



SSL Security Event

Compass Security / keyon

24. April 2003

Walter Sprenger
Dipl. El.-Ing. FH

Markus Isler
Dipl Informatik-Ing. ETHZ
CFO



Motivation



■ **Wieso dieser Event ?**

- Schwachstellen in OpenSSL

■ **Wieso Compass Security und keyon ?**

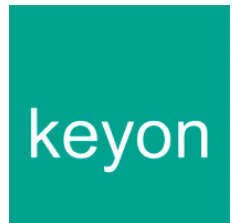
- Compass Security und keyon ergänzen sich

■ **Ziel des Events**

- Um welche Schwachstellen handelt es sich ?
- Wie funktionieren die Schwachstellen ?
- Welche Produkte sind betroffen ?
- Wie Wahrscheinlich ist ein erfolgreiches Ausnützen ?
- Welche Aktivitäten ergeben sich aufgrund der Schwachstellen ?



Compass Security



■ Penetration Tests

- Hacking im Auftrag
- Wardriving/Wardialing
- Vulnerability Scans

■ Security Review

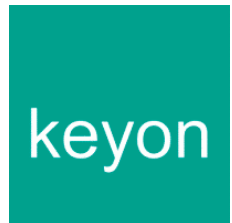
- Review von Applikationen und Infrastrukturen
- Umfassende Security Assessments

■ IT Security Training

- ISACA Kurse (Applikations- und Internet-Sicherheit)
- Technologietransfer zu Fachhochschule Rapperswil
- Forschungsprojekte mit ISB.ADMIN.CH



keyon



■ **Beratung und Projekt Management**

- Rechtliche, organisatorische und technische Beratung im Umfeld von e-Business Lösungen
- Erarbeiten und Umsetzen von e-Business Prozessen

■ **IT-Security**

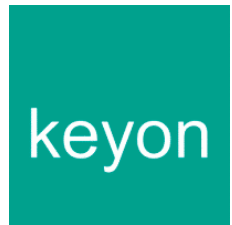
- Erarbeiten und Umsetzen von Sicherheitskonzepten
- PKI- und SSO Lösungen, PKI Enabling von Applikationen
- Integrationen von Smart Cards und Hardware Security Modulen

■ **Kundenspezifische Entwicklungen**

- Realisierung von komplexen Client- Server Lösungen basierend auf modernsten Technologien (J2EE, Web Services, .NET)
- Erweiterung und Integration bestehender Applikationen
- GUI basierte Desktop Frontends



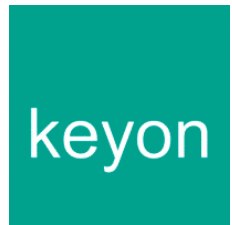
Agenda



- **Begrüßung**
- **Demo EPFL Schwachstelle**
- **Grundlagen**
- **Voraussetzungen fürs Ausnützen**
- **Erklärungen zu CBC und private Key Attacke**
- **Verhinderung der Attacken**
- **Fragen/Antworten/Diskussion**

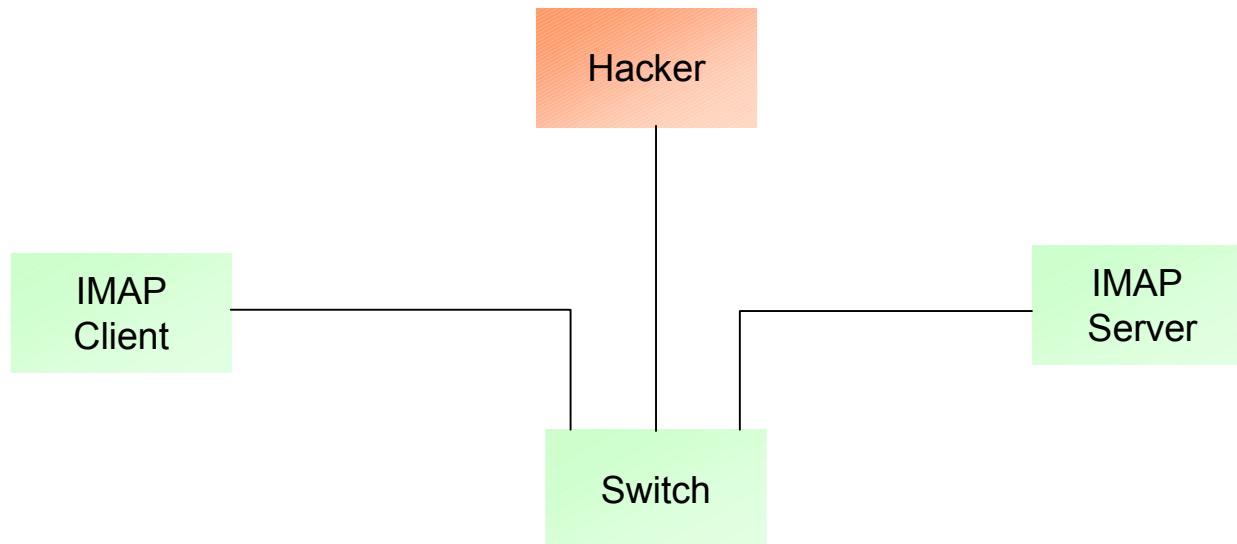


Agenda



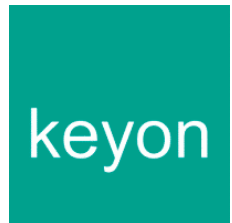
- Begrüssung
- **Demo EPFL Schwachstelle**
- Grundlagen
- Voraussetzungen fürs Ausnützen
- Erklärungen zu CBC und private Key Attacke
- Verhinderung der Attacken
- Fragen/Antworten/Diskussion

■ Netzwerk-Topologie (physikalisch)





