Inhalt

[Übungsszenarien 1](#_Toc131606756)

[Grundlegende Begriffe 1](#_Toc131606757)

[Protokoll Zusatzinformation 1](#_Toc131606758)

[Ablauf des Protokolls (Normaler Ablauf) 2](#_Toc131606759)

[Ablauf des Protokolls (Lauschangriff Ablauf) 5](#_Toc131606760)

[Ablauf des Protokolls (Man in the Middle Ablauf) 8](#_Toc131606761)

# Übungsszenarien

* Nutzer kann zwischen einzelnen Phasen, der Kombination aus mehreren (aufeinanderfolgenden) Phasen oder dem kompletten Ablauf auswählen
* Nutzer kann eine/mehrere/alle Rolle(n) auswählen
* Nutzer kann einen Schwierigkeitsgrad auswählen
* Nutzer wählt zwischen den Szenen aus

|  |  |  |
| --- | --- | --- |
| Szene | | |
| Normaler Ablauf | Sniffing | MITM |

# Grundlegende Begriffe

* rektilinear (**+**) vs. diagonaler (**x**) Filter
* (+) und (x) sind **Polarisationsschema (Polsch)**
* (+x++ x+x+) sind **Polarisationsschemata (Polscha)**
* (0) und (1) sind **Bit**
* (0011 0110) ist ein **Bitstrom**

# Protokoll Zusatzinformation

* Wahrscheinlichkeit, dass Bob die Bits richtig interpretiert: 75 %
* Wahrscheinlichkeit, dass Bob den richtigen Filter verwendet: 50 %
* Wahrscheinlichkeit, dass Eve nicht erwischt wird:

# Ablauf des Protokolls (Normaler Ablauf)

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Alice** | **Eve** | **Bob** |
| *Phase 0* | *Nachricht, die gesendet werden soll, bestimmen* |  |  |
| *Bestimme Schlüssellänge anhand der zu Verschlüsselnden Nachricht (3-Fache Länge der Nachricht)* |  |  |
| *Sendet die Schlüssellänge an Bob* | *🡪*  *🡨* | *Empfängt Schlüssellänge* |
| Phase 1  (initiale Kodierung) | Erzeugt Folge aus „0“ und „1“  Bsp.: 0101 1011 |  |  |
| Erzeugt die gleiche Anzahl an Polarisationsschemata    Bsp.: x++x +x+x |  |  |
| Bitfolge wird anhand von Polarisationsschemata in Photonen kodiert und gesendet  Bsp.: \ | -- / | \ | / | 🡪 | Wählt seine eigenes Polarisationsschemata aus  Bsp.: ++xx xx++ |
|  |  | Empfängt Photonen, die über den Übertragungskanal übertragen wurden |
| Phase 2 | Teilt unverschlüsselt Polarisationsschemata mit  Bsp.: x++x +x+x | 🡪 |  |
|  |  | Vergleicht mit seinem gewählten Polarisationsschemata und streicht die Stellen, bei denen er ein falsches Polarisationsschema ausgewählt hat |
|  |  | *Streicht die Stellen, bei denen er das falsche Polarisationsschema benutzt hat (0 hier falschen Detektor vewendet)*  *110011*  *Bsp.:* 011001 -> 0101 |
|  | 🡨 | Teilt Alice mit, an welchen Stellen er falsche Polarisationsschemata verwendet hat  Bsp.: 110011 |
| Streicht die Stellen, bei denen Bob das falsche Polarisationsschema benutzt hat  Bsp.: 011001 -> 0101 |  |  |
| Phase 3  (Wurde-abgehört?-Phase) | Frägt nach bestimmten Bits („0“ oder „1“ und Position) |  |  |
|  | 🡨 | Bob sendet die angeforderten Bits und streicht diese aus seinem Schlüssel |
| Alice prüft die Bits auf Gleichheit |  |  |
| Wenn alle Bits mit ihren übereinstimmen, wurde die Übertragung nicht abgehört und Alice streicht die geprüften Bits |  |  |
| Phase 4 | Nachricht auf tatsächliche Schlüssellänge Padden |  |  |
|  | Nachricht mit Schlüssel verschlüsseln |  |  |
|  | Nachricht senden | 🡪 | Nachrichten empfangen |
|  |  |  | Nachricht entschlüsseln |
|  |  |  | Padding entfernen |

