

**Clara-Fey-Schule**

**Facharbeit**

# **Perfect Forward Secrecy (PFS) mehr Sicherheit und Datenschutz**

Die Enthüllungen, über die staatlich geförderte Überwachung der elektronischen Kommunikation und die Gefahr der rückwirksamen Entschlüsselung der Daten, von Millionen von Menschen haben den Wunsch nach einer zusätzlichen Sicherheits- und Datenschutzschicht für die gesamte elektronische Kommunikation geweckt. In diesem Beitrag wird Perfect Forward Security, das vor mehr als zwanzig Jahren erfunden wurde, als Lösung für dieses Problem vorgestellt.

Verfasser: **Julian Haag**

Kurs: Leistungskurs Mathematik

Betreuer: Murk

Abgabetermin: 2.3.2022

# Inhaltsverzeichnis

<b>Inhaltsverzeichnis.....</b>	<b>2</b>
<b>Abkürzungsverzeichnis.....</b>	<b>3</b>
<b>1 Einführung .....</b>	<b>4</b>
<b>2 Was ist (Perfect) Forward Security.....</b>	<b>6</b>
2.1 PFS vs. FS.....	6
2.2 Zukünftige Sitzungen.....	6
<b>3 Wie erreicht man PFS .....</b>	<b>8</b>
3.1 Diffie-Dellman-Schlüsselaustauschprotokoll .....	8
3.2 Elliptische Kurvenkryptographie.....	9
3.3 Protokolle.....	10
3.3.1 Kein Forward Security.....	11
3.3.2 Hinzufügen von Forward Security.....	11
3.3.3 Hinzufügen von Perfect Forward Security .....	12
3.3.4 Future Secrecy .....	12
3.4 Typische Fehlerquellen.....	13
<b>4 IMPLEMENTIERUNG VON PFS IN VERSCHIEDENEN PROTOKOLLEN .....</b>	<b>14</b>
4.1 Transport Layer Security (TLS) Protocol .....	14
4.1.1 Implementierung von PFS in Apache – Linux .....	16
4.1.2 Implementierung von PFS in nginx .....	16
4.1.3 Typische Fehlerquellen.....	16
4.1.4 Anwendung im Alltag.....	17
4.2 Secure Shell (SSH) Protocol.....	17
4.2.1 Was kann schiefgehen .....	19
4.2.2 Anwendung im Alltag.....	19
<b>5 Abschluss .....</b>	<b>20</b>
<b>Selbständigkeitserklärung.....</b>	<b>21</b>

## Abkürzungsverzeichnis

DHE = Diffie-Hellman-Schlüsselaustausch

RSA = Rivest-Shamir-Adleman- Kryptosystem

PFS = Perfect-Forward-Security / Perfect Forward Secrecy

Bob & Alice = Synonyme für Sender und Empfänger<sup>1</sup>

RFC = Request for Comments<sup>2</sup>

TLS = Transport Layer Security<sup>3</sup>

VPN = Virtual private network<sup>4</sup>

IPSec = Internet Protocol Security<sup>5</sup>

VoIP = Voice over IP<sup>6</sup>

---

<sup>1</sup> [https://de.wikipedia.org/wiki/Alice\\_und\\_Bob](https://de.wikipedia.org/wiki/Alice_und_Bob)

<sup>2</sup> [https://en.wikipedia.org/wiki/Request\\_for\\_Comments](https://en.wikipedia.org/wiki/Request_for_Comments)

<sup>3</sup> [https://en.wikipedia.org/wiki/Transport\\_Layer\\_Security](https://en.wikipedia.org/wiki/Transport_Layer_Security)

<sup>4</sup> [https://en.wikipedia.org/wiki/Virtual\\_private\\_network](https://en.wikipedia.org/wiki/Virtual_private_network)

<sup>5</sup> <https://en.wikipedia.org/wiki/IPsec>

<sup>6</sup> [https://en.wikipedia.org/wiki/Voice\\_over\\_IP](https://en.wikipedia.org/wiki/Voice_over_IP)

# 1 Einführung

Millionen von Websites und Milliarden von Menschen verlassen sich auf Transport Layer Security (TLS), IPSec, VPN-Software und ähnliche Protokolle, um die elektronische Übertragung sensibler und persönlicher Daten zu schützen, in der Erwartung, dass die Verschlüsselung Sicherheit und Datenschutz garantiert. Die Schwierigkeit der Kryptanalyse der in diesen Protokollen verwendeten Verschlüsselungsalgorithmen gewährleistet die Sicherheit der verschlüsselten Nachrichten. Die kryptografischen Algorithmen haben jedoch oft eine Schwachstelle. Heute kann jeder eine verschlüsselte oder unlesbare Kommunikation während der Übertragung aufzeichnen. Später, wenn die verwendeten Schlüssel kompromittiert wurden oder die asymmetrische Kryptanalyse so weit fortgeschritten ist, dass sie geknackt werden kann, kann ein Angreifer die heutige Übertragung rückwirkend entschlüsseln.

Die Begriffe "Forward Security" und "Perfect Forward Security" haben sich im umgangssprachlichen Gebrauch leicht weitentwickelt, seit Whitfield Diffie, Paul C. Van Oorschot und Michael J. Wiener vor zwanzig Jahren ein Zwei-Parteien-Protokoll zur gegenseitigen Authentifizierung erfanden und den Begriff prägten.<sup>7</sup> Ihr Protokoll bietet einen authentifizierten Schlüsselaustausch, der sich auf die Verwendung asymmetrischer Techniken konzentriert. Sie definierten PFS als: "Ein authentifiziertes Schlüsselaustauschprotokoll bietet perfect forward secrecy, wenn die Offenlegung von langfristigem geheimen Schlüsselmaterial die Geheimhaltung der ausgetauschten Schlüssel aus früheren Sitzungen nicht gefährdet."<sup>8</sup>

Heute jedoch haben sich die Begriffe "Forward Security" (FS), "Perfect Forward Security" (PFS) und sogar "Future Secrecy"<sup>9</sup> leicht weiterentwickelt und bedeuten etwas anderes. Neben einer ausführlichen Erläuterung der nuancierten Unterschiede zwischen FS, PFS und Future Secrecy werden in diesem Papier eine einfache Erklärung, reale Anwendungen, Vorteile und die Implementierung von PFS in verschiedenen Protokollen behandelt, darunter Transport Layer Security (TLS), Off-the-record (OTR) Messaging, Secure Shell (SSH), Wireless Protected Access II Protocol (WPA2 EAP-PWD), Internet Protocol Security (IPSec) und Virtual Private Networking (VPN).

