Cryptography in Theory and Praxis: The Case of Encryption in IPsec

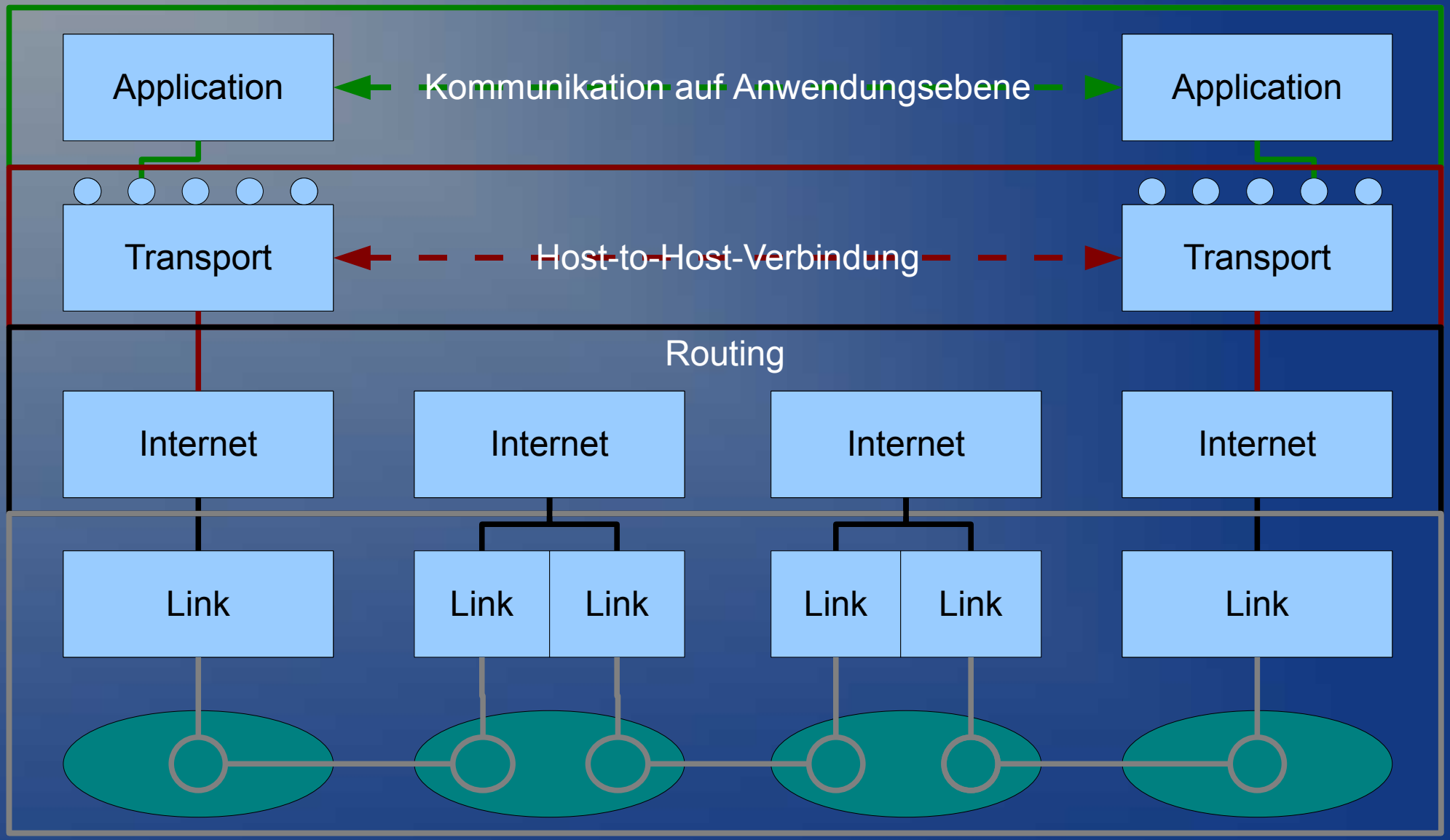
André Dau
nach einem Paper von Paterson und Yau

Studienstiftung des deutschen Volkes
Nizza - Sommerakademie 2009
Arbeitsgruppe 4 - Applied Cryptography and Security Engineering
September 2009