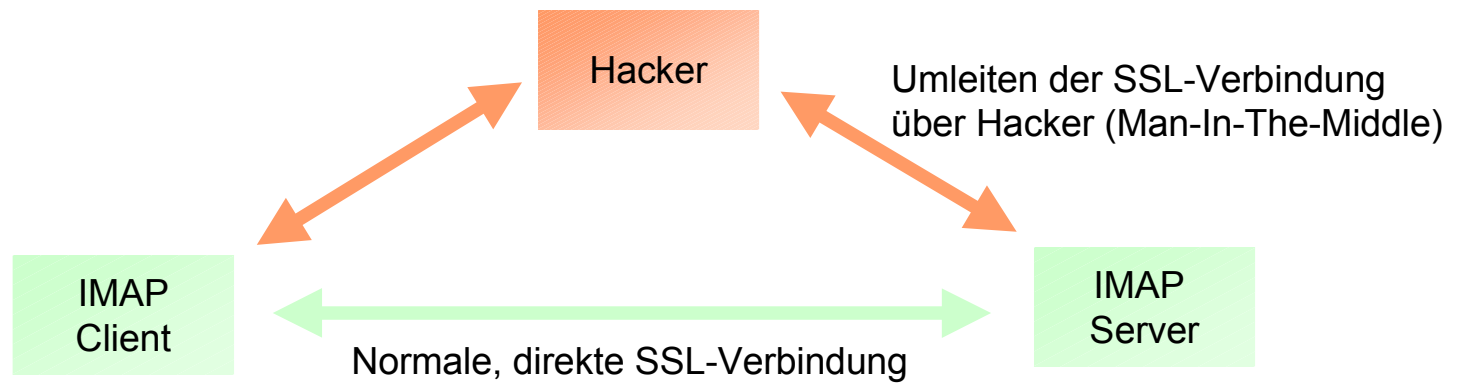
Demo Netzwerk



■ Netzwerk-Komponenten

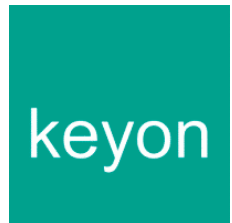
- IMAP-Client
 - Outlook Express als IMAP Client (IMAP over SSL)
 - Compass-Tool zum automatischen Mail-Sync
- Hacker
 - Linux Default-Installation
 - Omen (EPFL Demo Tool)
- IMAP-Server
 - IMAP Server (WU-IMAPrev1)
 - sTunnel mit OpenSSL 0.9.7
- Netzwerk
 - Switched Twisted-Pair Ethernet

■ Netzwerk-Verbindungen



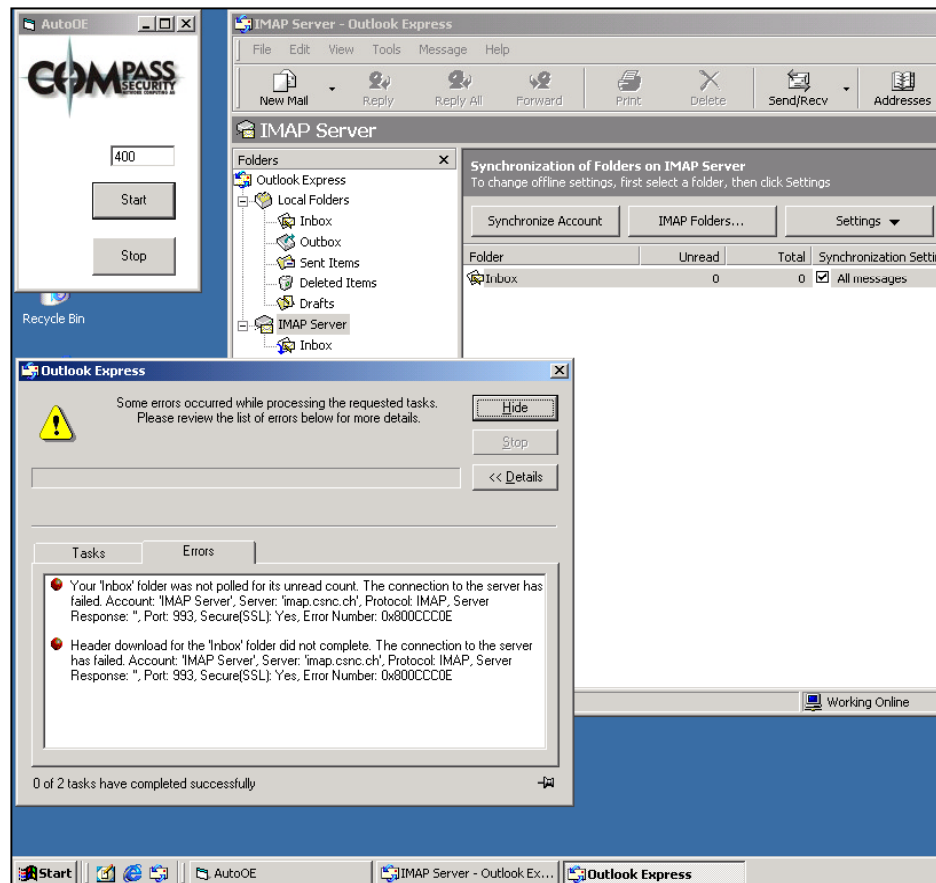


Man-In-The-Middle



- **Umleiten einer SSL-Verbindung**
 - ARP Spoofing
 - DNS Spoofing
 - Client-Konfiguration ändern (Virus, Trojaner)
 - Hosts-Datei verändern
 - Outlook Config verändern
 - Kontrolle über Netzwerk-Komponenten
 - Routing verändern

■ IMAP Client mit AutoSync-Tool





Screenshots Demo



■ EPFL Demo Tool: OMEN

```
debug]
decryption_failed

Testing A (0x41) 5340 [us]
decryption_failed

Testing B (0x42) 5381 [us]
decryption_failed

Testing C (0x43) 5370 [us]
decryption_failed

Testing D (0x44) 5410 [us]
decryption_failed

Testing E (0x45) 5328 [us]
decryption_failed

Testing F (0x46) 5395 [us]
decryption_failed

Testing G (0x47) 5351 [us]
decryption_failed

Testing H (0x48) 5389 [us]
decryption_failed

Testing I (0x49) 5813 [us]
bad_record_mac

Recheck...

Testing I (0x49) 5809 [us]
bad_record_mac

Testing 5338 [us]
decryption_failed

Testing ^A (0x1) 5376 [us]
decryption_failed

Testing ^B (0x2) 5338 [us]
decryption_failed

Testing ^C (0x3) 5373 [us]
decryption_failed

Testing ^D (0x4) 5340 [us]
decryption_failed

passwd]
????????IN "hmei?????? ?????????????????????????????????????

[TLS]
[C]<-----[HS]-[srvHello]----->[S]
[C]<-----[HS]-[SrvCert]----->[S]
[C]<-----[HS]-[HelloDone]----->[S]
[C]-----[HS]-[CipherXch]----->[S]
[C]-----[chngCipherSpec]----->[S]
[C]-----[HS]-[Finished]----->[S]
[C]<-----[chngCipherSpec]----->[S]
[C]<-----[HS]-[Finished]----->[S]
[C]<-----[App-Data]-[ 1]----->[S]
[C]<-----[App-Data]-[ 2]----->[S]
[C]-----[App-Data]-[ 3]----->[S]
[C]<-----[App-Data]-[ 4]----->[S]
[C]<-----[App-Data]-[ 5]----->[S]
[C]<-----[App-Data]-[ 6]----->[S]
[C]<-----[ALERT]----->[S]

[C]-----[HS]-[cliHello]----->[S]
[C]<-----[HS]-[srvHello]----->[S]
[C]<-----[HS]-[SrvCert]----->[S]
[C]<-----[HS]-[HelloDone]----->[S]
[C]-----[HS]-[CipherXch]----->[S]
[C]-----[chngCipherSpec]----->[S]
[C]-----[HS]-[Finished]----->[S]
[C]<-----[chngCipherSpec]----->[S]
[C]<-----[HS]-[Finished]----->[S]
[C]<-----[App-Data]-[ 1]----->[S]
[C]<-----[App-Data]-[ 2]----->[S]

[C]-----[HS]-[cliHello]----->[S]
[C]<-----[HS]-[srvHello]----->[S]
[C]<-----[HS]-[SrvCert]----->[S]
[C]<-----[HS]-[HelloDone]----->[S]
[C]-----[HS]-[CipherXch]----->[S]
[C]-----[chngCipherSpec]----->[S]
[C]-----[HS]-[Finished]----->[S]
[C]<-----[chngCipherSpec]----->[S]
[C]<-----[HS]-[Finished]----->[S]
[C]<-----[App-Data]-[ 1]----->[S]
[C]<-----[App-Data]-[ 2]----->[S]
```