---

<sup>7</sup> "Authentication and Authenticated Key Exchanges": <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.59.6682&rep=rep1&type=pdf>

<sup>8</sup> Section 4: Desirable Protocol Characteristics <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.59.6682&rep=rep1&type=pdf>

<sup>9</sup> Coined by Moxie Marlinspike and Trevor Perrin in <https://whispersystems.org/blog/advanced-ratcheting/>

**Anmerkung:** Die Begriffe Perfect Forward Security und Perfect Forward Secrecy werden in der Praxis synonym verwendet. In dieser Arbeit wird nur der Begriff "Security" verwendet, aber in der gängigen Literatur sollte man auch den Begriff "Secrecy" finden. Der Begriff "Future Secrecy" ist nicht gebräuchlich, aber in dieser Arbeit wird der Begriff verwenden, um eine spezielle Eigenschaft zu bezeichnen, die in Kürze definiert wird.

## 2 Was ist (Perfect) Forward Security

Forward Security ist ein Merkmal eines authentifizierten Schlüsselaustauschprotokolls, das sicherstellt, dass die Offenlegung eines langfristigen Identitätsschlüssels (wie eines SSL-Zertifikats oder eines SSH-Host-Schlüssels) die Vertraulichkeit, die in den Sitzungen vor der Kompromittierung verschlüsselten Nachrichten nicht beeinträchtigt. Die Sitzungsschlüssel bzw. die kurzfristigen Verschlüsselungsschlüssel sind unabhängig von den langfristigen Identitätsschlüssels - obwohl diese Sitzungsschlüssel durch die Identitätsschlüssels authentifiziert werden.

Jedes Protokoll funktioniert anders, aber im Allgemeinen wird ein Sitzungsschlüssel nicht in nachfolgenden Durchläufen des Protokolls verwendet, so dass es möglich ist, ihn nach jeder Sitzung zu zerstören. Die Zerstörung des Schlüssels unterbricht keine andere Kommunikation, und sobald diese Kurzzeitschlüssel zerstört sind, ist es nicht mehr möglich, einen mit diesen Schlüsseln verschlüsselten Text zu entschlüsseln. Somit bietet Forward Security eine Verteidigungsschicht gegen die rückwirkende Entschlüsselung von Sitzungen im Falle einer Kompromittierung von langfristigen Identitätsschlüssels.

### 2.1 PFS vs. FS

Forward Security wurde als Synonym für Perfect Forward Security verwendet, aber es gibt einen feinen Unterschied zwischen den beiden Begriffen. Der Unterschied bezieht sich speziell auf die Kompromittierung eines Sitzungsschlüssel - nicht eines langfristigen Identitätsschlüssels.

Bei Forward Security ermöglicht die Kompromittierung eines Sitzungsschlüssel die rückwirkende Entschlüsselung früherer Sitzungen. Perfect Forward Security hat die zusätzliche Eigenschaft, dass die Kompromittierung eines Sitzungsschlüssel *keine* Kompromittierung früherer Sitzungen erlaubt.

### 2.2 Zukünftige Sitzungen

Der feine Unterschied zwischen Forward Security und Perfect Forward Security gilt nur für die Kompromittierung von *Sitzungsschlüssel*, nicht für langfristige Identitätsschlüssels. Er gilt auch nur für Angriffe auf *früheren Sitzungen*. Der Begriff "Future Secrecy" wurde geprägt, um zu beschreiben, was mit *zukünftigen* Sitzungen geschieht, wenn ein *Sitzungsschlüssel* offengelegt wird.

Aber selbst mit FS, PFS und Future Secrecy gibt es nichts, was beschreibt, wie sich ein Protokoll gegenüber zukünftigen Sitzungen verhält, wenn ein langfristiger Identitätsschlüssels offengelegt wird. Diese Begriffe werden in der folgenden Tabelle zusammengefasst:

Szenario	Protokoll mit Forward Security	Protokoll mit Perfect Forward Security	Protokoll mit Future Secrecy
Kompromittierung des Identitätsschlüssels Angriffe auf frühere Sitzungen	Sicher (per Definition)	Sicher (per Definition)	Abhängig vom Protokoll
Kompromittierung des Identitätsschlüssels Angriffe auf künftige Sitzungen	Abhängig vom Protokoll	Abhängig vom Protokoll	Abhängig vom Protokoll
Kompromittierung des Sitzungsschlüssel Angriffe auf frühere Sitzungen	Unsicher (nach der Definition)	Sicher (per Definition)	Abhängig vom Protokoll
Kompromittierung des Sitzungsschlüssel Angriffe auf künftige Sitzungen	Abhängig vom Protokoll	Abhängig vom Protokoll	Sicher (per Definition)

### 3 Wie erreicht man PFS

In diesem Abschnitt wird untersucht, wie ein allgemeines Protokoll Perfect Forward Security erreichen kann, und es wird veranschaulicht, wie man Protokolle erstellen könnte, die Sicherheit vor einigen der oben beschriebenen Angriffe bieten bzw. nicht bieten. Es werden die Anforderungen an die Schlüssel, die Nachrichtenfolgen und die Widerstandsfähigkeit bzw. Anfälligkeit des Protokolls gegenüber Schlüsselkompromittierung erläutert. Bei diesen Entwürfen handelt es sich um triviale Beispiele, die die Vorwärtssicherheitseigenschaften veranschaulichen sollen - bestimmte Aspekte werden nicht berücksichtigt: So wird beispielsweise nicht auf langfristige Schlüsselauthentifizierung, Widerruf oder Replay-Angriffe eingegangen.

Da so viele Protokolle auf dem Diffie-Hellman-Schlüsselaustausch aufbauen, um zu einem Sitzungsschlüssel zu gelangen, wird dieser Schlüsselaustausch sowohl in der gewöhnlichen als auch in der elliptischen Kurvenvariante kurz beschreiben.

#### 3.1 Diffie-Dellman-Schlüsselaustauschprotokoll

Der Diffie-Hellman-Schlüsselaustausch<sup>10</sup> ist eine Implementierung der Kryptographie mit öffentlichem Schlüssel, die dazu dient, ein gemeinsames Geheimnis zwischen zwei Parteien über ein unsicheres öffentliches Medium zu ermitteln, ohne vorher etwas zu teilen. Bei diesem Algorithmus gibt es drei Parameter (auch als DH-Parameter bekannt):

- $p$  - eine sehr große Primzahl
- $g$  - eine primitive Wurzel<sup>11</sup> modulo  $p$ , auch bekannt als Generator der multiplikativen Gruppe der ganzen Zahlen<sup>12</sup> modulo  $p$ ; so dass  $g < p$
- $x$  - ein privater Schlüssel

Für jede Zahl  $n$  gibt es eine solche Potenz  $x$  von  $g$ , dass  $n \equiv g^x \pmod{p}$ . Dies ist der öffentliche Schlüssel. Aus der Kenntnis von  $x$ ,  $g$  und  $n$  kann nun jede Partei den gemeinsamen geheimen Schlüssel  $s$  berechnen. Beide Parteien halten ihr  $x$  und  $s$  geheim, während alle anderen Werte -  $p$ ,  $g$  und  $n$  - über einen unsicheren Kanal ausgetauscht werden.

