# Beweisbar sichere Verschlüsselung

ITS-Wahlpflichtvorlesung

Dr. Bodo Möller
Ruhr-Universität Bochum
Horst-Görtz-Institut für IT-Sicherheit
Lehrstuhl für Kommunikationssicherheit
bmoeller@crypto.rub.de





## Fallbeispiel: SSL/TLS

- Bisher in der Vorlesung: Diverse formale Angriffsspiele, Vorteile, Sicherheitsbeweise durch Reduktion ...

  Warum das alles?

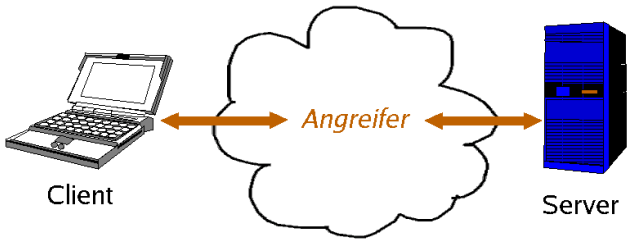
- *Ad-hoc-Konstruktionen* (ohne Sicherheitsbeweis) sind *oft fehlerhaft*!

  Fehler sind oft nicht leicht zu finden – ohne Beweis bleibt im Zweifel offen, ob es Fehler gibt

- Als Fallbeispiel: *SSL* (Secure Socket Layer), *TLS* (Transport Layer Security)

# Fallbeispiel: SSL/TLS (Forts.)

- *SSL* (Secure Socket Layer), *TLS* (Transport Layer Security): universelles kryptographisches Protokoll für Client-Server-Kommunikation per Byte-Strom (z. B. für WWW, E-Mail, ...)




- *Ziele*: Vertraulichkeit und Integrität der Daten, Authentisierung
- Übliche Protokollversionen: SSL 3.0, TLS 1.0 (= „SSL 3.1“)



# Fallbeispiel: SSL/TLS (Forts.)

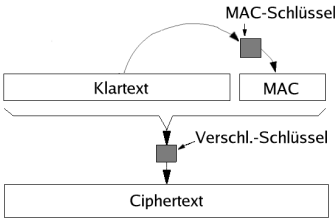
- Erst *Handshake* mit *Schlüsselaustausch,* dann *symmetrische Kryptographie* für Anwendungsdaten

```
Client                                               Server

ClientHello                  -------->
                                                ServerHello
                                               Certificate*
                                         ServerKeyExchange*
                                        CertificateRequest*
                             <--------      ServerHelloDone
Certificate*
ClientKeyExchange
CertificateVerify*
[ChangeCipherSpec]
Finished                     -------->
                                          [ChangeCipherSpec]
                             <--------             Finished

Application Data             <------->     Application Data

       Fig. 1 - Message flow for a full handshake

* Indicates optional or situation-dependent messages that are not
always sent.                                          [RFC 2246]
```

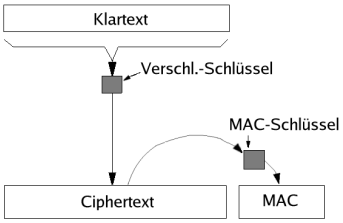
- Hier betrachten wir nur die symmetrische Kryptographie; Ziel: *authentisierte Verschlüsselung*

# Fallbeispiel: SSL/TLS (Forts.)

- Symmetrische Kryptographie in SSL/TLS: "*MAC-then-Encrypt*"




- Die *Sicherheit bewiesen* hatten wir dagegen für "*Encrypt-then-MAC*"
  (*LoR-CCA* und *INT-CTXT* bei LoR-CPA-sicherer Verschlüsselung und sicherer MAC)






# Fallbeispiel: SSL/TLS (Forts.)

- Der Sicherheitsbeweis (für Encrypt-then-MAC) sagt nichts aus zu SSL/TLS (MAC-then-Encrypt)
- Die *kryptographischen Primitive* in SSL/TLS können wir als *sicher* annehmen:
  - *Verschlüsselung* mit „*Stromchiffre*" (d. h. XOR mit PRG-Wert)
    → Stromchiffre/PRG als Primitive ist vermutlich *PRG-sicher*
  - ... oder *Verschlüsselung* mit *Blockchiffre* als *Cipher Block Chaining* (CBC)
    → Blockchiffre als Primitive ist vermutlich *PRP-sicher* (und mehr)
  - *MAC* vermutlich *sicher* (z B. SHA1-HMAC; hier keine Details)
- Aber ist die "*MAC-then-Encrypt*"-Konstruktion *sinnvoll*?