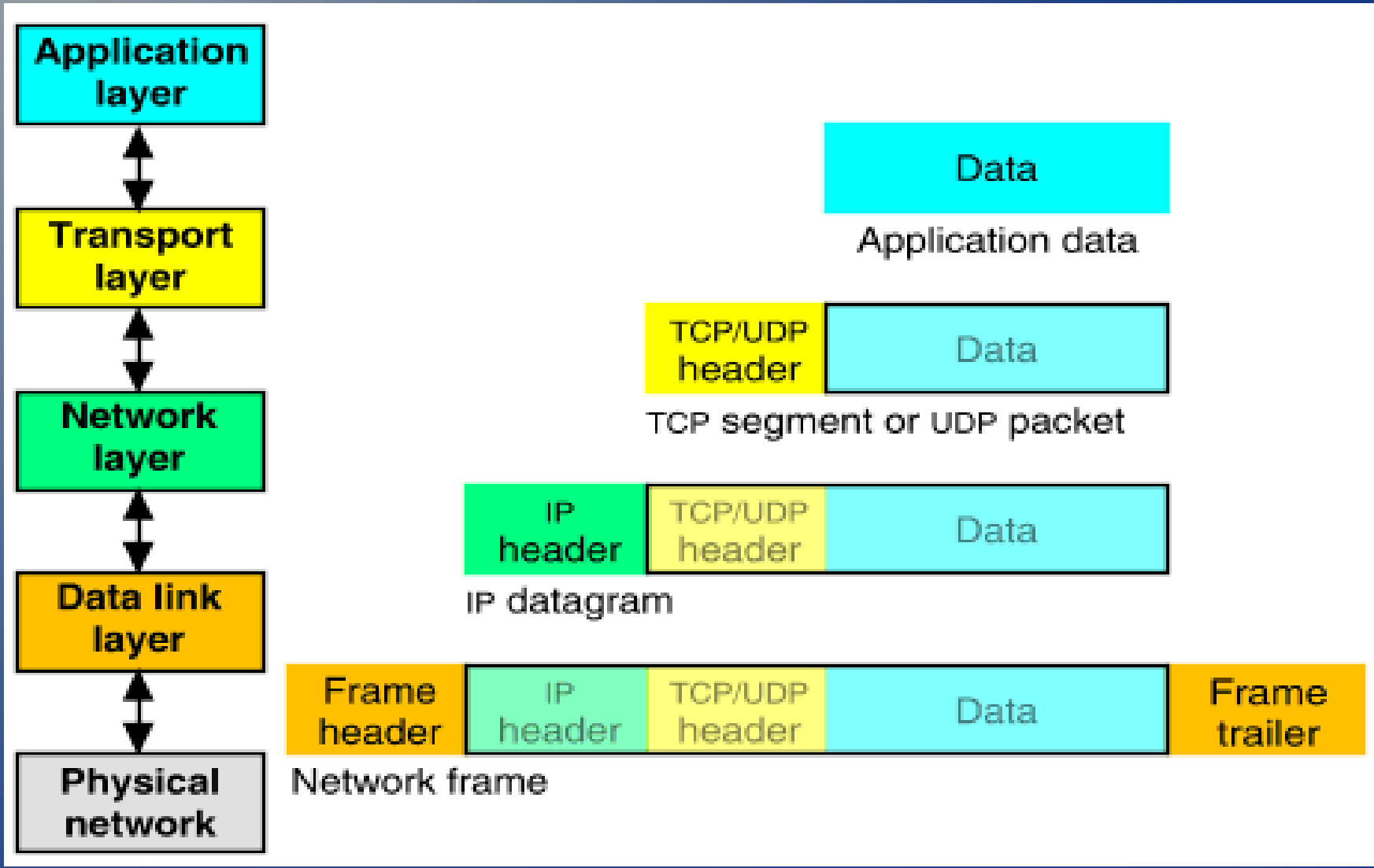
Inhaltsverzeichnis

Inhaltsverzeichnis

1. TCP/IP-Modell

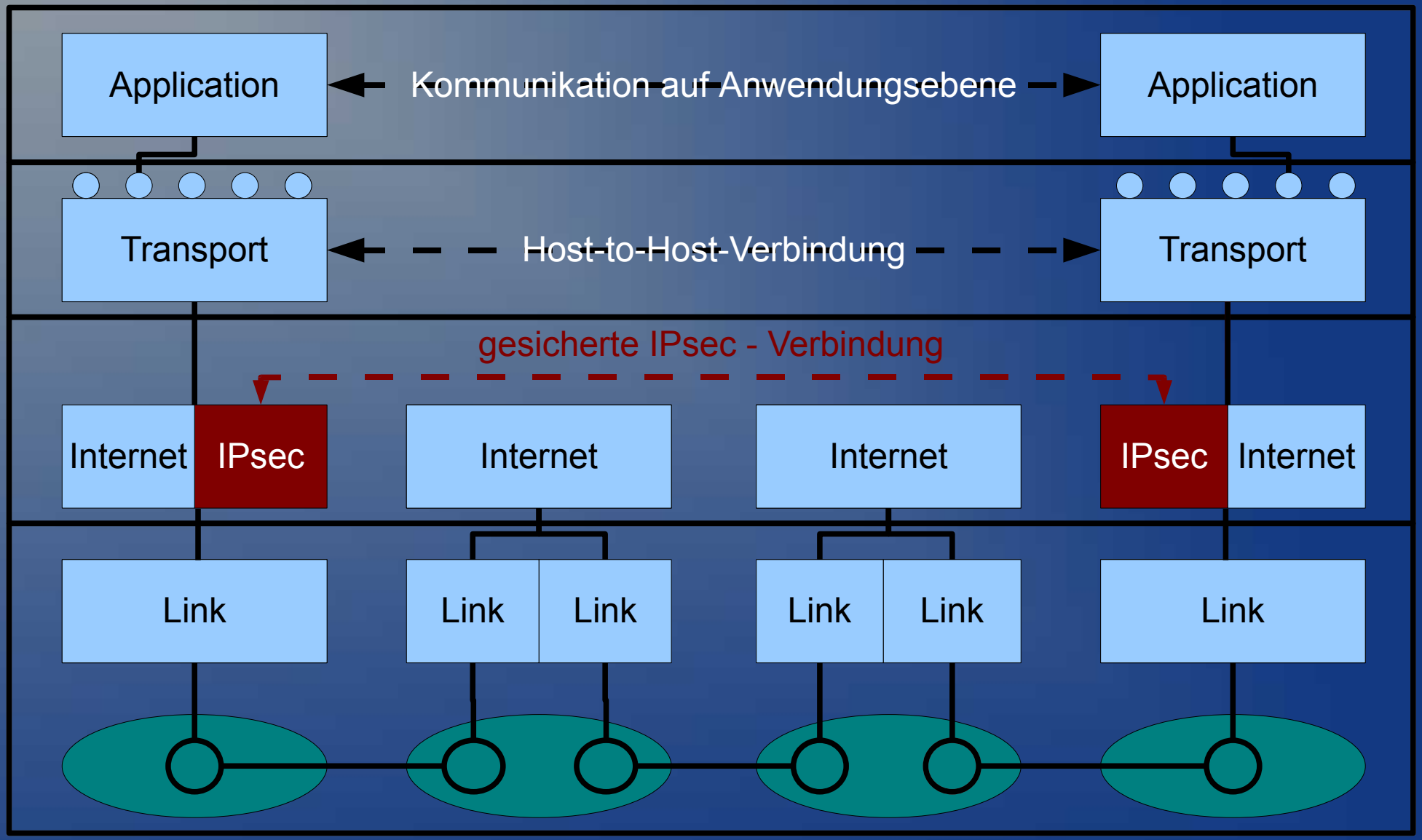


1. TCP/IP-Modell



http://uw713doc.sco.com/en/NET_tcpip/graphics/encapsulation.gif

2. IPsec

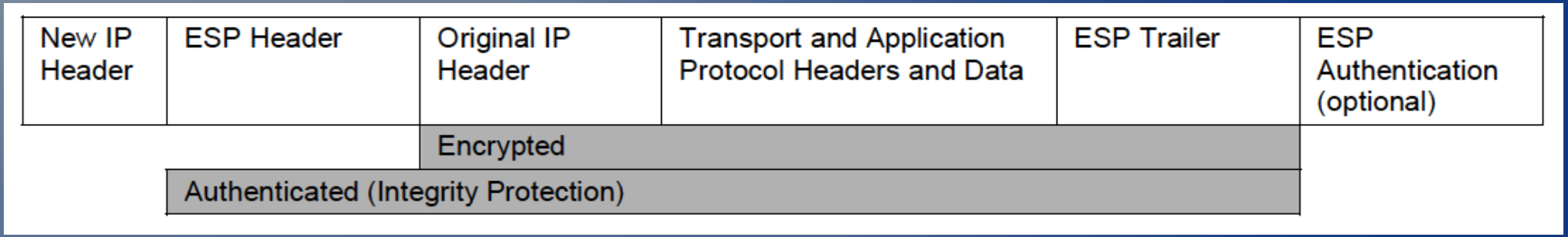

Application
Kommunikation auf Anwendungsebene
Application
Transport
Host-to-Host-Verbindung
Transport
gesicherte IPsec - Verbindung
Internet
IPsec
Internet
Internet
IPsec
Internet
Link
Link
Link
Link
Link
Link

2. IPsec

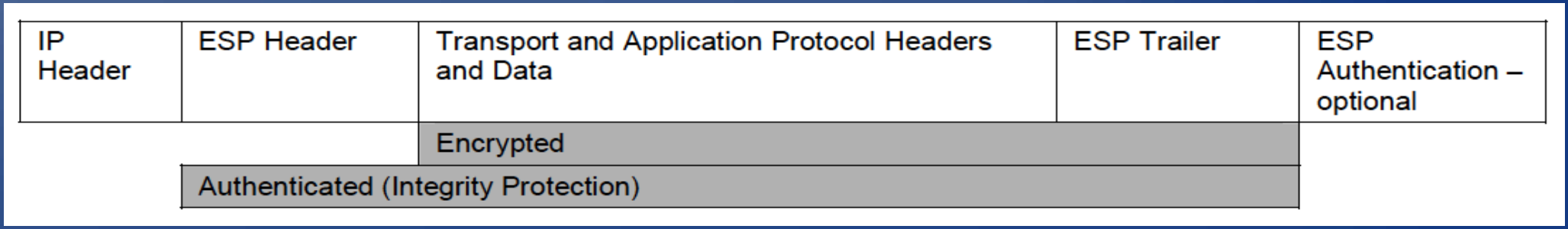
- Betrachtete Versionen: IPSec v2, IETF RFCs 2401-2412, November 1998; IPSec v3, IETF RFCs 4301-4309, December 2005
- Internet Key Exchange (IKE)
  - Aushandeln und Verwalten von Schlüsseln und Sicherheitsparametern
- Authentication Header (AH)
  - Authentifizierung und Integritätsschutz
- Encapsulating Security Paylod (ESP)
  - Vertraulichkeitsschutz und optional Authentifizierung und Integritätsschutz
  - Kein Schutz des äußersten IP-Headers