Betrachten man unsere alten Freunde Alice und Bob, die sicher kommunizieren wollen. Alice wählt eine zufällige ganze Zahl  $a$  als ihren privaten Schlüssel und berechnet ihren

---

<sup>10</sup> <http://www.ietf.org/rfc/rfc2631.txt>

<sup>11</sup> [http://math.arizona.edu/~savitt/mathcamp/1999/primitive\\_roots.pdf](http://math.arizona.edu/~savitt/mathcamp/1999/primitive_roots.pdf)

<sup>12</sup> [http://en.wikipedia.org/wiki/Multiplicative\\_group\\_of\\_integers\\_modulo\\_n](http://en.wikipedia.org/wiki/Multiplicative_group_of_integers_modulo_n)

öffentlichen Schlüssel als  $n_a \equiv g^a \pmod{p}$ . In ähnlicher Weise wählt Bob b und berechnet seinen öffentlichen Schlüssel als  $n_b \equiv g^b \pmod{p}$ . Beide tauschen ihre öffentlichen Schlüssel  $n_a$  und  $n_b$  aus. Um schließlich den gemeinsamen geheimen Schlüssel (symmetrischer Schlüssel) zu bestimmen, berechnet Alice  $s \equiv (n_b)^a \pmod{p}$  und Bob  $s \equiv (n_a)^b \pmod{p}$ .

Alice und Bob kommen beide auf denselben Wert, da  $(g^a)^b \pmod{p}$  und  $(g^b)^a \pmod{p}$  gleich sind. Sobald Alice und Bob das gemeinsame Geheimnis berechnet haben, können sie es als Verschlüsselungsschlüssel verwenden, der nur ihnen bekannt ist, um Nachrichten über denselben offenen Kommunikationskanal zu senden.

Es gibt keinen effizienten Algorithmus zur Lösung des diskreten Logarithmusproblems<sup>13</sup>, der es dem Gegner erleichtern würde, x aus der Kenntnis von p, g und n zu berechnen und das Diffie-Hellman-Problem zu lösen<sup>14</sup>. Die Sicherheit des Protokolls ist also so stark wie die Schwierigkeit, das diskrete Logarithmusproblem für sehr große Primzahlen effizient zu lösen.

## 3.2 Elliptische Kurvenkryptographie

Die Elliptische-Kurven-Kryptographie<sup>15</sup> (ECC) ist eine auf der Elliptischen-Kurven-Theorie<sup>16</sup> basierende Verschlüsselungstechnik für öffentliche Schlüssel, mit der sich schnellere, kleinere und effizientere kryptographische Schlüssel erstellen lassen. ECC generiert Schlüssel durch die Eigenschaften der elliptischen Kurvengleichung anstelle der traditionellen Methoden wie der Multiplikation zweier sehr großer Primzahlen.

Die ECC basiert auf den Eigenschaften einer bestimmten Art von Gleichung, die sich aus der mathematischen Gruppe<sup>17</sup> der Punkte ergibt, an denen die Linie die Achsen schneidet.

Eine elliptische Kurve ist eine Menge von Punkten (x, y) in einer Ebene, die eine Gleichung der Form  $y^2 = x^3 + ax + b$  (wobei a und b Konstanten sind) erfüllen, zusammen mit

---

<sup>13</sup> [http://en.wikipedia.org/wiki/Discrete\\_logarithm\\_problem](http://en.wikipedia.org/wiki/Discrete_logarithm_problem)

<sup>14</sup> [https://en.wikipedia.org/wiki/Diffie-Hellman\\_problem](https://en.wikipedia.org/wiki/Diffie-Hellman_problem)

<sup>15</sup> <http://tools.ietf.org/html/rfc6090>

<sup>16</sup> [https://en.wikipedia.org/wiki/Elliptic\\_curve](https://en.wikipedia.org/wiki/Elliptic_curve)

<sup>17</sup> Eine Gruppe ist eine Menge von Werten, für die Operationen (\*) an zwei beliebigen Mitgliedern der Gruppe durchgeführt werden können, um ein drittes Mitglied zu erzeugen, wobei die Gruppenaxiome wie Schließung, Assoziativität, Identität und Invertierbarkeit erfüllt werden. Mehr über Gruppen - [https://en.wikipedia.org/wiki/Group\\_\(mathematics\)](https://en.wikipedia.org/wiki/Group_(mathematics))

einem zusätzlichen Punkt  $o$ , der als Unendlichkeitspunkt bezeichnet wird. Für Anwendungen in der Kryptographie betrachten man ein endliches Feld<sup>18</sup> oder ein Galois-Feld mit  $q$  Elementen (wobei  $q$  eine endliche Menge von ganzen Zahlen modulo einer Primzahl ist), das als  $F_q$  oder  $GF(q)$  dargestellt wird.<sup>19</sup>

Noch einmal: Alice und Bob wollen sicher mit ECC kommunizieren. Sie einigen sich auf eine (nicht geheime) elliptische Kurve und einen (nicht geheimen) festen Kurvenpunkt  $F_q$ . Alice wählt eine geheime Zufallszahl  $k_a$ , die ihr geheimer Schlüssel ist, und veröffentlicht den Kurvenpunkt  $P_a = k_a * F_q$  als ihren öffentlichen Schlüssel. Bob tut dasselbe. Sein geheimer Schlüssel ist  $k_b$  und der Kurvenpunkt ist  $P_b = k_b * F_q$ . Um nun verschlüsselte Nachrichten zu senden, kann Alice einfach den gemeinsamen geheimen Schlüssel  $s = k_a * P_b$  berechnen. Bob kann die gleiche Zahl  $s$  berechnen, indem er  $k_b * P_a$  berechnet, da  $k_b * P_a = k_b * (k_a * F_q) = (k_b * k_a) * F_q = k_a * (k_b * F_q) = k_a * P_b$ .

Die Sicherheit des Verfahrens beruht auf der Annahme, dass es schwierig ist,  $k$  bei  $F_q$  und  $k * F_q$  zu berechnen. ECC kann ein höheres Maß an Sicherheit mit einem kleineren Schlüssel und daher mit geringerer Rechenleistung als andere Kryptosysteme bieten. Bei Protokollen, die auf elliptischen Kurven basieren, wird davon ausgegangen, dass es nicht möglich ist, den diskreten Logarithmus eines zufälligen elliptischen Kurvenelements in Bezug auf einen öffentlich bekannten Basispunkt zu finden - dies wird als Problem des diskreten Logarithmus der elliptischen Kurve bezeichnet.