**Mögliche Handlungsmöglichkeiten**

* Nachricht, die gesendet werden soll, bestimmen
* Nachricht, in Bits konvertieren
* Nachrichten länge bestimmen
* Schlüssellänge angeben (Eventuell feste Länge vorgeben (Nachrichten in Blöcke einteilen vgl. Blockchiffren)2,5 bis 3 fache der Nachrichten länge nachrechnen welche Wahrscheinlichkeit das Schlüssel zu kurz wird)
* Schlüssellänge senden
* Zufallszahl erzeugen (Schlüssellänge)
* Polscha gleicher Länge erzeugen (Zufallszahl mit Schlüssellänge)
* Photonen kodieren und senden
* Photonen empfangen und auslesen
* Empfangsbestätigung senden
* Polscha Teilen
* L == R ? (Bitfolgen übereinstimmende stellen ist auch nur ein negiertes XOR)
* Bitfolge übereinstimmen bestimmen (Stellen, wo das Schema übereinstimmt, bestimmen)
* Filtern einer Bitfolge nur nach (Löschen der Stellen die in der Bitfolge == 0)
* Fragen nach X Bits anbestimmten Stellen (Bitfolge senden)
* Bitfolge senden (Angeforderte Bits senden)
* Bestätigung senden, dass Bits übereinstimmt haben
* Padden der Nachricht auf Schlüssellänge
* XOR (Nachricht + Schlüssel)
* Bitfolge senden
* Bit to int

# Ablauf des Protokolls (Lauschangriff Ablauf)

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Alice** | **Eve** | **Bob** |
| *Phase 0* | *Nachricht, die gesendet werden soll, bestimmen* |  |  |
| *Bestimme Schlüssellänge anhand der zu Verschlüsselnden Nachricht (3-Fache Länge der Nachricht)* |  |  |
| *Sendet die Schlüssellänge an Bob* | *Eve hört Schlüssellänge ab*  *🡪* |  |
|  | *🡨* | *Bestätigt die Schlüssellänge (gleiche Zahl als Bestätigung)* |
| Phase 1  (initiale Kodierung) | Erzeugt Folge aus „0“ und „1“  Bsp.: 0101 1011 |  |  |
| Erzeugt die gleiche Anzahl an Polarisationsschemata    Bsp.: x++x +x+x |  |  |
| Bitfolge wird anhand von Polarisationsschemata in Photonen kodiert und gesendet  Bsp.: \ | -- / | \ | / | Wählt seine eigenes Polarisationsschemata aus  Bsp.: ++xx xx++  🡪 | Wählt seine eigenes Polarisationsschemata aus  Bsp.: ++xx xx++ |
|  | Empfängt Photonen, die über den Übertragungskanal übertragen wurden  und verändert dabei die Polarisation einiger Photonen  🡪 |  |
|  |  |  | Empfängt die veränderten Photonen, die über den Übertragungskanal übertragen wurden |
| Phase 2 | Teilt unverschlüsselt Polarisationsschemata mit  Bsp.: x++x +x+x | 🡪 |  |
|  |  | Vergleicht mit seinem gewählten Polarisationsschemata und streicht die Stellen, bei denen er ein falsches Polarisationsschema ausgewählt hat |
|  |  | *Streicht die Stellen, bei denen er das falsche Polarisationsschema benutzt hat (0 hier falschen Detektor vewendet)*  *110011*  *Bsp.:* 011001 -> 0101 |
|  | Hört Übereinstimmungen des Polarisationschemata ab  🡨 | Teilt Alice mit, an welchen Stellen er falsche Polarisationsschemata verwendet hat  Bsp.: 110011 |
| Streicht die Stellen, bei denen Bob das falsche Polarisationsschema benutzt hat  Bsp.: 011001 -> 0101 | Stricht die Stellen, bei denen Bob das falsche Polarisationsschema benutzt hat  Bsp.: 011001 -> 0101 |  |
| Phase 3  (Wurde-abgehört?-Phase) | Frägt nach bestimmten Bits („0“ oder „1“ und Position) | Eve hört die Nachricht ab und streicht die abgefragten Stellen |  |
|  | 🡨 | Bob sendet die angeforderten Bits und streicht diese aus seinem Schlüssel |
| Alice prüft die Bits auf Gleichheit |  |  |
| Im Normalfall wird festgestellt, dass die Bits nicht übereinstimmen und die Kommunikation wird, abgebrochen (Senden des Abbruchs an Bob) |  |  |
| Phase 4 | Nachricht auf tatsächliche Schlüssellänge Padden |  |  |
| Nachricht mit Schlüssel verschlüsseln |  |  |
| Nachricht senden | 🡪 | Nachrichten empfangen |
|  |  | Nachricht entschlüsseln |
|  |  | Padding entfernen |