# Fallbeispiel: SSL/TLS (Forts.)

- Kryptographischen Primitive in SSL/TLS können wir als sicher annehmen
- Ist die "*MAC-then-Encrypt*"-Konstruktion *sinnvoll*?

- Für *MAC-then-Encrypt mit Stromchiffre* in SSL/TLS wurde nachträglich die *Sicherheit bewiesen* (hier keine Details)

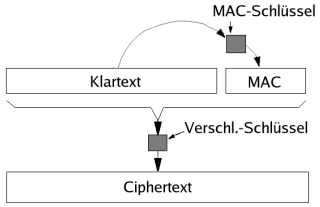
- Für *MAC-then-Encrypt mit CBC* sehen wir genauer hin:

  SSL 3.0, TLS 1.0, TLS 1.1 (leicht vereinfacht)

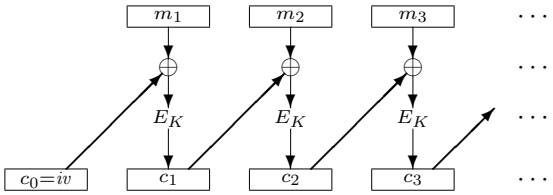


# MAC-then-Encrypt mit CBC in SSL 3.0

- MAC-then-Encrypt:




- CBC-Verschlüsselung:




- *Klartext* ∥ *MAC* muss *auf Vielfaches der Blocklänge* gebracht werden, um CBC-verschlüsseln zu können ("*Padding*")

# MAC-then-Encrypt mit CBC in SSL 3.0 (Forts.)

- *Klartext* ∥ *MAC* muss *auf Vielfaches der Blocklänge* gebracht werden, um CBC-verschlüsseln zu können ("*Padding*")
- *Blocklänge:* z. B. 8 Bytes
  *Klartext:* in SSL/TLS immer eine *ganze Anzahl Bytes*
  *MAC:* z. B. 10 Bytes
- Zusätzliche *Padding*-Bytes zum Erreichen eines Vielfachen der Blocklänge:

$$\underbrace{\cdots \boxed{K\,K\,K\,K\,K\,K\,K\,K}\,\boxed{K\,K\,M\,M\,M\,M\,M\,M}\,\boxed{M\,M\,M\,M\,P\,P\,P\,P}}_{\text{Eingabe zur CBC-Verschlüsselung}}$$

($K$: Klartext-Byte, $M$: MAC-Byte, $P$: Padding-Byte)

- Nach der CBC-*Entschlüsselung* muss das *Padding* wieder *entfernt* werden; das letzte Byte gibt dafür die Padding-Länge an
- SSL 3.0: Padding mit 00 oder $P$ 01 oder $P\,P$ 02 oder $P\,P\,P$ 03 oder ...



# MAC-then-Encrypt mit CBC in SSL 3.0 (Forts.)

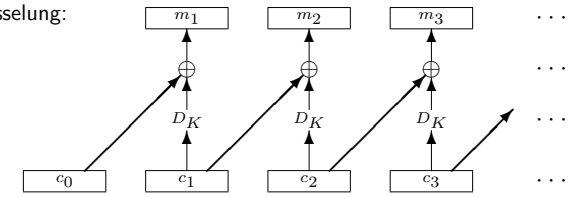
- Vorgehen *beim Empfänger* nach CBC-Entschlüsselung:

$$\overbrace{\cdots \boxed{K\,K\,K\,K\,K\,K\,K\,K}\,\boxed{K\,K\,M\,M\,M\,M\,M\,M}\,\boxed{M\,M\,M\,M\,P\,P\,P\,03}}^{\text{Ausgabe der CBC-Entschlüsselung}}$$

  *Entferne die letzten Bytes* (im Beispiel: $P\,P\,P$ 03) als Padding, prüfe dann MAC $M\,M\,M\,M\,M\,M\,M\,M\,M\,M$ auf Klartext $K\,K\,K\,\cdots\,K\,K\,K$
- Bei irgendeinem *Fehler* (unsinnige Padding-Längenangabe, fehlgeschlagene MAC-Überprüfung): Die SSL/TLS-*Verbindung wird beendet*
- Ein solches Verbindungsende ist *für einen Angreifer sichtbar!*
- Ein *aktiver Angreifer* kann CBC-verschlüsselte Daten gezielt manipulieren, aus Beobachtung der Reaktion (Fehler oder Nicht-Fehler?) etwas *Information erhalten* ...