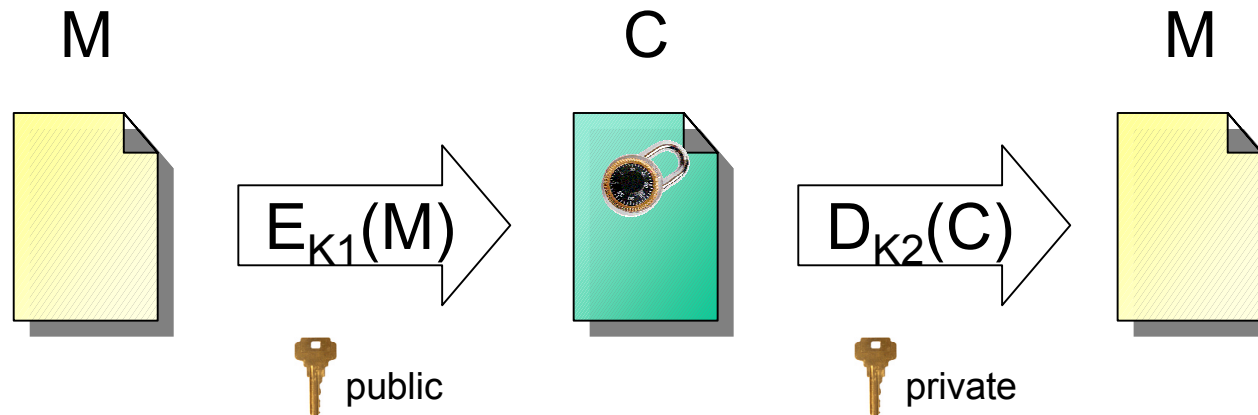
Agenda



- Begrüssung
- Demo EPFL Schwachstelle
- **Grundlagen**
- Voraussetzungen fürs Ausnützen
- Erklärungen zu CBC und private Key Attacke
- Verhinderung der Attacken
- Fragen/Antworten/Diskussion

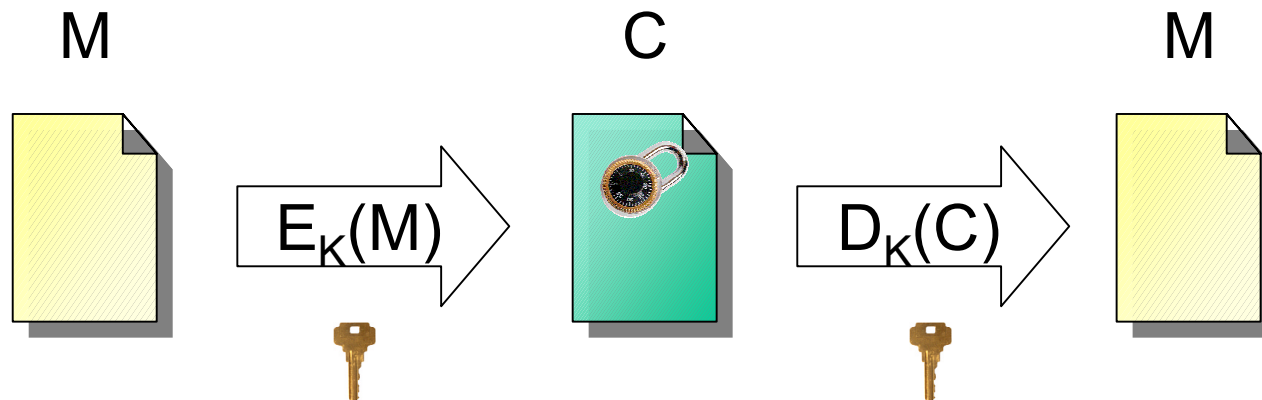
■ Public-Key (asymmetrisches) Kryptosystem

- Verschlüsselung (RSA)
- Signaturen (RSA, DSA)



■ Konventionelles (symmetrisches) Kryptosystem

- Verschlüsselung
- Block-Cipher (DES, 3-DES, IDEA, RC5)
- Stream-Cipher (RC4)





Boolsche Algebra



■ XOR 

$$0 \text{ XOR } 0 = 0$$

$$0 \text{ XOR } 1 = 1$$

$$1 \text{ XOR } 0 = 1$$

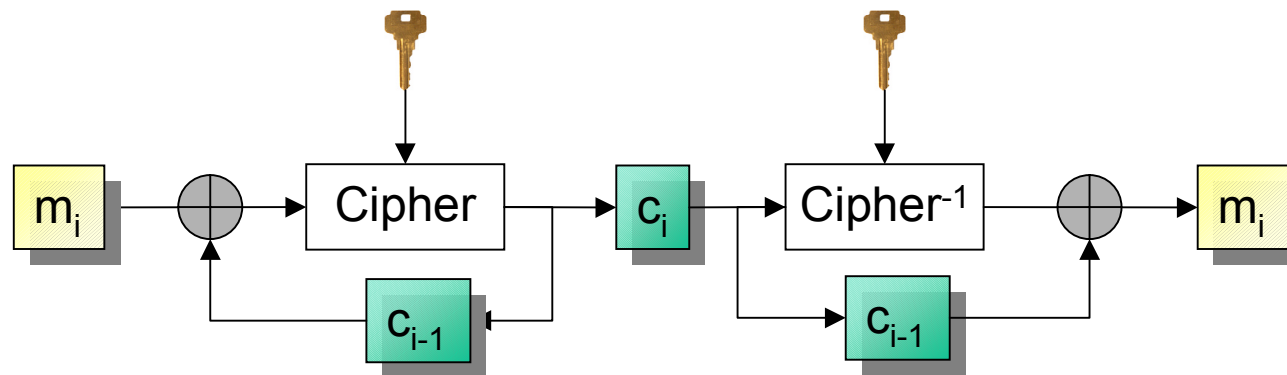
$$1 \text{ XOR } 1 = 0$$

■ Blockcipher

- DES, IDEA, 3-DES, RC5
- Blocklänge (z.B. 8 Byte)
- Padding

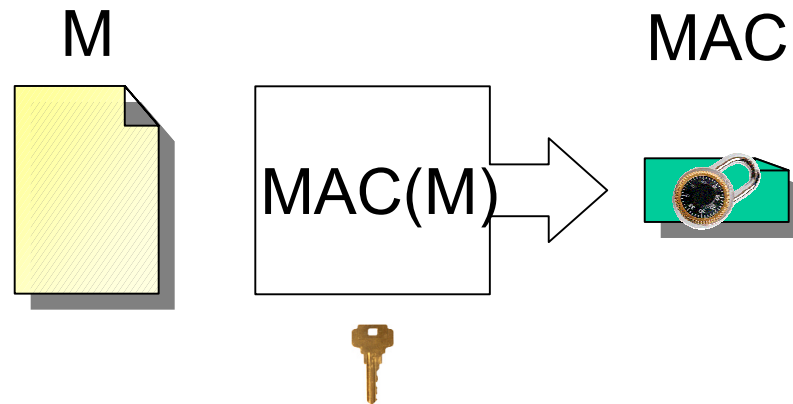


- CBC Cipher-Block-Chaining



■ Message Authentication Code (MAC)

