Alexander Becker (684303)

Aufgabe 1.1 Diffie-Hellman Schlüsselaustausch

1. Ein möglicher aktiver Angriff auf das Diffie-Hellman Protokoll wäre eine MitM-Attacke. Dieser nutzt die Schwäche aus, dass die Codierung der Nachricht vom anfänglichen Austausch der Geheimnisse abhängt. Wenn dieser Austausch abgefangen wird und dementsprechend Nachrichten vom Angreifer an beiden Parteien gesendet werden, hat der Angreifer dann einen Schlüssel für die Kommunikation zu Alice und zu Bob. Er kann dann Nachrichten mitlesen und selber Nachrichten verfassen. Damit bricht der Angreifer Confidentiality, Authenticity und Integrity. Verhindern könnte man diesen Angriff dadurch, dass man einen Message Authentication Code benutzt oder das beide Parteien signieren und verifizieren können. Dies würde aber eine Trusted Third Party vorraussetzen, sodass beide Parteien vorher sicher einen public key erhalten oder sicher einen secret key (MAC) ausmachen

   Ein möglicher passiver Angriff wäre zum Beispiel eine einmalige Kompromittierung von Alice oder Bob. Was bedeutet, dass der Angreifer dann selbst den Diffie-Hellman Schlüssel rekonstruieren kann, wenn die beiden Geheimnisse A und B, welche zum Beispiel für nur eine vergangene Session verwendet werden, nicht gelöscht werden, was indem Fall eine Schwachstelle ist. Dies könnte einfach verhindert werden, wenn die Geheimnisse gelöscht werden, sodass der Schlüssel nicht rekonstruiert werden kann.

2. Gegenüber dem normalen Diffie-Hellman werden beim *static Diffie-Hellman* g^A und g^B zu public keys, welche dann für beide Parteien über TTP erreichbar sein müssen. Der initiale Schlüsselaustausch über einen möglichen unsicheren Kanal wird dann durch so eine Instanz ersetzt. Darüber wird dann die Authenticity erfüllt.

   Bei dem *ephemeral Diffie-Hellman* werden die ausgetauschten Schlüssel als Session-Schlüssel verwendet und die Geheimnisse A und B werden gelöscht, was bedeutet, dass nach der Session der Schlüssel nicht mehr restaurierbar ist.

   Generell stellt *ephemeral Diffie-Hellman* Forward Secrecy sicher, sodass, falls eine Kompromittierung nach einer fertigen Kommunikation stattfindet, keine Nachrichten aus der Zukunft mit den gefundenen Wissen entschlüsselt werden können. *Static Diffie-Hellman* stellt dies nicht sicher, da nach einer Kompromittierung einer Partei der erhaltene Schlüssel für spätere Nachrichten noch zum Verschlüsseln verwendet werden kann, sofern die Kompromittierung nicht aufgefallen ist und ein neuer Schlüssel ausgetauscht wurde. Auf der anderen Seite ermöglicht der *ephemeral Diffie-Hellman* nicht die Authenticity, da dieser zum Beispiel durch eine mögliche MitM-Attacke, nicht sicher authentifizieren kann, ob die Nachricht nicht mitgelesen, verändert wurde oder komplett von jemand anderem kommt.

Aufgabe 1.2 Analyse eines Schlüsselaustausch-Protokolls

1. Nach dem Ablauf des Schlüsselprotokolls sollen Alice und Bob den selben Schlüssel ausgeben, wobei Alice k ausgibt und Bob $w \oplus t$ final ausgibt. Diesen können dann gleichgesetzt und vereinfacht werden.

$$
\begin{aligned}
k &= w \oplus t \\
&= u \oplus r \oplus t \\
&= s \oplus t \oplus r \oplus t \\
&= k \oplus r \oplus t \oplus r \oplus t \\
&= k \oplus (r \oplus r) \oplus (t \oplus t) \\
&= k
\end{aligned}
$$

   Dementsprechend geben Alice und Bob den selben Schlüssel aus.