### 3.3 Protokolle

Jedes sichere Kommunikationsprotokoll hat Authentifizierung, Vertraulichkeit und Integrität der Nachricht als seine Eigenschaften. Die Implementierungsspezifika dieser Eigenschaften können je nach den Anforderungen des Protokolls variieren. Auch unsere Spielzeugprotokolle zielen darauf ab, diese Eigenschaften zu erreichen.

Die Authentifizierung stellt sicher, dass zwei Parteien über einen öffentlichen Kanal mit dem beabsichtigten Empfänger kommunizieren. Die Authentifizierung kann mit Hilfe von Public-Key-Infrastrukturen und Zertifizierungsstellen oder durch die gemeinsame Nutzung der öffentlichen Schlüssel der beiden Parteien erreicht werden. Der Protokolldesigner kann die Vertraulichkeit durch die Verwendung eines starken Verschlüsselungsalgorithmus erreichen. Die Wahl des symmetrischen oder asymmetrischen Verschlüsse-

---

<sup>18</sup> Die Voraussetzung dafür, dass eine Menge von Elementen in einem Feld liegt, ist, dass Operationen wie Addition, Subtraktion, Multiplikation und Division möglich sind - diese Operationen führen immer zu einem Ergebnis, das im Feld liegt, mit Ausnahme der Division durch Null, die undefiniert ist. Mehr über das endliche Feld: [https://en.wikipedia.org/wiki/Finite\\_field](https://en.wikipedia.org/wiki/Finite_field)

<sup>19</sup> Weitere Einzelheiten über die Funktionsweise von ECC und die Anforderungen an die Wahl der elliptischen Kurve - <http://www.cryptoman.com/elliptic.htm>

lungsalgorithmus hängt vom Zweck und der Effizienz der Implementierung ab. Die Verwendung eines Nachrichtenauthentifizierungscodes (MAC) ist eine Möglichkeit, die Integrität der Nachricht zu überprüfen.

Jetzt hat man das Grundgerüst eines sicheren Kommunikationsprotokolls: Lassen Sie uns anhand von Beispielen die Eigenschaften Forward Security und Future Secrecy veranschaulichen. Alice und Bob wollen miteinander kommunizieren und haben auf irgendeine Weise Vertrauen in die langfristigen Identitätsschlüssel des jeweils anderen aufgebaut.

### 3.3.1 Kein Forward Security

Alice erstellt einen zufälligen symmetrischen Schlüssel, verschlüsselt ihn mit Bobs öffentlichem Schlüssel und übermittelt ihn dann an Bob. Bob entschlüsselt den symmetrischen Schlüssel, und beide beginnen, den symmetrischen Schlüssel für die Verschlüsselung der Kommunikation für den Rest der Sitzung zu verwenden. Wenn sie die Sitzung beenden und eine neue Sitzung beginnen, wird sie mit einem neuen Schlüssel wiederholt.

Wenn einer der beiden Identitätsschlüssel kompromittiert wird, ist die gesamte vergangene Kommunikation kompromittiert, da dieser Schlüssel zur Entschlüsselung aller unter ihm verschlüsselten symmetrischen Schlüssel verwendet werden kann. Die Kompromittierung des Identitätsschlüssels ermöglicht auch Angriffe auf alle zukünftigen Sitzungen.

Obwohl dieses triviale Protokoll so einfach ist, wie man es nur machen kann (und im Kern die Funktionsweise von TLS ohne PFS darstellt), verfügt es über Future Secrecy, da die Offenlegung eines einzelnen Sitzungsschlüssels keinen Angriff auf zukünftige (oder frühere) Sitzungen ermöglicht.

### 3.3.2 Hinzufügen von Forward Security

Betrachte man nun dasselbe Beispiel, aber dieses Mal haben Alice und Bob das Protokoll so geändert, dass es die Eigenschaft der Forward Security besitzt. Alice und Bob verwenden einen DH-Schlüsselaustausch (authentifiziert durch die langfristigen Identitätsschlüssel), um einen symmetrischen Verschlüsselungsschlüssel für die erste Sitzung zu erzeugen. Sie verwenden diesen Sitzungsschlüssel in dieser und allen zukünftigen Sitzungen.

Dieses Protokoll bietet Vorwärtssicherheit. Die Offenlegung des Sitzungsschlüssels ermöglicht jedoch Angriffe auf frühere Sitzungen, so dass dieses Protokoll keine perfekte Vorwärtssicherheit bietet. Es ermöglicht auch die Kompromittierung aller zukünftigen Sitzungen, so dass es auch keine Zukunftssicherheit bietet.

Die Offenlegung eines Identitätsschlüssels kompromittiert keine zukünftigen Sitzungen, es sei denn, das Protokoll wird von Grund auf neu gestartet. In diesem Fall kann der Angreifer einen Man-in-the-Middle-Angriff auf den DH-Austausch durchführen und sich als eine der beiden Parteien ausgeben.

### 3.3.3 Hinzufügen von Perfect Forward Security

Wie man gesehen hat, konnte der Angreifer durch die Kompromittierung des Sitzungsschlüssels alle vorherigen Sitzungen entschlüsseln. Man möchte nun dieses Problem beheben. Anstatt für jede Verbindung denselben symmetrischen Schlüssel zu verwenden, werden Alice und Bob das Geheimnis hashen. Für die erste Sitzung verwenden sie einen vereinbarten Schlüssel  $K$ , und am Ende der Sitzung speichern sie  $H(K)$ . Bei der nächsten Sitzung verwenden sie  $H(K)$  als Schlüssel, und nach Abschluss der Sitzung speichern sie  $H(H(K))$ .

Dieser Vorschlag bietet Perfect Forward Security - wenn ein Angreifer einen Sitzungsschlüssel kompromittiert, kann er frühere Sitzungen nicht entschlüsseln, da er die Hash-Funktion nicht umkehren kann. Allerdings kann ein Angreifer künftige Sitzungen entschlüsseln, da er die Hash-Funktion auch iterieren kann.

Wie beim vorherigen Protokoll beeinträchtigt die Offenlegung eines Identitätsschlüssels zukünftige Sitzungen nicht, es sei denn, das Protokoll wird von Grund auf neu gestartet. In diesem Fall kann der Angreifer einen Man-in-the-Middle-Angriff auf den DH-Austausch durchführen und sich gegenüber der anderen Partei als eine der beiden ausgeben.