- Datenauthentifikation
- Hashfunktion (SHA-1, MD5)
- Schlüssel





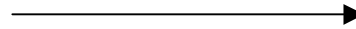
SSL Protocol



■ SSL Handshake

Client

ClientHello



Server

ServerHello

Certificate

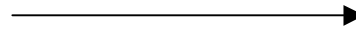
ServerHelloDone



ClientKeyExchange

ChangeCipherSpec

Finished



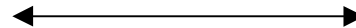
ChangeCipherSpec

Finished



■ Data

Application Data



Application Data



SSL Record Layer



■ SSLCiphertext



■ Generic Block Cipher





Attacken Übersicht

keyon

■ CBC Padding Attacke

- SSL, IPSEC, WTLS
- Entschlüsseln von strukturierten Daten (SSL Record Layer)

■ RSA Private Key Attacke

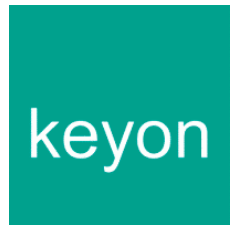
- RSA Implementation
- Herausfinden des RSA Private Keys
- OpenSSL (SSL Handshake)

■ RSA basierte Session Attacke

- SSL, TLS (SSL Handshake)
- Signieren mit RSA Private Key
- Entschlüsseln des SSL Master Secrets



Side-Channel Attacken



■ Side Channel Attacken

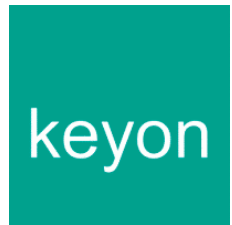
- Indirekter Angriff auf Algorithmus über Seitenkanal
- Schwachstelle in Implementation
- Schwachstelle in Protokoll
- Timing Attacke

■ Implementations-Schwachstellen

- Algorithmus ist ok
- Implementations-Fehler
- Protokoll Fehler



Agenda



- Begrüssung
- Demo EPFL Schwachstelle
- Grundlagen
- **Voraussetzungen fürs Ausnützen**
- Erklärungen zu CBC und private Key Attacke
- Verhinderung der Attacken
- Fragen/Antworten/Diskussion

■ CBC Padding Attacke

- Verschlüsselung im CBC-Modus
 - 80% wird RC4-MD5 (kein CBC) verwendet
- Man-In-The-Middle
 - TCP Datenpakete müssen verändert werden können
- Konstantes Zeitverhalten
 - "Nahe" beim Server
- Viele (ca. 2000) identische Verbindungen
 - Verbindung wird bei jedem Versuch abgebrochen und muss neu aufgebaut werden
 - Zu entschlüsselnde Daten immer an derselben Stelle

■ RSA basierte Session Attacke

- Man-In-The-Middle
 - Angreifer muss Verkehr belauschen können

- Konstantes Zeitverhalten
 - "Nahe" beim Server

- Viele (ca. 2.7 Millionen) identische Verbindungen
 - Verbindung wird bei jedem Versuch abgebrochen und muss neu aufgebaut werden

- SSL Verbindung kann erst im Nachhinein entschlüsselt werden



Voraussetzungen



■ RSA Private Key Attacke

- Kein Man-In-The-Middle
 - Angreifer ist selbst ein SSL Client
- Konstantes Zeitverhalten
 - "Nahe" beim Server
- Viele (ca. 1.5 Millionen) Verbindungen
 - können über längeren Zeitraum verteilt werden

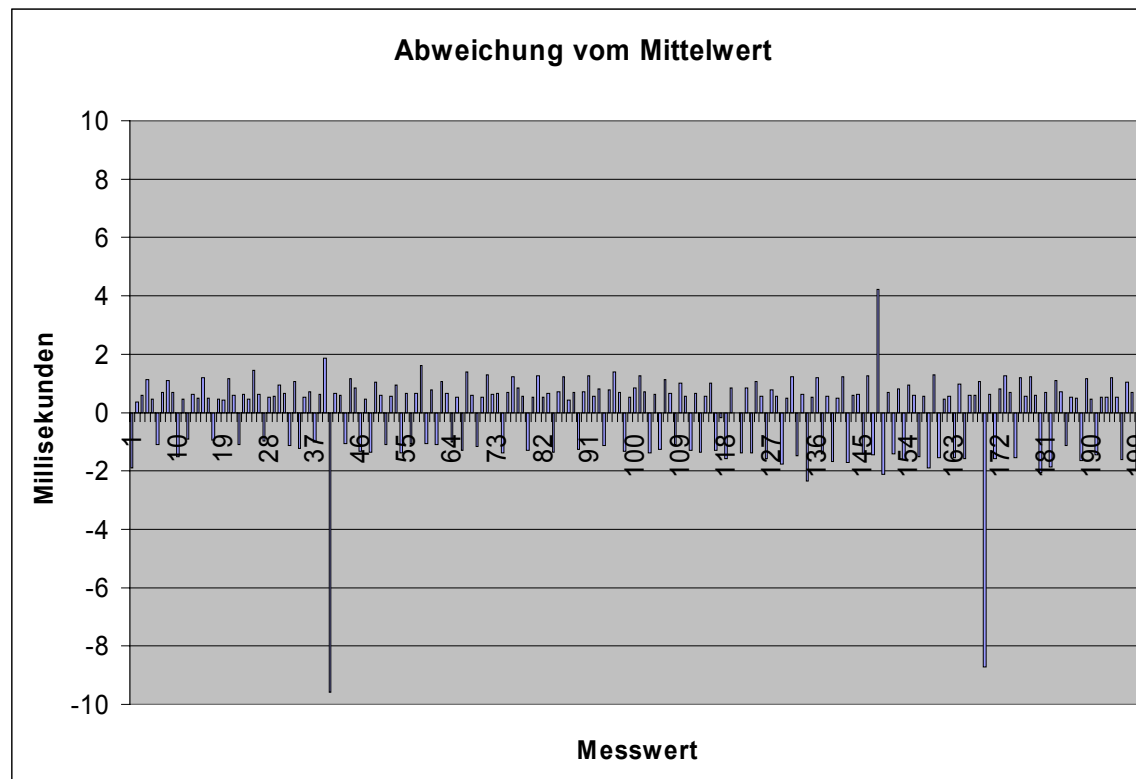
■ **Wahrscheinlichkeit: Zeitverhalten**

- Zeitunterschiede bei den 3 OpenSSL Timing Attacks
 - im Milisekunden Bereich

- Voraussetzung für konstantes Zeitverhalten:
 - Server steht nicht unter Last
 - Netzwerkkomponenten verzögern jedesmal gleich lang
 - wenig Netzverkehr