2.1 ESP

- Tunnel mode



- Transport mode



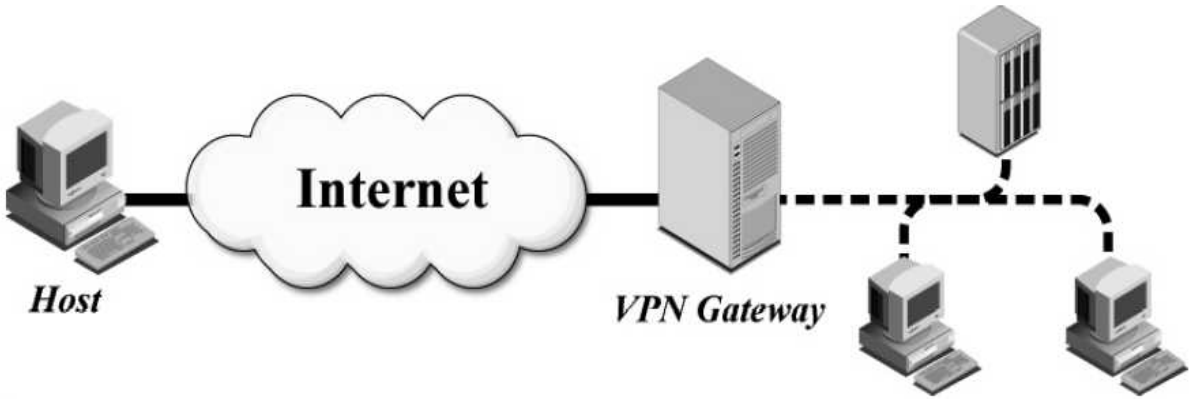
2.2 IPsec policies

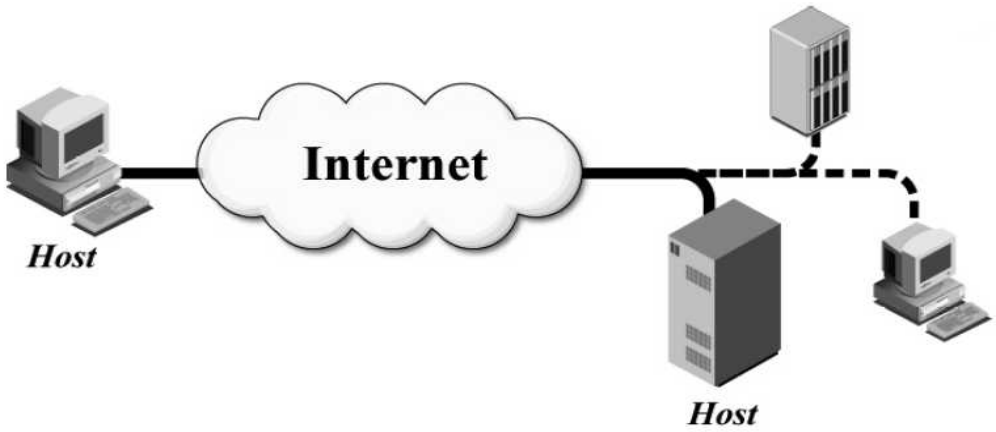
- Spezifizieren auf der Ebene der Transportschicht, welche IP-Pakete wie behandelt und gesichert werden sollen
- Sollten beim Senden UND Empfangen eines IP-Paketes überprüft werden
- Linux führt keine Prüfung bei empfangenen Paketen durch → Vorraussetzung für die Angriffe von Paterson und Yau

3. Virtual Private Networking

Gateway-to-Gateway

Host-to-Gateway





Host-to-Host

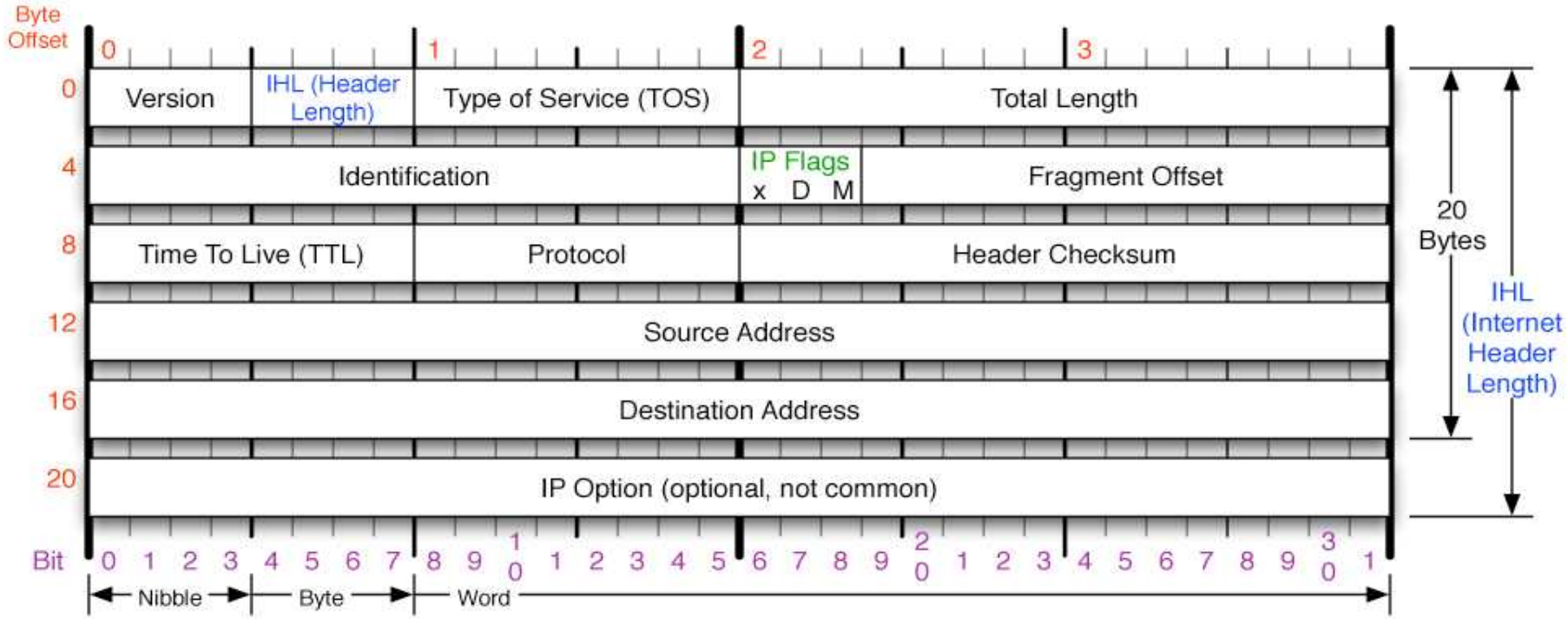
4. Cipher block chaining (CBC) mode

- Aufteilen der Daten in Blöcke P_1 bis P_n
- Blockweise Verschlüsselung der Daten
- C_i = encrypt(secret key , $P_i \oplus C_{i-1}$)
- C_0 ist ein beliebig wählbarer Initialisierungsvektor (IV)
- Beobachtung: Eine Änderung in P_i bewirkt eine Änderung aller Chiffretextblöcke ab C_i

5. Bit flipping attack for CBC mode

- Anwendung beim CBC mode
- Ziel: Gezieltes Verändern abgefangener verschlüsselter Pakete ohne die Kenntnis des Schlüssels
- Invertieren von einzelnen Bits im entschlüsselten Block *i* durch Invertieren der entsprechenden Bits in C_{i-1}
- Dadurch wird jedoch der entschlüsselte Block *i-1* randomisiert
- Alle anderen Blöcke bleiben unbeeinflusst

IP Header (version 4)



Version

Version of IP Protocol. 4 and 6 are valid. This diagram represents version 4 structure only.

Header Length

Number of 32-bit words in TCP header, minimum value of 5. Multiply by 4 to get byte count.

Protocol

IP Protocol ID. Including (but not limited to):

1 ICMP	17 UDP	57 SKIP
2 IGMP	47 GRE	88 EIGRP
6 TCP	50 ESP	89 OSPF
9 IGRP	51 AH	115 L2TP

Total Length

Total length of IP datagram, or IP fragment if fragmented. Measured in Bytes.