### 3.3.4 Future Secrecy

Betrachten man noch einmal die Angriffe:

1. Kompromittierung des Identitätsschlüssels - Angriff auf frühere Sitzungen
2. Kompromittierung des Identitätsschlüssels für Angriffe auf zukünftige Sitzungen
3. Kompromittierung des Sitzungsschlüssels für Angriffe auf frühere Sitzungen
4. Kompromittierung des Sitzungsschlüssels für Angriffe auf zukünftige Sitzungen

Bisher sind auf einen Schutz gegen die erste und dritte Eigenschaft eingegangen - fügen man nun die Eigenschaft "Future Secrecy" hinzu, um Sicherheit gegen die letzte Eigenschaft zu erreichen. Dieses Protokoll ist sogar deutlich einfacher als die letzten beiden. Alice und Bob führen einen DH-Austausch durch, der mit ihren langfristigen Identitätsschlüsseln authentifiziert wird. Sie verwenden diesen Austausch, um sich bei jedem Durchlauf des Protokolls auf einen neuen symmetrischen Schlüssel zu einigen.

Das war's (und in der Tat funktioniert TLS mit PFS in etwa so). Die Kompromittierung eines Identitätsschlüssels ermöglicht keine Kompromittierung früherer Sitzungen. Die Kompromittierung eines Sitzungsschlüssels ermöglicht keine Kompromittierung früherer oder künftiger Sitzungen.

Und es ist möglich, dieses Protokoll noch zu erweitern! Indem der DH-Schlüsselaustausch sowohl mit dem langfristigen Identitätsschlüssel als auch mit dem vorherigen Sitzungsschlüssel authentifizieren, kann man sicherstellen, dass ein Angreifer, der nur den Identitätsschlüssel kompromittiert, keine zukünftigen Protokolle angreifen kann - er

müsste sowohl den Identitätsschlüssel als auch den letzten Sitzungsschlüssel kompromittieren.

Die Aufbewahrung von Schlüsselmaterial aus der vorherigen Sitzung zur Authentifizierung der nächsten Sitzung ist jedoch kompliziert. Während Protokolle wie ZRTP<sup>20</sup> dazu in der Lage sind, ist dies bei TLS viel schwieriger, da hier jederzeit mehrere Sitzungen aktiv und offen sind. Wenn Sie mehr über dieses Konzept (oft als "Ratcheting" bezeichnet) erfahren möchten, empfehlen ich Ihnen die Arbeit von Moxie Marlinspike und Trevor Perrin.<sup>21</sup>

### 3.4 Typische Fehlerquellen

Selbst bei einem gut konzipierten Protokoll gibt es verschiedene Angriffe, die die Vertraulichkeit einer Verbindung gefährden können.

- **Kompromittierung des Langzeit-Identitätsschlüssels:** Wie bereits erläutert, kann der Langzeit-Identitätsschlüssel kompromittiert werden, was es einem Angreifer in der Regel ermöglicht, einen Impersonationsangriff<sup>22</sup> durchzuführen.
- **Schlechter Zufallszahlengenerator:** Jeder Schlüssel, der mit einem fehlerhaften oder defekten Zufallszahlengenerator generiert wurde, ist deutlich schwächer als seine angekündigte Bit Länge, was effizientere Brute-Force-Angriffe zur Wiederherstellung des Schlüssels ermöglicht.
- **Ungleiche Algorithmus-Stärke:** Unter bestimmten Umständen kann der PFS-Algorithmus (z. B. der DH-Handshake) schwächer sein als die Identitätsschlüssel. Dies ermöglicht es einem Angreifer, das schwächste Glied des Protokolls anzugreifen. Dies ist eine schlechte Praxis und sollte vermieden werden, aber der Einsatz kann zu Problemen führen, z.B. wenn TLS heute DH in 1024-bit-Gruppen verwendet, aber mit 2048-bit-RSA-Identitätsschlüsseln authentifiziert.
- **Fortschritte bei den Angriffsalgorithmen:** Die Public-Key-Kryptografie basiert auf Falltürfunktionen wie dem Faktorisieren oder dem diskreten Log-Problem - Verbesserungen beim Angriff auf diese Funktionen führen zur Schwächung jeder Public-Key-Kryptografie, einschließlich der PFS-Algorithmen.

---

<sup>20</sup> <http://tools.ietf.org/html/rfc6189>

<sup>21</sup> <https://whispersystems.org/blog/advanced-ratcheting/>

<sup>22</sup> [https://en.wikipedia.org/wiki/Spoofing\\_attack](https://en.wikipedia.org/wiki/Spoofing_attack)

## 4 IMPLEMENTIERUNG VON PFS IN VERSCHIEDENEN PROTOKOLLEN

In diesem Abschnitt werden mehrere häufig verwendete Protokolle und ihre Verwendung von Forward Security erläutert. Für jedes dieser Protokolle wird eine kurze Beschreibung und ein Durchgang durch das Protokoll gegeben, wobei der Schwerpunkt auf dem Schlüsselaustauschmechanismus mit und ohne Forward Security liegt. Später werden die Bedingungen, Konfigurationen und Vorbehalte zur Erreichung von Forward Security in diesen Protokollen sowie deren Nutzen in der Praxis erläutert.

### 4.1 Transport Layer Security (TLS) Protocol

Bei einem typischen TLS-Handshake erfolgt die Authentifizierung einseitig, d. h. nur der Server ist gegenüber dem Client authentifiziert. Bei einem Nicht-PFS-TLS-Handshake, der RSA-Schlüssel verwendet, wird der authentifizierte Schlüsselaustausch über den folgenden Mechanismus erreicht:

1. Der Server sendet seinen öffentlichen Schlüssel, der fast immer in einem x509 Zertifikat enthalten ist.
2. Nach erfolgreicher Überprüfung des Zertifikats antwortet der Client mit dem Pre-Master-Geheimnis, das mit dem öffentlichen Schlüssel des Servers verschlüsselt ist.
3. Der Server entschlüsselt die Schlüsselaustauschnachricht des Clients und beide Parteien leiten aus dem Pre-Master-Geheimnis einen gemeinsamen Sitzungsschlüssel ab.
4. Beide Parteien beginnen die Kommunikation über einen verschlüsselten Kanal unter Verwendung des gemeinsamen Sitzungsschlüssels.

Der oben beschriebene TLS-Handshake verfügt nicht über Forward Security. Der öffentliche Schlüssel des Servers wird zur Verschlüsselung des Schlüsselmaterials der einzelnen Sitzungen verwendet, das nur mit dem privaten Schlüssel des Servers entschlüsselt werden kann. Die Sicherheit aller Sitzungen hängt von einem einzigen statischen Schlüssel ab (dem privaten Schlüssel des Servers). Wenn der private Schlüssel des Servers compromittiert wird, ist die Sicherheit aller unter diesem Schlüssel aufgebauten Sitzungen gefährdet.