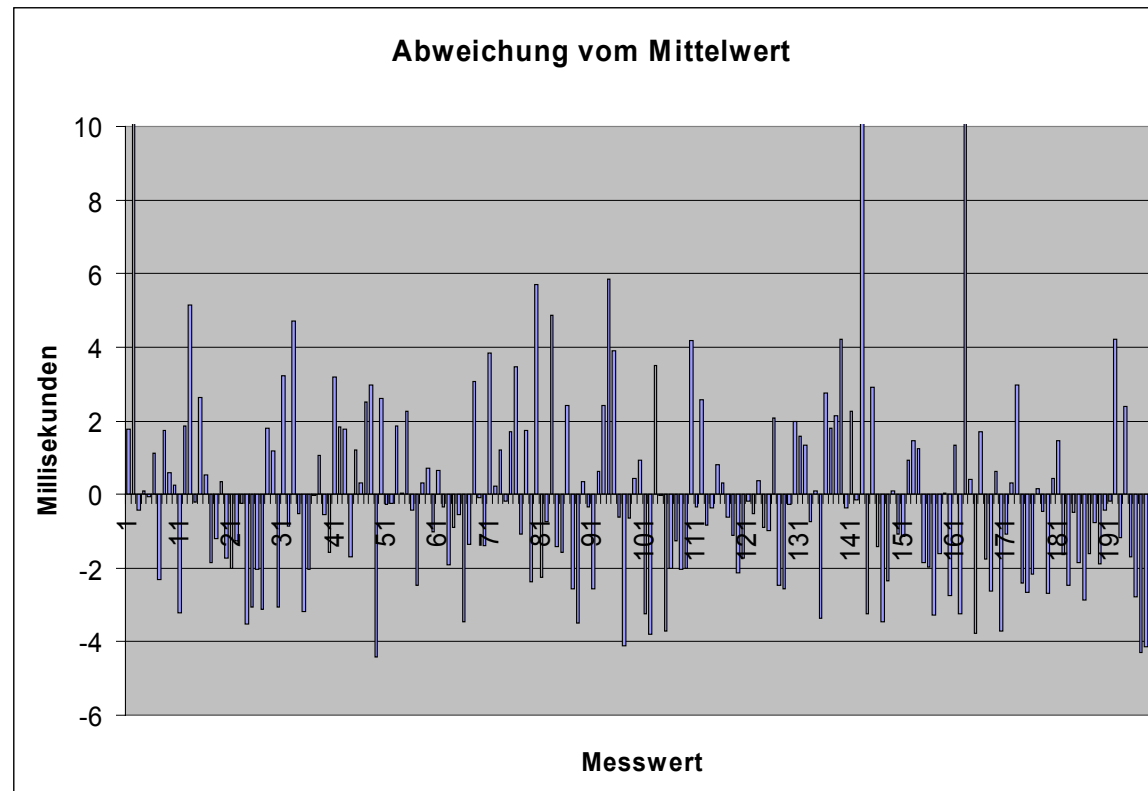
■ Wahrscheinlichkeit: Zeitverhalten

- Verzögerungen im Labor-Netzwerk (nur Client und Server)



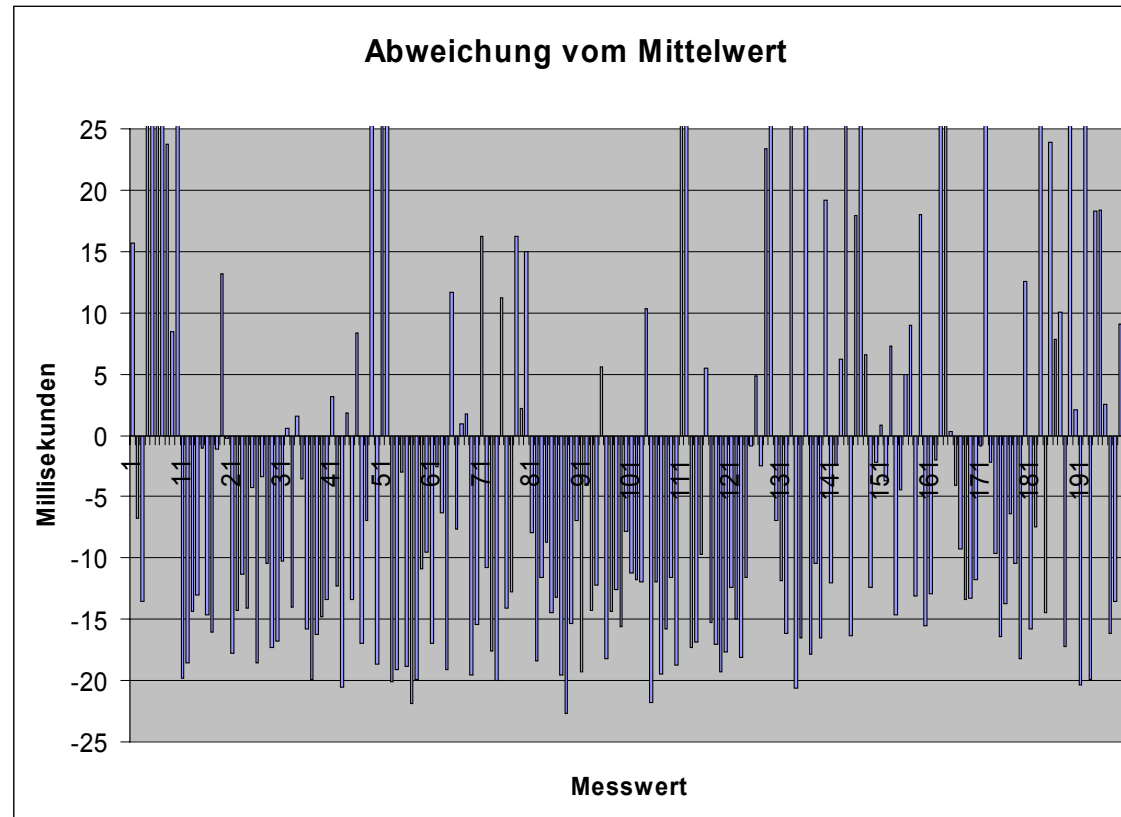
■ Wahrscheinlichkeit: Zeitverhalten

- Verzögerungen über einen Router/Firewall (Intranet zu DMZ)



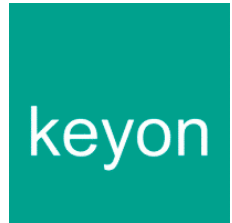
■ Wahrscheinlichkeit: Zeitverhalten

- Verzögerungen über 10 Router (Internet)





Wahrscheinlichkeit



■ **Wahrscheinlichkeit: Man-In-The-Middle**

- Wenn ein Angreifer "Man-In-The-Middle" sein kann, dann hat er folgende, einfachere Angriffs-Möglichkeiten:
 - Server simulieren und somit Passworte direkt empfangen
 - Belauschen jeglichen Verkehrs



Wahrscheinlichkeit

keyon

■ **Wahrscheinlichkeit: CBC Modus, Identische Verbindungen**

- 80% der SSL Verbindungen werden Standardmässig mit dem Algorithmus "RC4-MD5" betrieben
(Serverbetreiber oder User muss "RC4-MD5" explizit deaktivieren, dass CBC zum Einsatz kommt)
- Benutzer erhält zum Beispiel beim Outlook IMAP Client dauernd Fehlermeldungen. Ob der Benutzer 2000 mal versucht Mails zu holen mit diesen Fehlermeldungen?