2. Ein möglicher Angriff wäre, dass der Angreifer den gesamten gesendeten Verkehr mitschneidet. Wenn er die Information hat, wie das Protokoll funktioniert, kann er sich alle unbekannten Parameter erschließen. Initial schneidet er die Parameter s, u und w mit.

$$\begin{aligned} s &= k \oplus r \\ u &= k \oplus r \oplus t \\ w &= k \oplus r \oplus t \oplus r \\ &= k \oplus t \end{aligned}$$

Dann muss k nur im letzten Schritt berechnet werden:

$$\begin{aligned} w \oplus t &= k \oplus t \oplus t \\ &= k \end{aligned}$$

Aufgabe 1.3 Secrecy

1. Generell unterscheiden sich die beiden Eigenschaften darin, ob verschlüsselte Nachrichten aus der Vergangenheit oder noch zu erzeugende Nachrichten aus der Zukunft nach einer Kompromittierung einer beliebigen Partei entschlüsselt werden können. Genauer gesagt, soll für die Vergangenheit oder Zukunft die Sicherheit von Nachrichten geschützt werden. Bei der Erfüllung *forward secrecy* sollen also, ausgehend von einer temporären Kompromittierung des aktuellen Schlüssel zum Beispiel, alle zukünftigen Nachrichten der Partei aber trotzdem sicher sein. Bei der Erfüllung von *post-compromise security* sollen also, ausgehend von einer temporären Kompromittierung, alle vergangenen Nachrichten der Partei aber trotzdem sicher sein.
2. Wenn *forward secrecy* nicht erfüllt ist, kann also, wenn zum Beispiel ein Schlüssel kompromittiert wurde, der Angreifer dann alle zukünftigen Nachrichten von der Partei entschlüsseln und mitlesen. Wenn *post-compromise security* nicht erfüllt ist, kann also, wenn zum Beispiel ein Schlüssel kompromittiert wurde, der Angreifer dann alle vergangenen Nachricht von der Partei entschlüsseln und mitlesen.
3. Wenn ein Schlüsselaustausch-Protokoll beide Eigenschaften erfüllen soll, muss sichergestellt sein, dass wenn ein Schlüssel kompromittiert wurde, dieser nicht genutzt werden kann um Nachrichten aus der Vergangenheit oder in der möglichen Zukunft zu entschlüsseln. Trivialerweise kann dieser zum Beispiel nicht dafür benutzt werden, weil für jede neue Kommunikationsrunde ein neuer Schlüssel erzeugt wird.

Aufgabe 1.4 Extended Triple Diffie-Hellman Protocol

1. Im gesamte initialen Prozess führt Alice drei oder vier Diffie-Hellman Schlüsselaustausch-Operationen durch. Dann kann aus den drei ausgetauschten Schlüsseln ein geteilter symmetrischer Schlüssel sk abgeleitet werden, welcher dann als Schlüssel für die Kommunikation genommen werden kann. DH_1 und DH_2 sind jeweils dazu da um die Authentification sicherzustellen. DH_3 und DH_4 auf der anderen Seite werde für die Erreichung von Forward Secrecy verwendet.
2. a) Eve kann bei dieser Variante immer noch einen Man in the Middle Angriff durchführen und damit initial den Schlüsselaustausch abfangen und ihren Schlüssel weiterleiten. Da der Counter nicht verschlüsselt ist, kann sie dementsprechend die Schlüssel auch hashen. Durch die statischen Identity Keys kann Eve diese auch durch die Nachrichten erhalten und kann alles mitlesen oder etwas selber verfassen, da sie sich immer mit dem richtigen Identity Key ausweisen kann.

   b) Generell ist diese Variante auch nicht sicher. Alice und Bob wissen sie nicht wie der Identity Key der anderen Partei aussehen sollte, sodass sie diese nicht verfizieren können. Eve müsste dann nur das Prekey-Bundle abfangen und dann an die jeweilige Person weitersenden. Somit

kann sie sich jeweils als Bob oder Alice ausgeben und kann die gesamte Kommunikation mitlesen.

c) Bei diesem Verfahren muss nur sichergestellt werden, dass die Schlüssel, welche für eine erfolgreiche Nutzung der Signatur gebraucht werden, sicher übertragen werden. Falls das nicht der Fall ist, kann Eve SPK_B,IK_B und die jeweilige Signatur austauschen.