# MAC-then-Encrypt mit CBC in SSL 3.0 (Forts.)

- CBC-Entschlüsselung:




- Struktur der CBC-entschlüsselten Daten:

$$\cdots \boxed{K\,K\,K\,K\,K\,K\,K\,K}\,\boxed{K\,K\,M\,M\,M\,M\,M\,M}\,\boxed{M\,M\,M\,M\,P\,P\,P\,03}$$

- Ein *aktiver Angreifer* kann CBC-verschlüsselte Daten gezielt manipulieren, aus Beobachtung der Reaktion (Fehler oder Nicht-Fehler?) etwas *Information erhalten:*
- Einfachster Fall für einen Angriff: ein *voller Block* mit *Padding*

$$\cdots \boxed{K\,K\,K\,K\,K\,K\,M\,M}\,\boxed{M\,M\,M\,M\,M\,M\,M\,M}\,\boxed{P\,P\,P\,P\,P\,P\,P\,07}$$

- Wenn der *Angreifer* den *letzten Ciphertextblock abändert,* ergibt sich:

$$\cdots \boxed{K\,K\,K\,K\,K\,K\,M\,M}\,\boxed{M\,M\,M\,M\,M\,M\,M\,M}\,\boxed{P'\,P'\,P'\,P'\,P'\,P'\,P'\,??}$$



# MAC-then-Encrypt mit CBC in SSL 3.0 (Forts.)

- *Annahme:* ein *voller Block* mit *Padding*
  Vom Sender CBC-verschlüsselt:

$$\cdots \boxed{K\,K\,K\,K\,K\,K\,M\,M}\,\boxed{M\,M\,M\,M\,M\,M\,M\,M}\,\boxed{P\,P\,P\,P\,P\,P\,P\,07}$$

- Wenn der *Angreifer* den *letzten Ciphertextblock abändert,* ergibt sich beim Entschlüsseln:

$$\cdots \boxed{K\,K\,K\,K\,K\,K\,M\,M}\,\boxed{M\,M\,M\,M\,M\,M\,M\,M}\,\boxed{P'\,P'\,P'\,P'\,P'\,P'\,P'\,??}$$

- Der Empfänger wird das akzeptieren (ohne jeglichen Fehler), falls $?? = 07$, falls also $c_{n-1} \oplus D_K(c) = \ldots 07$
  ($c_{n-1}$: vorletzter Ciphertextblock, $c$: letzter Ciphertextblock nach Änderung)
- Der Angreifer kann für $c$ einen *früheren Ciphertextblock* $c_x$ wiederverwenden, vom Sender erzeugt beim Verschlüsseln von $m_x$ $(= c_{x-1} \oplus D_K(c_x))$;

  dann wird die geänderte Nachricht akzeptiert,
  falls zufällig $c_{n-1} \oplus \underbrace{m_x \oplus c_{x-1}}_{=D_K(c_x)} = \ldots 07$ ist.

  $c_{n-1}$ und $c_{x-1}$ sind für den Angreifer sichtbar;
  *durch den Angriff erfährt er ein wenig über den Klartext* in $m_x$

# MAC-then-Encrypt mit CBC in SSL 3.0 (Forts.)

- Also: Ein *aktiver Angreifer* kann bei CBC-Verschlüsselung SSL 3.0 unter Umständen *ein wenig Information* über einen Klartext erhalten.
- Sehr geringes Informations-Leck, aber schon das widerlegt die kryptographische Sicherheit
- Diese Verschlüsselung ist *nicht INT-CTXT-sicher* und *nicht CCA-sicher!*
- Wie sieht es bei TLS 1.0 aus ...?