Protokoll



■ IMAP Trace

0000 OK CAPABILITY completed

0001 LOGIN "hmeier" "MY.,PASS"

0001 OK User hmeier authenticated

0002 IDLE

0002 OK IDLE completed

| | | | | | |
|------|-------|--------------|--------------|----|----|
| XXXX | LOGIN | „<username>“ | „<password>“ | CR | LF |
|------|-------|--------------|--------------|----|----|



Protokoll

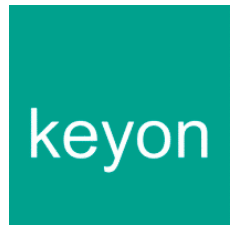
keyon

■ HTTP Trace

```
POST http://www.sicherheitstest.ch/cgi-bin/cookieTest HTTP/1.0
Accept: image/gif, image/x-bitmap, image/jpeg, image/pjpeg
, application/vnd.ms-excel, application/msword, application/vnd.ms-
powerpoint, */*
Referer: http://www.sicherheitstest.ch/browser/cookieTest.html
Accept-Language: de-ch
Content-Type: application/x-www-form-urlencoded
Proxy-Connection: Keep-Alive
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0; .NET CLR)
Host: www.sicherheitstest.ch
Content-Length: 46
Pragma: no-cache
Cookie: sessionId=245423534534509435
```



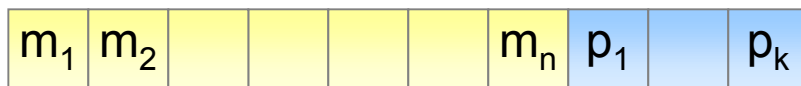
Agenda



- Begrüssung
- Demo EPFL Schwachstelle
- Grundlagen
- Voraussetzungen fürs Ausnützen
- **Erklärungen zu CBC und private Key Attacke**
- Verhinderung der Attacken
- Fragen/Antworten/Diskussion

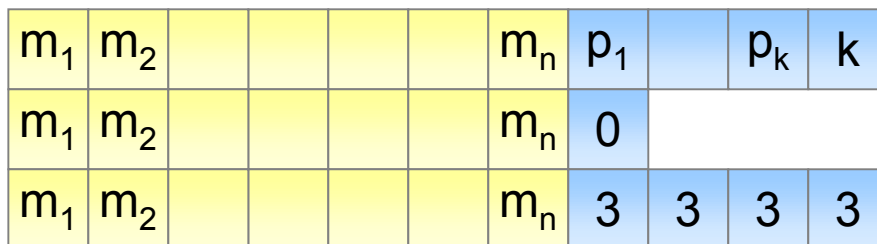
■ Padding

- Block-Cipher (DES, IDEA, 3-DES, RC5)
- Blocklänge (z.B. 8 Byte)