Fragment Offset

Fragment offset from start of IP datagram. Measured in 8 byte (2 words, 64 bits) increments. If IP datagram is fragmented, fragment size (Total Length) must be a multiple of 8 bytes.

Header Checksum

Checksum of entire IP header

IP Flags

x D M

x 0x80 reserved (evil bit)
D 0x40 Do Not Fragment
M 0x20 More Fragments follow

RFC 791

Please refer to RFC 791 for the complete Internet Protocol (IP) Specification.

http://www.visi.com/~mjb/Drawings/IP_Header.png

7. Abarbeitung des IP-Headers unter Linux

- Wird von JEDER Zwischenstelle ausgeführt
- 1. Überprüfen von Version und IHL
- 2. Überprüfen der Header Checksumme
- 3. Überprüfen der tatsächlichen und angegebenen Paketlänge
- Falls ein Fehler in 1.-3. auftritt, verwirf das Paket OHNE Fehlermeldung an den Sender
- Falls IHL größer 5 ist, sind Optionen im Header enthalten

7. Abarbeitung des IP-Headers unter Linux

- 4. Prüfe falls vorhanden Optionen
  - Bei einem Fehler (z.B. Formatierung) benachrichtige den Sender über den fehlerhaften Header (ICMP)
- 5a. Falls das Paket weitergeleitet werden muss, prüfe ob TTL größer 0 ist
  - Wenn nein, verwirf das Paket OHNE Benachrichtigung
  - Sonst, verringere TTL um eins und leite es weiter

7. Abarbeitung des IP-Headers unter Linux

- 5b. Falls das Paket für diesen Host bestimmt ist, prüfe das Protokollfeld, um den nächst höheren Dienst zu bestimmen
  - Falls das angegebene Protokoll nicht unterstützt wird, benachrichtige den Sender über den Fehler (ICMP)

8. Voraussetzungen für die Angriffe

- Betriebssystem: Linux Kernelversion 2.6.8.1
- Nach dem Entschlüsseln eines empfangenen Paketes wird die Einhaltung der IPsec policies nicht geprüft (gegeben mit Linuxkernel 2.6.8.1)
- ESP wird im encryption-only Tunnelmodus (kein Integritätsschutz) verwendet
- Verschlüsselungsverfahren wird im CBC-Modus mit 64 oder 128 Bit Blocklänge verwendet
- Der ESP-Trailer enthält eine minimale Anzahl Padding-Bytes
- Es wird IPv4 verwendet

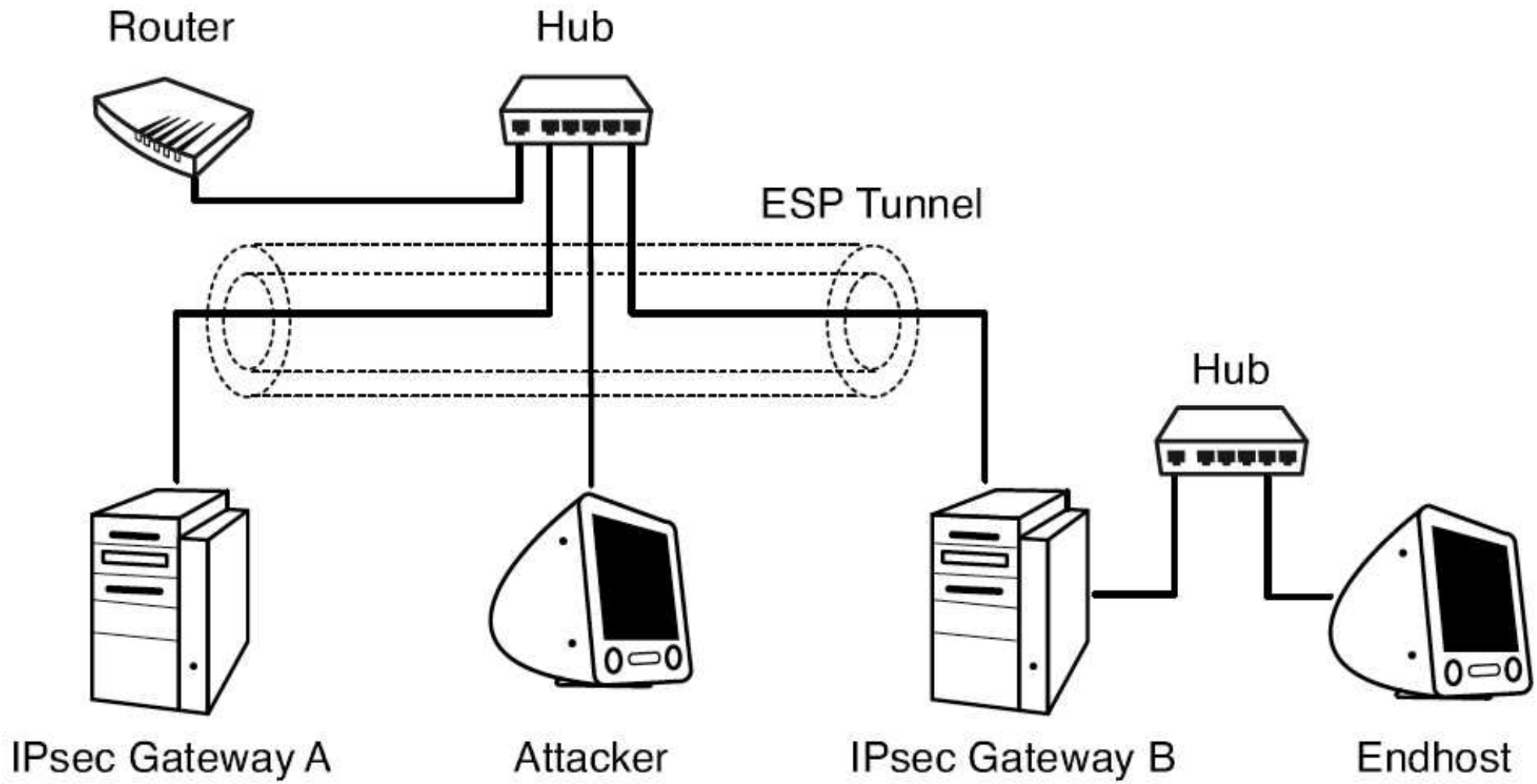
9. Grundidee aller Angriffe

- Fange einzelne verschlüsselte Pakete ab
- Nutze die Art und Weise aus, wie IP-Pakete vom Empfänger behandelt werden und

  verändere gezielt den inneren verschlüsselten IP-Header mit Hilfe von bit flipping, um

  den Empfänger dazu zu bringen, Teile des Klartextes an den Angreifer weiterzuleiten
- Der Angreifer führt KEINE Kryptoanalyse durch, sondern lässt sich den Klartext liefern