**Neue mögliche Handlungsmöglichkeiten:**

* Schlüssellänge abhören

# Ablauf des Protokolls (Man in the Middle Ablauf)

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Alice** | **Eve** | **Bob** |
| *Phase 0* | *Nachricht, die gesendet werden soll, bestimmen* |  |  |
| *Bestimme Schlüssellänge anhand der zu Verschlüsselnden Nachricht (3-Fache Länge der Nachricht)* |  |  |
| *Sendet die Schlüssellänge an Bob*  *🡪* | *Eve hört Schlüssellänge ab* |  |
|  | *Eve ändert ggf. Schlüssellänge ab und sendet diese an Bob*  *🡪* |  |
|  |  | *Empfängt Eves Schlüssellänge* |
| Phase 1  (initiale Kodierung) | Erzeugt Folge aus „0“ und „1“  (Schlüssellänge Alice)  Bsp.: 0101 1011 |  |  |
| Erzeugt die gleiche Anzahl an Polarisationsschemata    Bsp.: x++x +x+x |  |  |
| Bitfolge wird anhand von Polarisationsschemata in Photonen kodiert und gesendet  Bsp.: \ | -- / | \ | /  🡪 | Wählt seine eigenes Polarisationsschemata aus  (Schlüssellänge Alice)  Bsp.: ++xx xx++ |  |
|  | Empfängt Photonen, die über den Übertragungskanal übertragen wurden  (Alice Photonen) |  |
|  | Wertet Photonen mit dem Polarisationsschemata aus  Bsp.: 01010101 |  |
|  | Erzeugt Folge aus (Schlüssellänge Eve) „0“ und „1“  Bsp.: 0101 1011 |  |
|  | Erzeugt die gleiche Anzahl an Polarisationsschemata    Bsp.: x++x +x+x |  |
|  | Bitfolge wird anhand von Polarisationsschemata in Photonen kodiert und gesendet an Bob  Bsp.: \ | -- / | \ | /  🡪 |  |
|  |  | Wählt seine eigenes Polarisationsschemata aus  (Schlüssellänge Eve)  Bsp.: ++xx xx++ |
|  |  | Empfängt die veränderten Photonen, die über den Übertragungskanal übertragen wurden  (Photonen von Eve) |
|  |  |  | Wertet Photonen mit dem Polarisationsschemata aus  Bsp.: 01010101 |
| Phase 2 | Teilt unverschlüsselt Polarisationsschemata mit  Bsp.: x++x +x+x  🡪 | Fängt  Polarisationsschemata ab  (Alice Polscha)  Bsp.: x++x+x+x |  |
|  | Vergleicht mit (Alice Eve) gewählten Polarisationsschemata und streicht die Stellen, bei denen sie ein falsches Polarisationsschema ausgewählt hat |  |
|  | *Streicht die Stellen, bei denen sie das falsche Polarisationsschema benutzt hat (0 hier falschen Detektor vewendet)*  *110011*  *Bsp.:* 011001 -> 0101 |  |
|  | Teilt Alice mit, an welchen Stellen sie falsche Polarisationsschemata verwendet hat  Bsp.: 110011  *🡨* |  |
|  | Teilt unverschlüsselt Polarisationsschemata mit  (Eve Polscha)  Bsp.: x++x +x+x  🡪 |  |
|  |  | Empfängt  Polarisationsschemata  (Eve Polscha)  Bsp.: x++x+x+x |
|  |  | Vergleicht mit seinem gewählten Polarisationsschemata und streicht die Stellen, bei denen er ein falsches Polarisationsschema ausgewählt hat |
|  |  | *Streicht die Stellen, bei denen er das falsche Polarisationsschema benutzt hat (0 hier falschen Detektor vewendet)*  *110011*  *Bsp.:* 011001 -> 0101 |
|  | Streicht die Stellen, bei denen Bob das falsche Polarisationsschema benutzt hat  *110011*  *Bsp.:* 011001 -> 0101 | Teilt mit, an welchen Stellen er falsche Polarisationsschemata verwendet hat  Bsp.: 110011  🡨 |
| Streicht die Stellen, bei denen Eve das falsche Polarisationsschema benutzt hat  Bsp.: *110011*  011001 -> 0101 |  |  |
| Phase 3  (Wurde-abgehört? -Phase) | Frägt nach bestimmten Bits („0“ oder „1“ und Position)  🡪 | Eve sendet die angeforderten Bits und streicht diese aus ihrem Schlüssel  🡨 |  |
|  | Frägt nach bestimmten Bits („0“ oder „1“ und Position)  🡪 |  |
|  | 🡨 | Bob sendet die angeforderten Bits und streicht diese aus seinem Schlüssel |
| Alice prüft die Bits auf Gleichheit | Eve prüft die Bits von Bob auf Gleichheit |  |
|  | Wenn alle Bits mit ihren übereinstimmen, wurde die Übertragung (Zwischen Eve und Bob) nicht abgehört und Alice streicht die geprüften Bits  (Eve könnte auch hier schon die Chiffre für Bob erzeugen) |  |
| Wenn alle Bits mit ihren übereinstimmen, wurde die Übertragung (Zwischen Alice und Eve) nicht abgehört und Alice streicht die geprüften Bits |  |  |
| Phase 4 | Nachricht auf tatsächliche Schlüssellänge Padden |  |  |
| Nachricht mit Schlüssel verschlüsseln |  |  |
| Chiffre senden  🡪 | Chiffre empfangen |  |
|  | Chiffre mit OTP (Alice Eve Schlüssel) entschlüsseln |  |
|  | Padding entfernen |  |
|  | Nachricht für Bob auf tatsächliche Schlüssellänge Padden |  |
|  | Chiffre senden  🡪 |  |
|  |  | Chiffre empfangen |
|  |  | Chiffre mit OTP (Eve Bob Schlüssel) entschlüsseln |
|  |  | Padding entfernen |