- $n + k = \text{Vielfaches von Blocklänge}$

■ SSL Padding

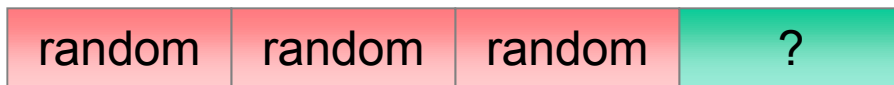


■ CBC Attacke

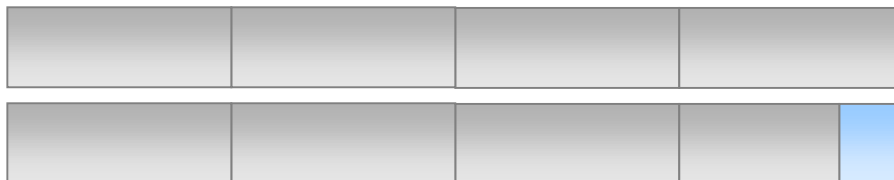
- Original Cipher Text



- Manipulierter Cipher Text



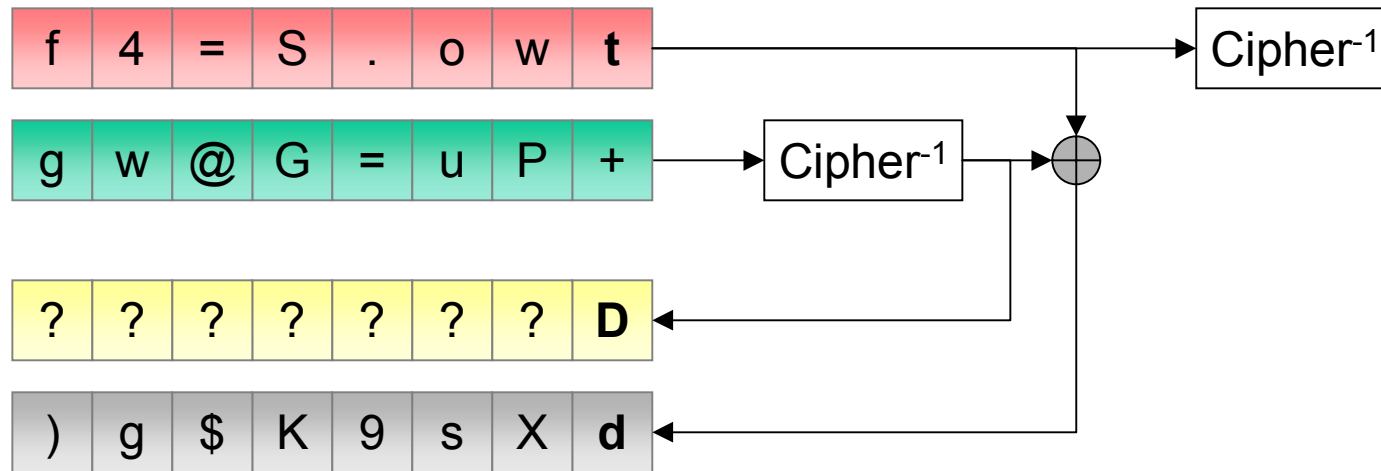
- Manipulierter Klartext



⇒ decryption failed alert

⇒ bad record mac alert

■ Last Word Oracle

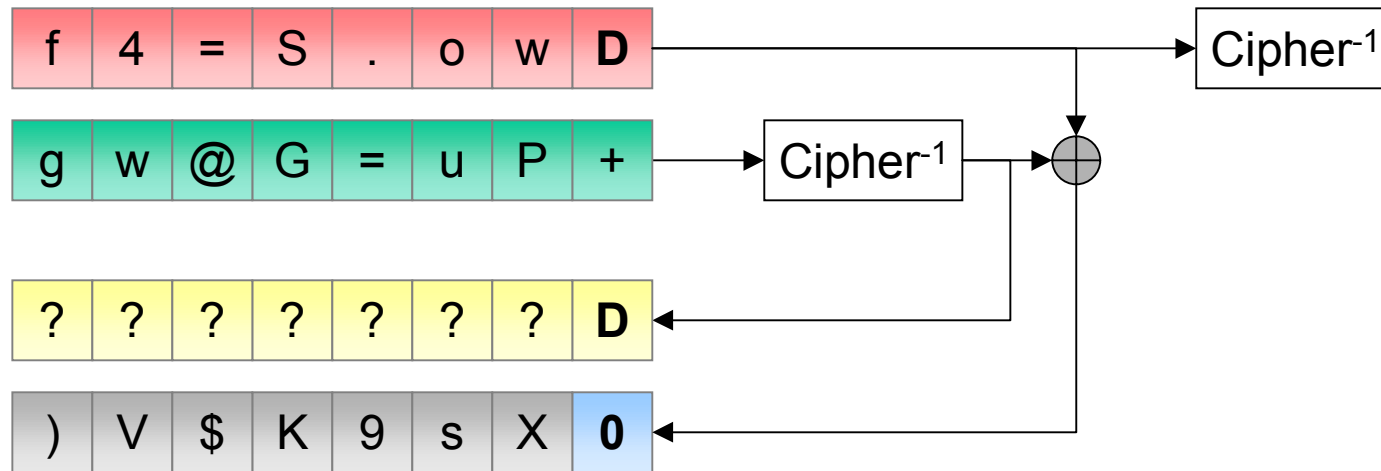


decryption failed alert

$$t \text{ XOR } D = d$$

$$w = \frac{255}{256}$$

■ Last Word Oracle

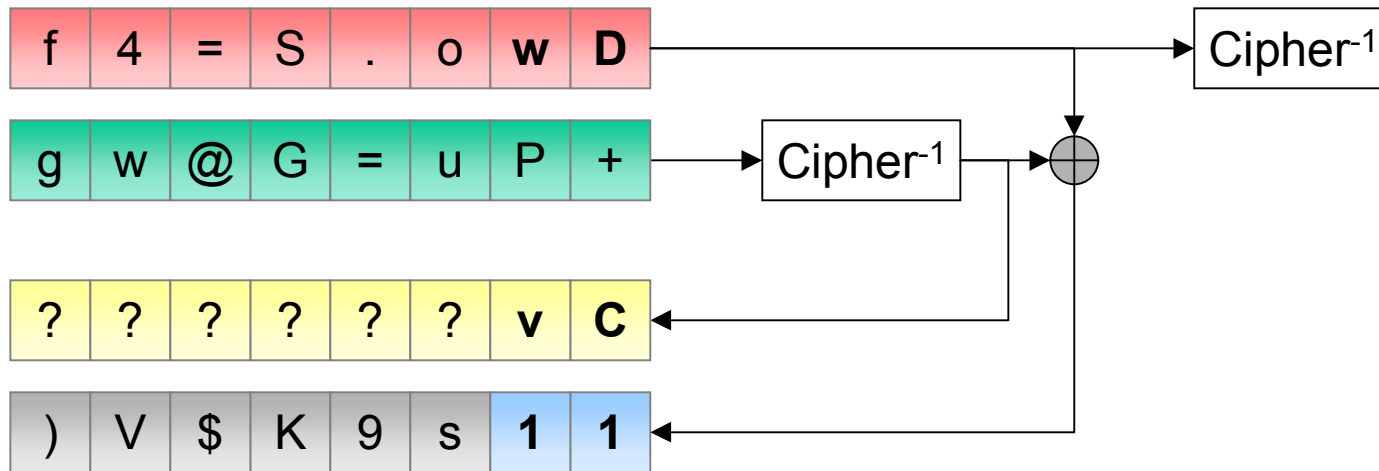


bad record mac alert

$$D \text{ XOR } D = 0$$

$$w = \frac{1}{256}$$

■ Last Word Oracle



bad record mac alert

$$D \text{ XOR } C = 1$$

$$w \text{ XOR } v = 1$$

$$w = \frac{1}{2^{16}}$$



CBC Attacke

keyon

■ Problem

- Alerts sind verschlüsselt
- decryption failed alert
- bad record mac alert

■ Lösung

- Timing Attacke
- Antwortzeiten der Alerts sind unterschiedlich
- $t(\text{bad record mac alert}) > t(\text{decryption failed alert})$
- Unterschied: $\sim 1 \text{ ms}$

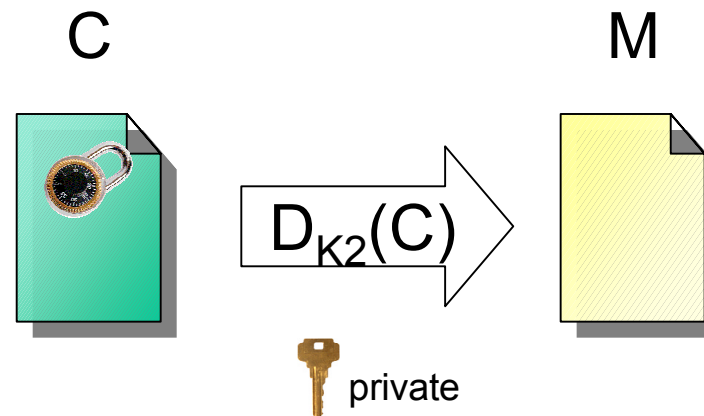
⇒ **Attacke anfällig auf Störungen**

- **Block Decryption Oracle**
 - Entschlüsseln von Blöcken
- **Decryption Oracle**
 - Entschlüsseln von ganzen Meldungen
- **Aufwand**
 - Blocklänge 8 Byte (DES, 3-DES, IDEA, RC5)
 - N Blöcke
 - Aufwand = $1024 \cdot N$ Oracle Aufrufe

⇒ **Attacke ist sehr effizient**

■ Timing Attacke auf RSA Implementation

- RSA decryption, RSA signing
- bisher für Hardware Security Tokens und Smartcards



■ OpenSSL RSA Implementation

- $m = c^d \bmod N$, $N = p \cdot q$
- Chinese Remainder Theorem (CRT)
 - $m_1 = c^{d_1} \bmod p$ (d1 vorausberechnet aus d)
 - $m_2 = c^{d_2} \bmod q$ (d2 vorausberechnet aus d)
 - m_1 kombiniert (nach CRT) mit m_2 ergibt m
- Timing Attacke auf CRT