10. Versuchsaufbau



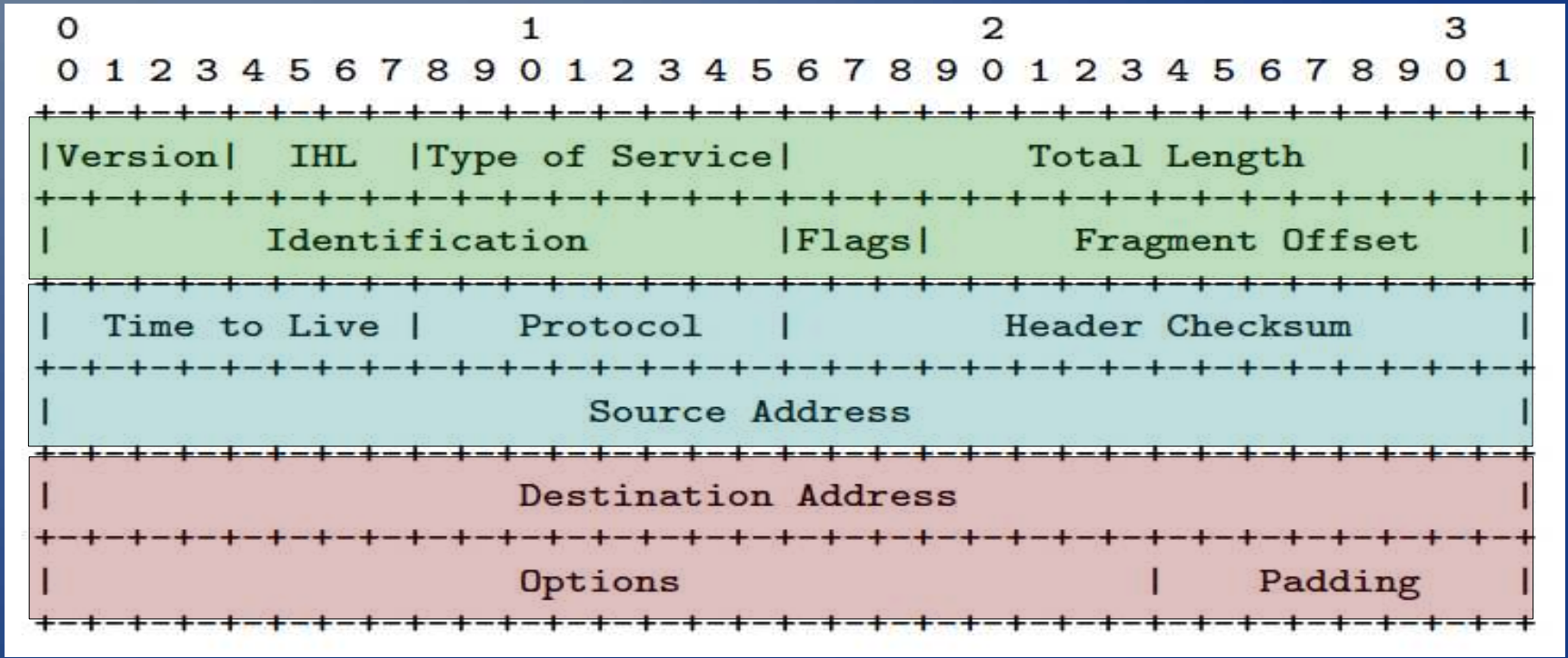
11. Erster Angriff: Überschreiben der Zieladresse

- Fall: 64 Bit – Verschlüsselung
- Angreifer muss große Teile der Zieladresse kennen

C1

C2

C3

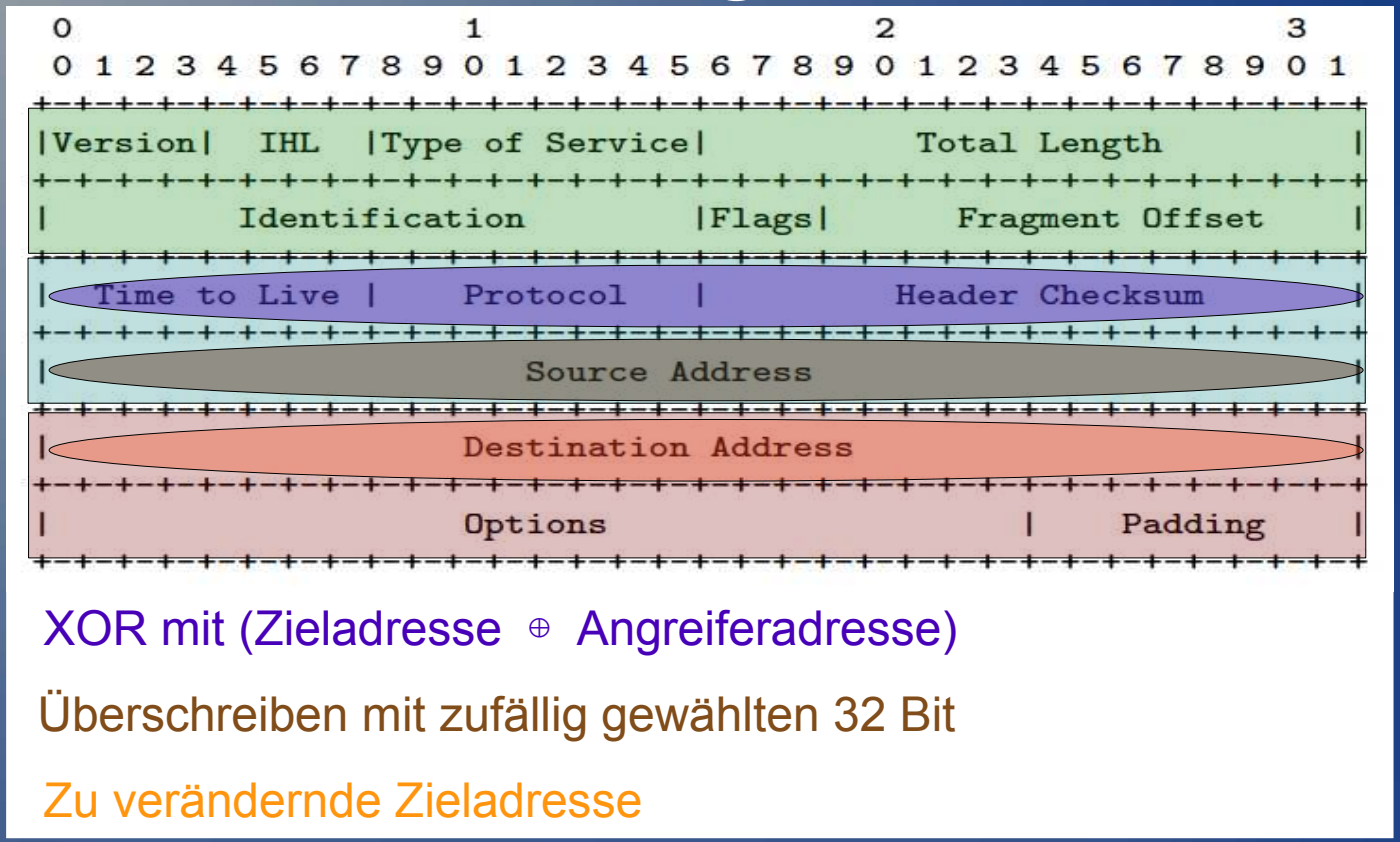


11. Erster Angriff: Idee

- Nutze das Wissen über die Zieladresse im inneren verschlüsselten Paket und überschreibe die Zieladresse mit Hilfe von bit flipping
- Gateway entschlüsselt das innere Paket und routet es zum Rechner des Angreifers

11. Erster Angriff: Phase eins

C1
C2
C3



- Wähle so lange zufällige Werte für die letzten 32 Bit von C2, bis TTL groß genug und die Checksumme korrekt ist
- Falls die Zieladresse nicht vollständig bekannt ist, unbekannte Stellen durchprobieren

11. Erster Angriff: Phase zwei

- Wiederverwenden des in Phase 1 ermittelten Headers und Trailers, um restliche Pakete zu entschlüsseln
- Verwende C_0, C_1, C_{2b}, C_3 und C_{q-2}, C_{q-1}, C_q und füge dazwischen *q-6* beliebige aufeinanderfolgende codierte Blöcke ein
- Der erste Block nach C_3 (sei dies D_j) muss nach dem Entschlüsseln durch XOR mit $C_3 \oplus D_{j-1}$ wiederhergestellt werden

11. Erster Angriff: Effizienz

- Die Wahrscheinlichkeit bei vollständiger Kenntnis der Zieladresse einen erfolgreichen Header zu erzeugen ist ungefähr 2^{-17}
- Mit ca. *60%-iger* Wahrscheinlichkeit ist man damit nach 2^{17} Iterationen erfolgreich
- Paterson und Yau benötigten mit ihrer Beispielimplementation für den 128-Bit Fall ca. 2^{15} Iterationen (im Paper keine Informationen für den 64-Bit-Fall vorhanden)