Die Eigenschaft von PFS stellt sicher, dass keine langfristige Schlüsselkompromittierung die Sicherheit vergangener Sitzungen beeinträchtigen kann, was bei TLS durch die authentifizierte Diffie-Hellman (DH)-Schlüsselvereinbarung erreicht wird. Im Gegensatz zu RSA-Handshakes dient der langfristige RSA-Schlüssel des Servers bei DH-Handshakes nur der Authentifizierung: Er wird nur zum Signieren der DH-Schlüsselparameter des Servers verwendet. In TLS wird PFS oft als Ephemeral Diffie-Hellman bezeichnet

und in den Chiffriernamen mit DHE abgekürzt. Wenn Ephemeral DH verwendet wird, erzeugen beide Parteien bei jedem Handshake einen neuen DH-Schlüssel, und es wird Perfect Forward Security erreicht, da die Sicherheit jeder Sitzung von einer anderen Instanz des DH-Problems abhängt.

Bei einem RSA-Handshake muss der Server eine Entschlüsselungsoperation mit seinem privaten Schlüssel durchführen; bei einem Ephemeral DH-Handshake muss der Server jedoch zusätzlich zu zwei Potenzierungsoperationen für die DH-Parameter eine Signieroperation durchführen. Diese Operationen können zwar optimiert werden, sind aber dennoch kostspielig. Für effizientere DH-Operationen spezifiziert TLS daher eine Erweiterung für elliptische Kurven, die eine gleichwertige Sicherheit bei geringerem Rechenaufwand bieten. ECC in TLS<sup>23</sup> umfasst zwei Arten des Diffie-Hellman-Schlüsselaustauschs mit elliptischen Kurven (ECDH) - den Schlüsselaustausch mit festen Schlüsseln mit ECDH-Zertifikaten (denen PFS fehlt) und den ephemeren ECDH-Schlüsselaustausch mit einem RSA- oder ECDSA-Zertifikat.

In einem TLS-Server kann PFS konfiguriert werden, indem der Server die Reihenfolge der TLS-Chiffriersuiten beachtet und die ECDHE- und DHE-Suiten an den Anfang der Liste stellt. Die Suiten, die für PFS aktiviert werden müssen, sind:

- DHE-DSS-AES-<k>-GCM-SHA<h>
- DHE-RSA-AES-<k>-GCM-SHA<h>
- DHE-DSS-AES-<k>-CBC-SHA<h>
- DHE-RSA-AES-<k>-CBC-SHA<h>
- DHE-RSA-AES-<k>-CBC-SHA
- ECDHE-ECDSA-AES<k>-GCM-SHA<h>-P<c>
- ECDHE-ECDSA-AES<k>-CBC-SHA<h>-P<c>
- ECDHE-RSA-AES-<k>-CBC-SHA<h>-P<c>
- ECDHE-RSA-AES-<k>-CBC-SHA-P<c>

Hier möglichen Werte für die Schlüsselgrößen (k) sind 256 und 128; die Hash-Digest-Größen (h) sind 512, 384 und 256 und die Kurvengrößen (c) sind 384 und 256. Darüber hinaus können die GCM-Suiten (die in TLS 1.2 unterstützt werden, aber nicht weit verbreitet sind) in der Hierarchie höher angesiedelt werden, da sie eine bessere Sicherheit bieten als der Cipher Block Chaining (CBC)-Modus.

---

<sup>23</sup> <http://tools.ietf.org/html/rfc4492>

#### 4.1.1 Implementierung von PFS in Apache<sup>24</sup> – Linux

PFS erfordert Apache 2.2.\* oder höher. Hier ist eine Beispielkonfiguration für mod\_ssl, die funktioniert, um PFS für den aktuellen stabilen Zweig von Apache 2.4.\* zu aktivieren:

```
SSLProtocol +TLSv1 +TLSv1.1 +TLSv1.2
SSLHonorCipherOrder On
SSLCipherSuite ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-RSA-AES128-GCM-SHA256:
ECDHE-RSA-AES128-SHA256:ECDHE-RSA-RC4-SHA:ECDHE-RSA-AES256-SHA:DHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES256-SHA256:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA:DHE-RSA-AES128-SHA:RC4-SHA:AES256-GCM-SHA384:AES256-SHA256:CAMELLIA256-SHA:ECDHE-RSA-AES128-SHA: AES128-GCM-SHA256:AES128-SHA256:AES128-SHA:CAMELLIA128-SHA
```

#### 4.1.2 Implementierung von PFS in nginx<sup>25</sup>

Um PFS in nginx zu implementieren, fügen Sie die folgenden Chiffren-Suites in die Konfigurationsdatei ein.

```
ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
ssl_prefer_server_ciphers on;
ssl_ciphers "EECDH+ECDSA+AESGCM EECDH+aRSA+AESGCM EECDH+ECDSA+SHA384 EECDH+ECDSA+SHA256 EECDH+aRSA+SHA384 EECDH+aRSA+SHA256 EECDH+aRSA+RC4 EECDH EDH+aRSA RC4 !aNULL !eNULL !LOW !3DES !MD5 !EXP !PSK !SRP !DSS";
```

#### 4.1.3 Typische Fehlerquellen

Der Client gibt die Liste der von ihm unterstützten Cipher Suites und elliptischen Kurven in der ClientHello-Nachricht des TLS-Handshake bekannt. Wählt der Server eine ECDH- oder ECDHE-Chartersuite, wählt er auch eine Kurve, die sowohl der Server als auch der Client unterstützen. Entscheidet sich der Server jedoch für eine DH- oder DHE-Verschlüsselung, kann er eine beliebige DH-Gruppe wählen. Einige Server wählen DH-Gruppen, die so klein wie 256 Bits sind. Kleinere DH-Gruppen bedeuten, dass die Verbindung mit relativ geringem Aufwand aufgebrochen werden kann. Die Implementierung von PFS kann also durch die Verwendung zu kleiner DH-Gruppen verpfuscht werden. Im Idealfall sollte die DH-Gruppe der Größe des RSA-Schlüssels entsprechen oder diese übertreffen (vorzugsweise > 1024 Bits).

Außerdem gibt es für den Server keine Möglichkeit, die Sicherheitsanforderungen des Klienten zu ermitteln. Hält der Client die gewählte Gruppe nicht für stark genug, bricht er die Verbindung ab. Für ECC erlauben die TLS-Erweiterungen einem Client, die Verwendung bestimmter Kurven und Punktformate auszuhandeln. Wenn der Vorschlag von Gillmore<sup>26</sup> angenommen wird, wird TLS auch die Aushandlung von DH-Feldern unterstützen.