⇒ **Attacke auf p und q**
⇒ **Faktorisierung von N**
⇒ **Brechen des privaten Schlüssels**

■ Vereinfachung

- Schritt 1 und 2 verwenden den selben Code
- $g^d \bmod q$

■ square and multiply

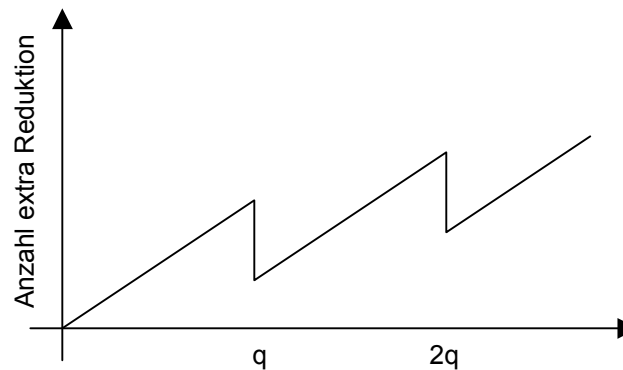
- $x = g^{1011} \bmod q$
 - $x = 1$
 - $x = x^2 \cdot g \bmod q$
 - $x = x^2 \bmod q$
 - $x = x^2 \cdot g \bmod q$
 - $x = x^2 \cdot g \bmod q$
- quadrieren: $\log_2(d)$
- multiplizieren: $\log_2(d) / 2$

■ Montgomery Reduktion

- $z = x \cdot y \bmod q$
 - $z' = x \cdot y$
 - $z = z' \bmod q$
- Variablen in Montgomery Form bringen
- Montgomery Reduktion durchführen
- Letzter Schritt
 - Output $> q$?
 - ja: extra Reduktion
 - nein: keine extra Reduktion

■ Montgomery Reduktion

- Wahrscheinlichkeit für eine extra Reduktion:
 - steigt je näher g an q kommt.
 - Fällt schlagartig bei $g \bmod q = 0$.



⇒ extra Reduktion
⇒ Timing Attacke 1

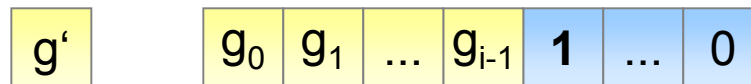
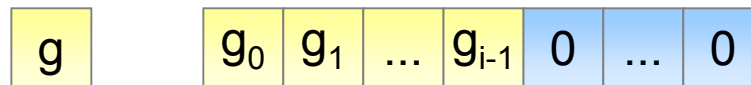
■ Multiplication Routines (OpenSSL)

- $z = x \cdot y$
- $m = \text{words}(x), n = \text{words}(y)$
- Karatsuba
 - $m = n$
 - $\text{Zeit} = O(n^{1.58})$
- normal
 - $m \neq n$
 - $\text{Zeit} = O(n \cdot m)$

⇒ Multiplication Routine Seleciton
⇒ Timing Attacke 2

■ Timing Attacke gegen OpenSSL

- 1024 Bit RSA Private Key
- Schätzung g für q : $2^{512} > g > 2^{511}$
- Timing für Decryption für Kombination der 3 höchsten Bits
 \Rightarrow 2 Peaks für p und q
 \Rightarrow 1. Peak für q ($q < p$)
- Finden des i -ten Bits von q



- $|\text{decTime}(g) - \text{decTime}(g')|$
 - gross \Rightarrow i -tes Bit ist 0
 - klein \Rightarrow i -tes Bit ist 1

■ Timing Attacke gegen OpenSSL

- $\text{gross} \Rightarrow i\text{-tes Bit ist } 0$
- $\text{decTime}(g) - \text{decTime}(g') > 0 \Rightarrow \text{Extra Reduction}$
- $\text{decTime}(g) - \text{decTime}(g') < 0 \Rightarrow \text{Multiplication Routine}$

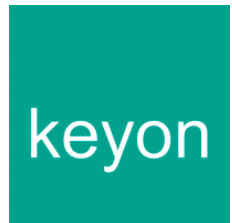
■ Coppersmith's Algorithm

- Nach der Hälfte der Most Significant Bits
- Faktorisierung

$\Rightarrow N$ faktorisiert
 \Rightarrow RSA Private Key gebrochen



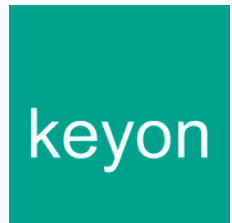
Agenda



- Begrüssung
- Demo EPFL Schwachstelle
- Grundlagen
- Voraussetzungen fürs Ausnützen
- Erklärungen zu CBC und private Key Attacke
- **Verhinderung der Attacken**
- Fragen/Antworten/Diskussion



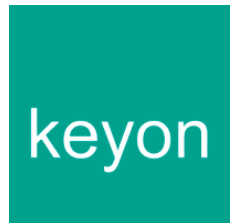
Verhinderung



- **Falls OpenSSL eingesetzt wird:**
 - Folgende Produkte setzen z.B. auf OpenSSL
 - Apache/mod_ssl und Varianten davon
 - sTunnel
 - sslwrap
 - Auf aktuelle Version updaten (Version 0.9.7b)



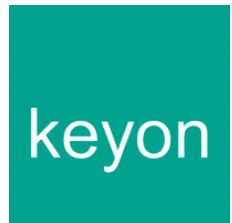
Verhinderung



- **Falls nicht OpenSSL eingesetzt wird:**
 - Bestätigung vom Hersteller verlangen, dass:
 - die 3 Schwachstellen in seinem Produkt getestet wurden respektive ein Source Code Review stattgefunden hat
 - die Schwachstellen entweder nicht vorhanden sind oder ein Patch bzw. eine neue Version lieferbar ist
 - Bestätigungsfragen an Hersteller:
 - http://www.csnc.ch/downloads/docs/lectures/ssl_statement.pdf



Verhinderung

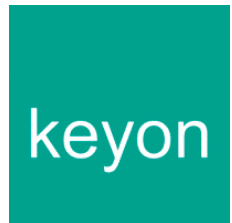


■ CBC Padding Attacke

- deaktivieren des CBC Modus, bis Patch eingespielt ist
- überwachen der Logdateien auf folgende Einträge
 - "decryption failed"
 - "bad mac decode"



Verhinderung

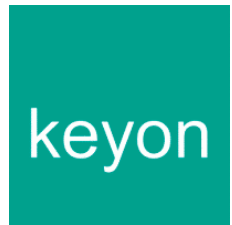


■ RSA Private Key Attacke

- RSA Blinding einschalten
 - Ist per Default AUS-geschaltet
- Mit dem Update (OpenSSL 0.9.7b) ist RSA Blinding per Default EIN-geschaltet
- RSA Blinding bei den meisten Hardware Security Modulen implementiert



Agenda



- Begrüssung
- Demo EPFL Schwachstelle
- Grundlagen
- Voraussetzungen fürs Ausnützen
- Erklärungen zu CBC und private Key Attacke
- Verhinderung der Attacken
- **Fragen/Antworten/Diskussion**