12. ICMP - Nachrichten

- Werden bei bestimmten Fehlern in Folge der Bearbeitung von IP-Paketen generiert und an den Sender des Paketes geschickt (z.B. Fehler in den IP-Optionen oder nicht unterstütztes Transportschicht-Protokoll)
- Enthalten Informationen zum Fehler, den gesamten IP-Header und Teile des Payloads
- Linux erlaubt eine maximale ICMP-Nachrichtengröße von 576 Byte

  → relativ viel Payload in der ICMP-Nachricht enthalten

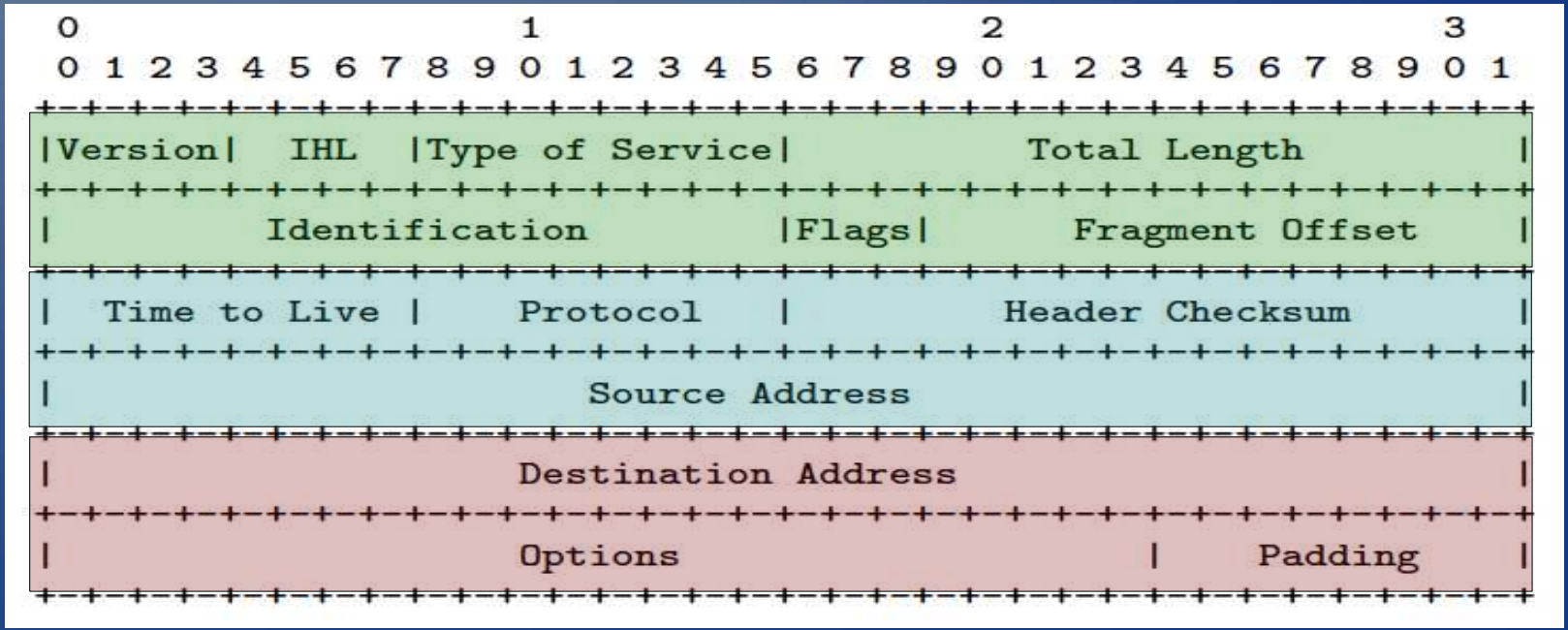
13. Zweiter Angriff: IP-Header-Optionen-Manipulation

- Fall: 64 Bit – Verschlüsselung
- Angreifer muss ICMP-Nachrichten des Gateways, die nicht durch den Tunnel gehen, abfangen können

C1

C2

C3

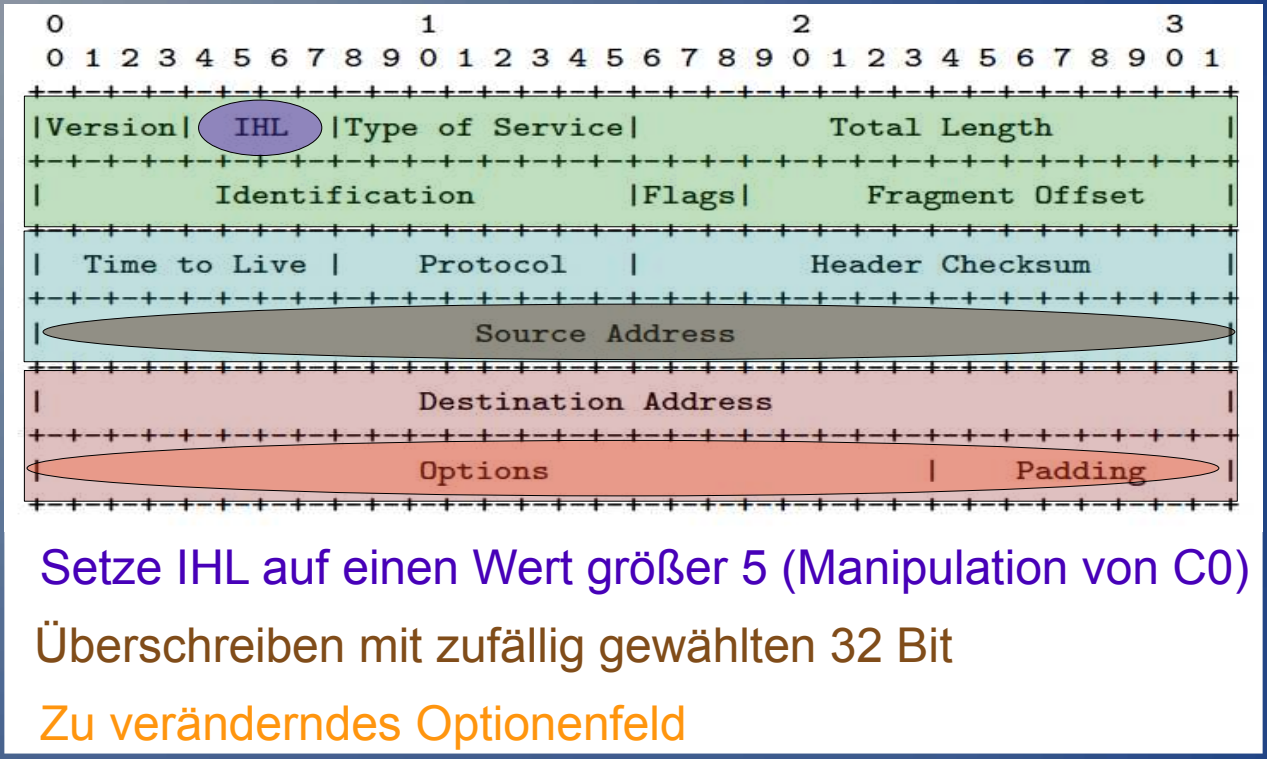


13. Zweiter Angriff: Idee

- Setze den IHL-Wert auf einen Wert größer 5, damit Teile des Payloads als Optionenfelder interpretiert werden
- Probiere verschiedene Werte für das Optionenfeld, bis eine „parameter problem“ ICMP-Nachricht vom Gateway erzeugt wird
- Ändere dabei die Quelladresse, sodass ICMP-Nachrichten nicht durch den Tunnel geleitet werden
- Fange die unverschlüsselten ICMP-Nachrichten ab, die Teile des entschlüsselten IP-Paketes enthalten