<sup>24</sup> <https://httpd.apache.org/>

<sup>25</sup> <https://www.nginx.com/>

<sup>26</sup> <http://tools.ietf.org/html/draft-ietf-tls-negotiated-dl-dhe>

Eine weitere Möglichkeit, PFS zu vereiteln, ist eine unsachgemäße Sitzungsverwaltung auf der Serverseite. Die Sitzungswiederaufnahme<sup>27</sup> erfolgt bei TLS entweder über die Sitzungs-ID<sup>28</sup> oder über Sitzungstickets. Wenn der Server die Sitzungsschlüssel lange nach Beendigung der Sitzung auf der Serverseite speichert, können diese Sitzungsschlüssel später zur Entschlüsselung der verschlüsselten Nachrichten verwendet werden, um die Leistung zu optimieren.

Die TLS-Implementierung ist Perfectly Forward Secure (Schutz früherer Sitzungen vor der Kompromittierung von Sitzungsschlüsseln) und Forward Secret (Schutz künftiger Sitzungen vor der Kompromittierung von Sitzungsschlüsseln); die Offenlegung eines langfristigen Identitätsschlüssels ermöglicht es dem Angreifer jedoch, sich als Server auszugeben und künftige Sitzungen zu kompromittieren.

#### 4.1.4 Anwendung im Alltag

TLS ist das Rückgrat für die sichere Kommunikation über das Internet. Am häufigsten wird es für die Absicherung von Webbrowser-Sitzungen verwendet, aber auch für andere Aufgaben wie die Absicherung von E-Mail-Servern oder jede Art von Client-Server-Transaktionen findet es breite Anwendung. TLS kann auch zum Tunneln eines gesamten Netzwerkstapels verwendet werden, um ein VPN aufzubauen, und zur Authentifizierung und Verschlüsselung des Session Initiation Protocol (SIP), das häufig bei VoIP (Voice over IP) verwendet wird.

## 4.2 Secure Shell (SSH) Protocol

Das Secure Shell-Protokoll (SSH) ist ein Protokoll für die sichere Fernanmeldung und andere sichere Netzwerkdienste über ein unsicheres Netzwerk. SSH besteht aus drei Protokollen, die normalerweise über TCP laufen:

- **Transportschichtprotokoll:** Bietet Server-Authentifizierung, Datenvertraulichkeit und Datenintegrität mit Forward Security; die Transportschicht kann optional Kompression bieten.
- **Benutzeroauthentifizierungsprotokoll:** Authentifiziert den Benutzer gegenüber dem Server.
- **Verbindungsprotokoll:** Multiplexen mehrerer logischer Kommunikationskanäle über eine einzige SSH Verbindung.

---

<sup>27</sup> <http://tools.ietf.org/html/rfc5077>

<sup>28</sup> [https://en.wikipedia.org/wiki/Session\\_ID](https://en.wikipedia.org/wiki/Session_ID)

Unser Interesse gilt dem Transport Layer Protocol, bei dem der Schlüsselaustausch auf der Grundlage eines öffentlich-privaten Schlüsselpaars des Servers erfolgt. Wie bereits erwähnt, besteht die Lösung für PFS in der Verwendung eines ephemeren Schlüsselaustauschs. Anstatt Nachrichten immer mit demselben statischen Schlüssel zu verschlüsseln, handeln die Teilnehmer eines Nachrichtenaustauschs Geheimnisse durch einen ephemeren Schlüsselaustausch aus.

### SSH v1:

Nach dem Aufbau der ersten TCP-Verbindung sendet der Server dem Client zwei Schlüssel: einen Host-Schlüssel und einen Server-Schlüssel. Der Host-Schlüssel ist ein dauerhafter asymmetrischer Schlüssel, der zur Identifizierung des Servers dient, während der Server-Schlüssel ein temporärer asymmetrischer Schlüssel ist. Der Client verschlüsselt den Sitzungsschlüssel doppelt mit dem Serverschlüssel und dem Hostschlüssel. Der Serverschlüssel wird in regelmäßigen Abständen verworfen, standardmäßig jede Stunde.

Der Serverschlüssel kann nach jeder Sitzung verworfen werden, aber die Dauer wird je nach Bedarf festgelegt. Der Benutzer kann die Lebensdauer und Größe des ephemeren Serverschlüssels festlegen, indem er die Optionen `Key_regeneration_interval` (gibt an, wie lange in Sekunden der Server warten soll, bevor er seinen Schlüssel automatisch regeneriert) und `ServerKeyBits` (gibt an, wie viele Bits im Serverschlüssel verwendet werden sollen) in der Datei `sshd_config` entsprechend einstellt. Dieser Serverschlüssel bietet PFS in SSH v1. Sobald der Server den Serverschlüssel zerstört, ist es nicht mehr möglich, den Sitzungsschlüssel wiederherzustellen.

### SSH v2:

Nach dem Aufbau der TCP-Verbindung einigen sich beide Systeme über die `SSH_MSG_KEXINIT`-Meldung auf einen Sitzungsschlüssel unter Verwendung des DH-Schlüsselaustauschs.<sup>29</sup> SSH v2 verwendet den langfristigen Host-Schlüssel nur zur Authentifizierung des Servers während des DH- oder ECDH-Schlüsselaustauschs; es verschlüsselt keine Daten damit.

In SSH v2 wird DH zum Einrichten der Sitzungsschlüssel verwendet. DH bietet von Haus aus PFS, ohne dass ein zweiter Serverschlüssel erforderlich ist, wie es bei SSH v1 der Fall war. SSH v2 zerstört die Informationen, die den Sitzungsschlüssel kompromittieren würden, sofort nach Beendigung der Sitzung und nicht erst einige Zeit später. SSH v2 unterstützt auch die Verwendung des in RFC-5656<sup>30</sup> definierten elliptischen Kurvenalgorithmus. Der Schlüsselaustausch zwischen den Servern beginnt mit der Nachricht `SSH_MSG_KEX_ECDH_INIT` nach der ersten TCP-Verbindung.

Ein Benutzer kann die DH-Austauschmethoden in `sshd_config` wie folgt konfigurieren:

---

<sup>29</sup> <http://tools.ietf.org/html/rfc4253#section-8>

<sup>30</sup> <http://tools.ietf.org/html/rfc5656>

KexAlgorithms ecdh-sha2-nistp256, ecdh-sha2-nistp384, ecdh-sha2-nistp521, diffie-hellman-group-exchange-sha256, diffie-hellman-group14-sha1, diffie-hellman-group-exchange-sha1, diffie-hellman-group1-sha1

Der "diffie-hellman-<group\_value>-exchange-<hash\_value>" spezifiziert Diffie-Hellman Group und Key Exchange mit HASH. Nach RFC-4419<sup>31</sup> sind die akzeptablen Gruppen mit Mindest- und Maximalwerten einer Modulationslänge von k Bits 1024 bis 8192.