Aufgabe 1.5 Eavesdropping bei der Double Ratchet

1. Wenn Alice zu einem Zeitpunkt von Eve kompromittiert wurde, kann Eve solange sie noch im System von Alice alle Nachrichten lesen und die aktuell genutzten Schlüssel sehen. Wenn Eve durch die Kompromittierung den sending chain key ck_1 erhält, kann Eve alle Nachrichten nur in dem aktuellen Rachet Schritt lesen. Dabei kann sie aber nicht alte oder neue mögliche Nachrichten entschlüsseln. Wenn Eve den aktuellen Root Key rk_1 enthält, kann sie alle Nachrichten aus dem asymmetrischen Ratchet Schritt lesen. Das bedeutet, dass Eve sich aus dem Root Key auch die jeweiligen Chain Keys berechnen kann und so dann Zugriff auf die Nachrichten enthält. Dennoch bekommt sie dadurch keinen Zugriff auf die Diffie-Hellman Chain, sodass ihr das auszutauschende Geheimnis fehlt für den nächsten Root Key beziehungsweise den nachfolgenden asymmetrischen Ratchet Schritt. Sie ist also aus dem nächsten Schritt wieder ausgeschlossen.
2. Um die Kommunikation bei einer Double Ratchet letztendlich die ganze Zeit mitlesen zu können, kann Eve kein passiver Angreifer bleiben. Auch wenn sie einmal Zugang zu dem System hatte, bringen ihr die gewonnenen Informationen nichts über den Schritt hinaus. Selbst als aktiver Angreifer mit einem MitM-Angriff braucht Eve die Identity Keys von Alice und Bob, dazu auch noch Zugriff auf Diffie-Hellman und Root Key Chain. Diese gesamten Vorbedingungen machen einen Angriff aber sehr unwahrscheinlich.
3. a) Bei dieser Variante wird die Eigenschaft der Post Compromise Security nicht mehr erfüllt, da bei einer Kompromittierung der Chain und Secret Key der symmetrischen Ratchet erhalten werden kann. Damit können dann alle weiteren Keys berechnet werden.

   b) Als eine invertierbare Funktion hat diese die Eigenschaft, dass die Funktion in die Rückrichtung verwendet werden kann. Dementsprechend wird die Eigenschaft der Forward Secrecy von dieser Variante nicht erfüllt, da ein Angreifer die KDF invertieren kann und von dem Zeitpunkt aus alle vorherigen Nachrichten lesen kann.

   c) Die MAX_SKIP-Konstante beschreibt wie viele Schlüssel für Nachrichten noch gespeichert werden können, sodass diese noch später noch gelesen werden können. Selbst wenn die Double Ratchet schon einen Schritt weitergegangen ist. Wenn diese also auf einen maximalen Wert gesetzt wurde, werden dementsprechend Schlüssel für eine sehr große Menge an Nachrichten noch gespeichert. Heißt, dass Eve erstmal nur alle verschlüsselten Nachrichten speichern muss, sodass sie dann bei einer Kompromittierung dann Zugriff auf all die Schlüssel erhält und diese nutzen kann um die gespeicherten Nachrichten zu entschlüsseln. Somit wäre die Forward Secrecy nicht mehr gewährleistet.

Aufgabe 1.6 Implementierung des Signal-Protokolls

```
1 Enter name:
2 Alex
3 Successfully authenticated. Token:
4 DAAAANTEkcddRej2iGMTDxAAAACM2fVjUh
5 0eP7rb6qxoGvwMz02cpl7gb6h9u0uHeuNn
6 Enter first message:
7 Hi
8 Message from Bob:
9
```

Alexander Becker (684303)

```
Hi Alex!
I received your message:

-----
Hi
-----

I have some interesting quiz questions for you. If you solve all of them correctly,
I will give you a secret token!
Please write "challenge", so I can send you the first one.
Enter message (or leave empty to exit):
challenge
Change DH key (1: yes)?
1
Message from Bob:

Alright, I didn't tell you that it is a regular expressions quiz,
but now it is too late :P
Your response messages have to match the regular expressions
I send you.
If you need help, you can check this page:
https://www.rexegg.com/regex-quickstart.html

Here is the first one: (its){2,}
Enter message (or leave empty to exit):
itsits
Change DH key (1: yes)?
1
Message from Bob:

That was a good start! Now let's get to something more complicated:
[a-z0-9\.]{5,}@[a-z0-9]+\.[a-z]{3}
Enter message (or leave empty to exit):
aaaaa@aa.aaa
Message from Bob:

I'm genuinely impressed. There are only two left,
so here is the next one: ^.{3,}(its)+?.{4,8}
Enter message (or leave empty to exit):
aaaits?aaa
Change DH key (1: yes)?
1
Message from Bob:

Aaaaand the final one. Beware, this one is really nasty:
<([\w]{3,}).*><([\w]+[\d]+).*>.+?<\/\2>.*?<\/\1>
Enter message (or leave empty to exit):
<aaa><a1>a</a1></aaa>
Change DH key (1: yes)?
1
Message from Bob:

Wow, you made it through the entire quiz! =)
I guess I now have to stick to my earlier promise,
so here is the secret token: {DAAAAPL7ES5k-
cqKXTpa5RAAAACIAfbBeca1L__l9nSKKYCEDk3Mb9
```

```
T7bTXvsy4Ykw7Fm_-cOlvpeww}
Enter message (or leave empty to exit):
```