13. Zweiter Angriff: Phase eins

C1
C2
C3



- Setze die letzten 32-Bit von C_2 auf einen zufälligen Wert bis eine ICMP-Nachricht empfangen wird
  - TTL, Checksumme und Quelladresse müssen passende Werte haben
  - Das Optionenfeld muss ein „parameter problem“ auslösen

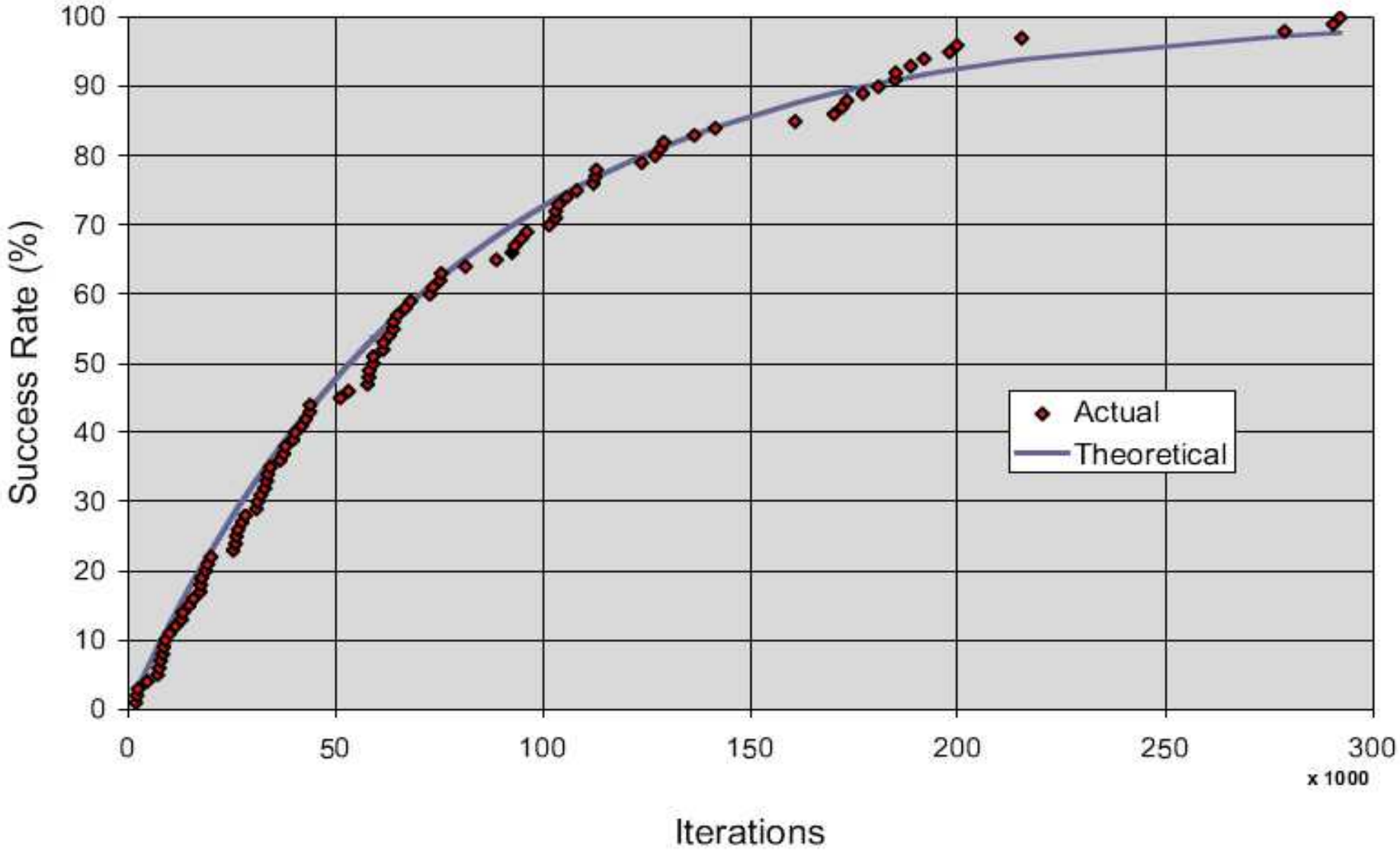
11. Zweiter Angriff: Phase zwei

- Verwende, wie beim ersten Angriff, einen erfolgreichen Header wieder
- Ersetze jeweils je nach erlaubtem ICMP-Payload die vorderen Blöcke
- Da möglicherweise nur Teile des IP-Paketes in der ICMP-Nachricht enthalten sind, benötigt man eventuell mehrere ICMP-Nachrichten für ein IP-Paket

13. Zweiter Angriff: Effizienz

- *85%-ige* Wkt., dass bei zufälliger Wahl der Quelladresse und der Optionen eine ICMP-Nachricht generiert wird
- Wkt. für korrekte Checksumme ca. 2^{-16}
- → Wkt. für Erfolg nach *t* Iterationen ca. $1 – (1 – 0.85 \times 2^{-16})^t$
- Testclient benötigte im Schnitt ca. 77600 Iterationen (bei ihrem Aufbau ca. 2.5 min)
- Limitierung durch maximale ICMP-Generierungsrate des Gateways und maximalen Payload der ICMP-Nachricht

13. Zweiter Angriff: Effizienz

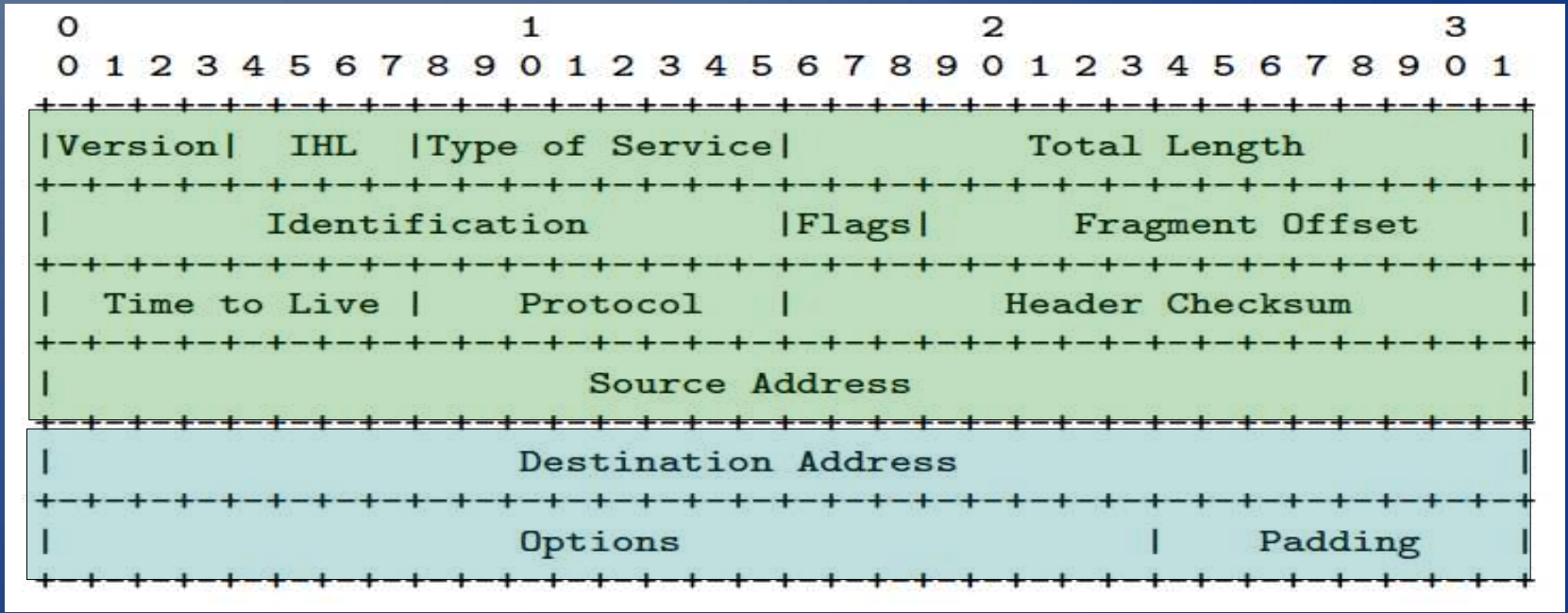


14. Dritter Angriff: Protokollfeld-Manipulation

- Fall: 128 Bit – Verschlüsselung
- Angreifer muss ICMP-Nachrichten des Gateways, die nicht durch den Tunnel gehen, abfangen können

