit Patches, mit richtigen Parametern/Algorithmen betrieben!) sicher
- In idealen Modellen (**ideale Verschlüsselung**) sicher
  - Bedeutet, dass TLS-Verbindung „so sicher“ wie sicherer Kanal
  - **Allerdings:** Einige Angriffe (folgend) widersprechen dem nicht
- **Aber:** Angriffe auf konkrete Varianten/Kombinationen

Angriffe auf TLS

- Einige konkrete Angriffe auf TLS(-Varianten):
  - `ChangeCipherSpec` Drop (1996, auf SSL < 3.0)
  - Angriff auf schwaches RSA-padding (1998, 2014 POODLE)
  - CRIME (2002/2012, bei eingeschalteter Komprimierung)
  - TLS Renegotiation Attack (2009)
  - BEAST (2011, auf chained IVs im CBC-Mode, TLS<1.1)
  - Angriff auf RC4 (2013, braucht viele Sessions)
  - Lucky 13 (2013, Angriff auf CBC, braucht viele Sessions)
    Problem: TLS gibt aus, *warum* Dec fehlschlägt
  - (Heartbleed kein Protokoll- sondern Implementierungsproblem)
  - ...

Angriffe auf TLS

- Deshalb:
  - Aktuelle TLS-Version (1.1/1.2, mit Patches) benutzen
  - Kompression ausschalten
  - RC4 nicht benutzen (d.h. als Möglichkeit entfernen)
  - AES in nicht-CBC-Modus benutzen (z.B. AES-GCM)

Beispielangriff: `ChangeCipherSpec` Drop

- **Beobachtung:** Handshake-Ablauf bei SSL <3.0:
  1. Client und Server tauschen unverschlüsselt Informationen aus (public keys, CipherSuite-Präferenz, Sitzungsnummer, Authentifikation)
  2. Client sendet `ChangeCipherSpec` (mit Parametern), um auf verschlüsselte Kommunikation umzuschalten
  3. Client und Server senden `Finished` und beenden Handshake
- Angenommen, Server sendet sofort danach Nutzdaten
- **Angriff:** aktiver Angreifer unterdrückt `ChangeCipherSpec`
- **Konsequenz:** Server sendet Nutzdaten unverschlüsselt
- **Fix:** zusätzliche Bestätigung von Client vor Nutzdaten

Beispielangriff auf RSA-Padding

- **Beobachtung:** von SSL zum Key Transport genutzte RSA-Variante berechnet

$$C = \mathrm{Enc}(pk, \mathrm{pad}(M)) = (\mathrm{pad}(M))^e \bmod N$$

  mit „schlechtem“ (d.h. naivem) Padding (PKCS#1.5)
- Angriff auf schlechtes Padding seit 1998 bekannt
  - Idee: verändere C homomorph, beobachte, ob verändertes C noch als gültig akzeptiert wird
  - Allein aus Gültigkeit lässt sich Information über M ableiten
  - Benötigt viele (mehrere Tausend) Versuche

Beispielangriff auf RSA-Padding (konkreter)

- Konkreteres PKCS#1.5-Padding (vereinfacht):

$$C = \mathrm{pad}(K)^e = (\texttt{0x0002}||\texttt{rnd}||\texttt{0x00}||K)^e \bmod N$$

- K wird kurz sein (und immer mit vielen Nullbits anfangen)
- **Ziel:** finde viele gültige Faktoren α_i, so dass

$$\mathrm{Dec}(\alpha_i^e \cdot C \bmod N) = \underbrace{\alpha_i \cdot (\texttt{0x0002}||\texttt{rnd}||\texttt{0x00}||K) \bmod N}_{=:M_i}$$

gültig ist (so dass M_i mit 0x0002 beginnen muss)

Beispielangriff auf RSA-Padding (konkreter)

- **Ziel:** finde viele gültige Faktoren α_i, so dass

$$\mathrm{Dec}(\alpha_i^e \cdot C \bmod N) = \underbrace{\alpha_i \cdot (\texttt{0x0002}||\texttt{rnd}||\texttt{0x00}||K) \bmod N}_{=:M_i}$$

  gültig ist (so dass M_i mit 0x0002 beginnen muss)
  - Gültigkeit wird mit Injizieren von $\alpha_i^e \cdot C \bmod N$ überprüft
  - In alten SSL-Versionen sagt Server „RSA-Padding falsch“ *oder* „Key K zu groß“, sagt also, ob Padding gültig
- Viele gültige $M_i \Rightarrow$ grobes Intervall, in dem K liegt

Beispielangriff auf RSA-Padding (konkreter)

- Viele gültige $M_i \Rightarrow$ grobes Intervall, in dem K liegt
- Kombination geeignet vieler Intervallbestimmungen liefern K
- Bricht PKCS#1.5 (auch in TLS)
- **Gewählte Gegenmaßnahme:** patche SSL/TLS (wähle K zufällig, wenn Padding ungültig), obwohl RSA-OAEP bekannt

Beispielangriff: CRIME (Prinzip)

- **Annahme:** Kompression in TLS eingeschaltet
  - Übertragen wird komprimiertes $\mathrm{comp}(M)$ statt M
  - TLS nutzt DEFLATE-Kompression (LZ77/Huffman-Variante)
  - Prinzip: $\mathrm{comp}(\texttt{Fliegen fliegen}) = \texttt{Fliegen f(-8,6)}$
- **Strategie:** verändere *vorangehenden* Klartextteil, um zu sehen, ob *folgender* (unbekannter) Klartextteil komprimiert
- **Beispiel:** bringe Client dazu, zunächst 1234 zu senden, anschließend seine vierstellige numerische PIN ABCD
  - Client verschlüsselt $C := \mathrm{Enc}(K, \mathrm{comp}(1234\texttt{ABCD}))$
  - Länge $|C|$ hängt direkt ab von Länge $|\mathrm{comp}(1234\texttt{ABCD})|$
  - Insbesondere: Chiffrat am kürzesten, wenn ABCD $= 1234$

Beispielangriff: CRIME (konkreter)

- **Konkreter:** Angreifer möchte Cookie von Client stehlen
  - Client hat laufende Session mit Server (z.B. `www.ebay.com`)
  - Client hat *geheimes* HTTP-Cookie `ABCD`, das Browser automatisch an alle Nachrichten an Server anhängt
  - Angreifer schleust JavaScript-Code bei Client ein
  - JavaScript-Code lässt Client `WXYZABCD` an Server senden
  - Angreifer belauscht $C := \mathrm{Enc}(K, \mathrm{comp}(\texttt{WXYZABCD}))$, erfährt so Länge $|\mathrm{comp}(\texttt{WXYZABCD})|$, und „wie nahe `WXYZ` an `ABCD` ist“
  - Viele Wiederholungen/`WXYZ` $\Rightarrow$ Angreifer erfährt Cookie
- **Wichtig:** Böser JavaScript-Code kennt nur `WXYZ`, nicht `ABCD`, kann aber Client dazu bringen, `WXYZABCD` an Server zu senden

Zusammenfassung TLS

- TLS historisch gewachsen, Quasi-Standard, hochrelevant
- Viele Versionen, Einstellungsmöglichkeiten, Angriffe, Fixes
- **Frage:** warum unterstützen viele Browser (bis vor etwa 5 Jahren) nur 1999-Standard?
- **Antwort:** Angriffe aufwändig, Kompatibilität
- **Frage:** warum verwendet man nicht bessere Algorithmen? (z.B. RSA-OAEP statt schlecht gepaddetem RSA mit Hotfix)
- **Antwort:** einfacher zu fixen als zu ersetzen