# MAC-then-Encrypt mit CBC in SSL 3.0 und TLS 1.0

- *TLS 1.0:* andere Padding-Handhabung ...

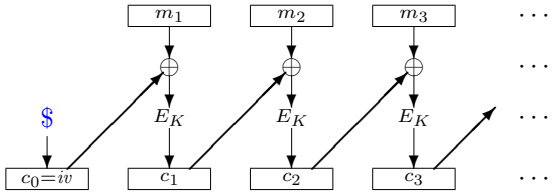
$$\underbrace{\cdots \boxed{K\ K\ K\ K\ K\ K\ K\ K}\ \boxed{K\ K\ M\ M\ M\ M\ M\ M}\ \boxed{M\ M\ M\ M\ P\ P\ P\ P}}_{\text{Eingabe zur CBC-Verschlüsselung}}$$

  Padding mit 00 oder 01 01 oder 02 02 02 oder 03 03 03 03 oder ...

- Der Empfänger kann und muss *jedes Byte überprüfen!*
- Ein Angriff wie vorhin (SSL 3.0) ist deshalb *nicht möglich*
- Aber ...

# MAC-then-Encrypt mit CBC in SSL 3.0 und TLS 1.0 (Forts.)

- Aber SSL 3.0 hat noch ein *anderes Problem,* das in TLS 1.0 nicht gelöst ist
- Der *IV* (Initialisierungsvektor) für die CBC-Verschlüsselung sollte *zufällig* sein (das war Voraussetzung für den Sicherheitsbeweis):

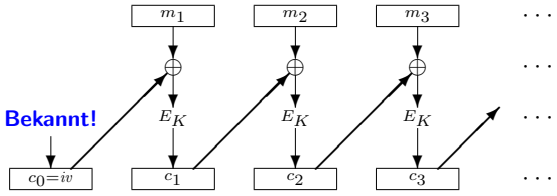



- Bei SSL 3.0 und TLS 1.0 wird mehrfach CBC-verschlüsselt,
  nur der erste IV ist (pseudo-)zufällig;
  danach dient jeweils der *letzte gesendete Ciphertextblock* als *neuer IV*
- Das heißt, der *IV* ist bereits *vor der Verschlüsselung bekannt!*
- Das ist ein Problem, wenn der Angreifer den Klartext wählen kann (Verschlüsselungsorakel!)



# MAC-then-Encrypt mit CBC in SSL 3.0 und TLS 1.0 (Forts.)

- Nur der erste IV ist (pseudo-)zufällig;
  danach dient jeweils der *letzte gesendete Ciphertextblock* als *neuer IV*




- Der Angreifer kann $m_1$ gezielt so zu $c_0$ auswählen, dass $c_0 \oplus m_1$ als Eingabe zu $E_K$ eine „Wiederholung“ von früher ist $(= c'_{j-1} \oplus m'_j)$
- Angriffe z. B. im “Left-or-Right”- oder “Real-or-Random”-Modell:
  Angreifer kann testen, ob die CBC-Verschlüsselung tatsächlich $m_1$ und $m'_j$ verwendet hat

## MAC-then-Encrypt mit CBC in TLS 1.1

- Erst die Protokollversion *TLS 1.1* vermeidet auch dieses Problem
- TLS 1.1 hat für CBC keine „IV-Verkettung“ wie SSL 3.0 und TLS 1.0, sondern beim *CBC*-Verschlüsseln werden *stets neue IVs* erzeugt
- TLS 1.1 bleibt bei “*MAC-then-Encrypt*”:
  Für das *konkrete* Verfahren mit CBC wurde die Sicherheit bewiesen!

  (Der Beweis ist spezieller als der Sicherheitsbeweis für Encrypt-then-MAC.
  Er benötigt strengere Sicherheitsvoraussetzungen für die Blockchiffre als die,
  die wir zur CBC-Verschlüsselung kennengelernt haben.)



## Fazit

- Scheinbar kleine Details können große Folgen für die Sicherheit haben
- Deshalb ist „beweisbare Sicherheit“ wertvoll,
  auch wenn sie keinen vollständigen Beweis der Sicherheit bietet
  (denn die Sicherheit der Primitive ist eine unbewiesene Annahme):

  Entwurfsfehler wie bei SSL 3.0 und TLS 1.0 lassen sich vermeiden