SSH-Sitzungen, die aus einem Schlüsselaustausch unter Verwendung der Diffie-Hellman-Methoden (einschließlich "diffie-hellman-group-sha" und "diffie-hellman-group-sha") resultieren, sind selbst dann sicher, wenn privates Schlüssel-/Authentifizierungs-material später offengelegt wird, aber wie bei allen PFS-Protokollen nicht, wenn die Sitzungsschlüssel offengelegt werden.

#### 4.2.1 Was kann schiefgehen

SSH v2 erreicht Perfect Forward Security, indem es einen DH-Schlüsselaustausch als Standard-Schlüsselaustauschmechanismus verwendet. Wenn ein Sitzungsschlüssel kompromittiert wird, kann der Angreifer nichts über frühere oder zukünftige Sitzungen erfahren. Die Offenlegung eines langfristigen Host-Schlüssels ermöglicht es dem Angreifer jedoch immer noch, sich als Server auszugeben und zukünftige Sitzungen zu kompromittieren.

Im Allgemeinen ist es bei SSH schwierig, dass PFS schief geht. Wenn die privaten DH-Parameter für den Client oder den Server offengelegt werden, wird auch der Sitzungsschlüssel offengelegt, aber diese Elemente werden in SSH v2 regelmäßig und direkt nach Abschluss des Schlüsselaustauschs weggeworfen. Es ist möglich, einen Server so zu ändern, dass er eine schwächere Gruppe (z. B. die 1024-Bit-Diffie-Hellman-Gruppe1-sha1) vor einer stärkeren Gruppe (z. B. die 2048-Bit-Diffie-Hellman-Gruppe14-sha1) bevorzugt, aber dazu müsste eine Voreinstellung außer Kraft gesetzt werden.

#### 4.2.2 Anwendung im Alltag

Secure Shell (SSH) wird für die sichere Datenkommunikation, die Fernanmeldung an der Befehlszeile, die Fernausführung von Befehlen und andere sichere Netzwerkdienste zwischen zwei vernetzten Computern verwendet. Es wird häufig als Ersatz für Telnet und andere unsichere Remote-Shell-Protokolle verwendet.

---

<sup>31</sup> <http://tools.ietf.org/html/rfc4419>

## 5 Abschluss

Perfect Forward Security bietet die Sicherheit vergangener Sitzungen, selbst wenn langfristige Identitätsschlüssel ermittelt wurde. Wen nach der Beendigung einer Sitzung alle Beteiligten die privaten Schlüssel vernichtet haben, ist es rechnerisch schwierig, die Sitzung wiederherzustellen, deswegen der "perfekte" Teil von Perfect Forward Security. Auch wenn PFS zweifellos eine Verbesserung ist, so hat sie doch ihren Preis. PFS erfordert eine DH-Berechnung, zusammen mit der Berechnung mit dem RSA- oder (EC)DSA-Algorithmus für den Authentifizierungsmechanismus.

Trotz des erhöhten Rechenaufwands wird PFS von mehreren großen Internetanbietern als wichtiges Sicherheitsmerkmal angesehen. Seit Ende 2011 bietet Google den Nutzern seines Gmail-Dienstes standardmäßig Perfect Forward Security mit TLS an, ebenso wie Google Docs und die verschlüsselte Suche, neben anderen Diensten. Seit November 2013 bietet Twitter den Nutzern seines Dienstes Forward Security mit TLS an. Im Juli 2014 unterstützten 51,3 % der TLS-fähigen Websites einige der Chiffriersuiten, die Forward Security bieten.<sup>32</sup>

Die Unmöglichkeit, die Sitzung wiederherzustellen, hängt damit zusammen, dass es derzeit keinen effektiven Algorithmus gibt, um Faktorisierungs- oder diskrete logarithmische Probleme mit moderner Rechenleistung zu lösen. PFS kann sich jedoch nicht gegen eine erfolgreiche Kryptoanalyse der zugrunde liegenden Chiffren oder des (EC)DH-Austauschs wehren. Insbesondere der Shor's-Algorithmus<sup>33</sup> in Verbindung mit der potenziellen Leistung zukünftiger Quantencomputer<sup>34</sup> ist in der Lage, das Faktor- und das diskrete Logarithmusproblem<sup>35</sup> zu lösen und gleichzeitig die effektive Schlüsselstärke symmetrischer Chiffren zu halbieren. Kryptographen arbeiten an neuen kryptographischen Algorithmen der Post-Quanten-Ära - Ring-LWE<sup>36</sup> und Supersingular Isogeny Diffie-Hellman Key Exchange,<sup>37</sup> die die Vorwärtssicherheit unterstützen können. Auch wenn die Quanteninformatik eine Bedrohung für klassische kryptografische Algorithmen darstellen kann, benötigen sowohl die Quanteninformatik als auch die Post-Quanten-Kryptografie mit PFS-Unterstützung Zeit, um ausgereift zu sein. Bis dahin wird PFS, wenn es richtig implementiert ist, eine zusätzliche Ebene der Privatsphäre für die elektronische Kommunikation bieten.

---

<sup>32</sup> <https://www.trustworthyinternet.org/ssl-pulse/>

<sup>33</sup> [https://en.wikipedia.org/wiki/Shor's\\_algorithm](https://en.wikipedia.org/wiki/Shor's_algorithm)

<sup>34</sup> [http://en.wikipedia.org/wiki/Quantum\\_computer](http://en.wikipedia.org/wiki/Quantum_computer)

<sup>35</sup> [http://en.wikipedia.org/wiki/Factoring\\_problem](http://en.wikipedia.org/wiki/Factoring_problem)

<sup>36</sup> [http://en.wikipedia.org/wiki/Ideal\\_lattice\\_cryptography#Ring-LWE](http://en.wikipedia.org/wiki/Ideal_lattice_cryptography#Ring-LWE)

<sup>37</sup> [http://en.wikipedia.org/wiki/Supersingular\\_Isogeny\\_Key\\_Exchange](http://en.wikipedia.org/wiki/Supersingular_Isogeny_Key_Exchange)

## **Selbständigkeitserklärung**

Hiermit erkläre ich, dass ich die vorliegende Arbeit ohne fremde Hilfe angefertigt und nur die in den Literaturverzeichnissen angeführten Quellen und Hilfsmittel verwendet habe.

Ort, Datum

Unterschrift