C1

C2.1



14. Dritter Angriff: Idee

- Manipuliere das Protokollfeld des Headers, um eine „protocol unreachable“ ICMP-Nachricht zu provozieren
- Verändere die Quelladresse, damit die ICMP-Nachricht nicht durch den Tunnel geht
- Da das Protokollfeld erst beim endgültigen Empfänger geprüft wird, stammt die ICMP-Nachricht diesmal vom Empfänger und nicht vom Gateway

14. Dritter Angriff: Phase eins

- Manipuliere den IV (C_0) so, dass sowohl das Protokollfeld als auch die Quelladresse genau in einem Bit verändert werden
- Die Bits müssen so gewählt werden, dass die neue Protokoll-ID mit Sicherheit nicht vom Empfänger unterstützt wird und die Quelladresse geroutet werden kann und nicht hinter dem Tunnel liegt
- Korrigiere die Checksumme durch weitere Manipulation des IV so lange, bis eine ICMP-Nachricht empfangen wird

14. Dritter Angriff: Phase eins

- Bei einer Änderung von genau einem Bit in genau einem der 16-Bit-Blöcke lässt sich die Checksumme durch ein XOR mit einer von 17 möglichen Masken korrigieren
- Die Wkt. für die Masken sind geometrisch verteilt

Mask	Probability
0000000000000001	1/2
0000000000000011	1/4
0000000000000111	1/8
⋮	⋮
1111111111111111	2^{-16}
1111111111111110	2^{-16}

Table 1. Table T_{15} of masks and their probabilities for bit position 15.

14. Dritter Angriff: Phase eins

- Eine Änderung von zwei Bits entspricht einem XOR mit zwei Masken
- Die Checksumme lässt sich also nach Veränderung von zwei Bits durch eine der 17 x 17 = 289 möglichen Maskenkombinationen korrigieren
- Probiert man alle Kombinationen beginnend mit denen mit der höchsten Wahrscheinlichkeit durch, benötigt man im Mittel weniger als sieben Iterationen, bis man die korrekte Checksumme erhält

14. Dritter Angriff: Phase zwei

- Wie Phase zwei des 2. Angriffs
- Wiederverwenden des erfolgreichen Headers

14. Dritter Angriff: Effizienz

- Testclient benötigte im Schnitt 6.53 Iterationen (1.34 Sekunden mit kurzen Pausen zwischen den Iterationen)

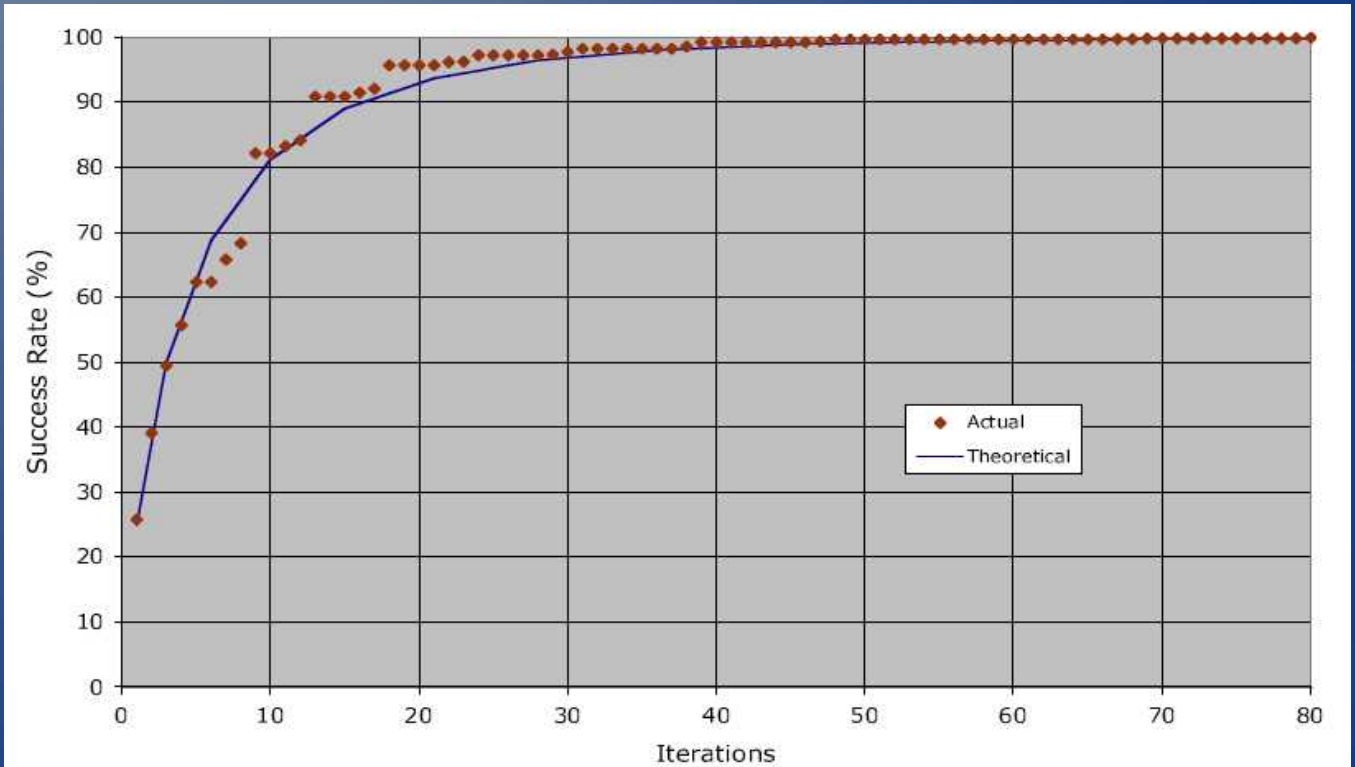


Fig. 9. Performance of 128-bit attack based on manipulation of protocol field.

15. Fazit

- Angriff funktioniert unabhängig vom genauen Verschlüsselungsalgorithmus
- ESP im encryption-only-Modus ist nicht nur in der Theorie, sondern auch in der Praxis unsicher
  - → ESP sollte NIE im encryption-only Modus verwendet werden!
- Zusätzliche Authentifizierung muss an den **richtigen** Stellen stattfinden

16. Gegenmaßnahmen

- Zusätzlich Integritätsschutz und Authentifizierung verwenden
  - Wichtig: Authentifizierung muss auch an den Gateways stattfinden
- Beste Lösung: Authenticated Encryption
  - Alternativ: Encryption + Message Authentication Code
- IPsec policy checks auch nach dem Entschlüsseln von empfangen Paketen
- Server hinsichtlich der Generierung von ICMP-Nachrichten einschränken

Quellen

- 1. K. Paterson & A. Yau, „Cryptography in Theory and Practice: The Case of Encryption in IPsec“, Eurocrypt, 2006
- 2. S. Frankel, K. Kent, R. Lewkowski, A. Orebaugh, R. Ritchey, S. Sharma, „Guide to IPsec VPNs“, Recommendations of the National Institute of Standards and Technology, Special Publication 800-77, December 2005
- 3. Wikipedia.org:
  - http://en.wikipedia.org/wiki/IPv4
  - http://en.wikipedia.org/wiki/Cipher_block_chaining#Cipher-block_chaining_.28CBC.2
  - http://en.wikipedia.org/wiki/Internet_Control_Message_Protocol