16.10.2018

Seminar IT-Sicherheit

How to Backdoor Diffie-Hellman

Sommersemester 2018

Mojdeh Tootchi Fatidehi
Master Informatik
inf102666@fh-wedel.de

Dozent:
Prof. Dr. Gerd Beuster
gb@fh-wedel.de

Inhaltsverzeichnis

1 Einführung

Diffie-Hellman ist ein asymmetrisches Verfahren in der Kryptographie, das auf Basis diskreter Logarithmen und der Restklassengruppen modulo p (Primzahl) aufgebaut ist. Schwierigkeiten in der Berechnung der Umkehrfunktion des diskreten Logarithmus bestätigen, dass der Diffie-Hellman-Schlüsselaustausch sicher ist. Diffie-Hellman wird sicherer, wenn die Primzahl groß genug und geeignet ist. In der Praxis kann man nicht einfach die geeignete große Primzahl finden.

Im Jahre 2016 meldete der Entwickler des Tools Socat eine Sicherheitslücke [6]. Der Modulus, der prim sein muss, war nicht prim. Das Protokoll ist dann natürlich nicht mehr sicher und es besteht eine Angriffsgefahr. Hierin liegt eine grosse Herausforderung. Die Diskussion um Parameter mit backdoor wird immer wieder geführt.

Ebenfalls im Jahre 2016 veröffentlichte David Wong von der Firma NCC einen Artikel, in dem beschrieben wird, wie sich eine backdoor in Parametern für Diffie-Hellman verstecken lässt [7]. Er hat gezeigt, dass die Parameter von Diffie-Hellman eine NOBUS-backdoor (Nobody But Us) erzeugen.

In dieser Seminararbeit werde ich erst zeigen, wie das Diffie-Hellman-Verfahren funktioniert, welche Parameter es zur Verfügung stellt und seinen Zusammenhang mit diskreten Logarithmen aufzeigen. Es sollen ebenso manche Angriffe auf Diffie-Hellman dargestellt werden und gezeigt werden, wie man eine backdoor für Diffie-Hellman erzeugen kann, um einen Angriff abzuwehren. Eine backdoor hat zum Zweck, dass nicht jeder die Veschlüsselung brechen kann. Nur derjenige, der die Parameter erstellt hat, soll mit diesem exklusiven Wissen in der Lage sein, die Verschlüsselung anzugreifen.

2 Diffie-Hellman-Verfahren

2.1 Diffie-Hellman (DH)

Diffie-Hellman ist ein asymmetrisches Verfahren in der Kryptographie, in dem sich zwei oder mehrere Kommunikationspartner auf ein gemeinsames Geheimnis *key k* einigen können und dieses nicht in der Kommunikationsleitung übertragen wird. In diesem Verfahren verfügt jeder Kommunikationspartner über einen *private key* und einen *public key*. Jeder public key wird mit Hilfe des jeweiligen private key, einer großen Primzahl p und einem Generator g ($g < p$) der multiplikativen Gruppe berechnet. Hier wird der public key an die anderen Kommunikationspartner geschickt.

2.2 Diffie-Hellman-Schlüsselaustausch

Das Diffie-Hellman Verfahren ist so: Alice und Bob beabsichtigen sich auf einen Schlüssel k zu verständigen. Aber k wird nicht über die Kommunikationsleitung übertragen. Alice und Bob einigen sich auf eine Primzahl p und einen Generator g mod p kleiner als p, die public sein können.

Alice

- wählt eine geheime zufällige Zahl a kleiner $p-2$ als ihren private key.
- berechnet ihren public key:

$$A = g^a \bmod p$$

- schickt das Ergebnis an Bob.

Bob

- wählt eine geheime zufallige Zahl b kleiner $p-2$ als seinen private key.

- berechnet seinen public key:

$$B = g^b \bmod p$$

- schickt das Ergebnis an Alice.

Alice

- berechnet den k:

$$k = B^a \bmod p$$

Bob

- berechnet den k:

$$k = A^a \bmod p$$

2.3 Beispiel

Sei $p = 17, g = 3$. Alice wählt $a = 7$ und berechnet $g^a \bmod p = 11$ und schickt das Ergebnis an Bob. Bob wählt $b = 4$, berechnet $g^b \bmod p = 13$ und schickt das Ergebnis an Alice. Alice berechnet $k = B^a \bmod p = 4$ und Bob berechnet $k = A^b \bmod p = 4$ [1, S.154].

3 Sicherheit von Diffie-Hellman

Bevor im weiteren Angriffe auf Diffie-Hellman erläutert werden, sollen einige Definitionen und Sätze dargestellt werden:

Definition 3.1 (Die Euler'sche phi-Funktion) $\varphi(n)$, $n > 0$ *ist die Anzahl der Zahlen kleiner als* n*, die zu* n *teilerfremd sind.*

Definition 3.2 (der chinesische Restsatz (CRT)) *Sei* $n > 0$, $n = p_1 \cdot p_2 \cdot \ ... \ \cdot p_i$ *die Faktorisierung von* n *und* $p_1, ..., p_i$ *natürliche Zahlen, die zueinander teilerfremd sind und seien* $a_1, ..., a_i$ *ganze Zahlen.*

$$x \equiv a_1 \ mod \ p_1, \ \ x \equiv a_2 \ mod \ p_2, ..., \ \ x \equiv a_i \ mod \ p_i.$$

Dann kann $x \bmod n$ *berechnet werden.*

Satz 3.1 *Falls* p *eine Primzahl ist, gilt* $\varphi(p) = p - 1$ *und falls* n *sich als Produkt aus zwei verschiedenen Primzahlen* p, q *zusammensetzt, gilt* $\varphi(n) = \varphi(p \cdot q) = (p - 1)(q - 1)$.

Satz 3.2 *"Die Menge aller primen Restklassen modulo* n *bildet eine endliche abelsche Gruppe bezüglich der Multiplikation"[1, S.33].*

Die Gruppe der primen Restklassen modulo n heißt *prime Restklassengruppe* modulo n von der Ordnung $\varphi(n)$.

3.1 Diskreter Logarithmus (DL)

Sei p eine Primzahl. Die prime Restklassengruppe mod p ist eine Gruppe von der Ordnung $\varphi(p) = p - 1$. Sei g eine Primitivwurzel (Generator) mod p. Dann gibt es für jede Zahl $y \in \{1, 2, ..., p-1\}$ einen Exponenten $x \in \{0, 1, 2, ..., p-2\}$ mit

$$y \equiv g^x \bmod p$$

Dieser Exponent x heißt *diskreter Logarithmus* von y zu Basis g. Man schreibt:

$$x = \log_g y$$

Dieser diskrete Logarithmus ist schwer zu berechnen und es gibt bisher keine schnellen und effizienten Algorithmen, die dieses Problem lösen. Jedoch sind Algorithmen bekannt, die mit höherer Laufzeit das Problem lösen.

3.2 Diffie-Hellman-Sicherheit

Durch die unsichere Leitung werden die Zahlen p, g, A und B bekannt. Ein Angreifer wird nicht die diskreten Logarithmen a von A und b von B zur Basis von g kennen. Er muss $K = g^{ab} \bmod p$ berechnen. Das ist das *Diffie Hellman Problem*. Um das Diffie-Hellman-Problem zu lösen, muss man die diskreten Logarithmen mod p berechnen. Wer den diskreten Logarithmus effizient lösen kann, kann auch das Diffie-Hellman-Problem wirkungsvoll berechnen.

4 Angriff auf Diffie-Hellman

Um Diffie-Hellman angreifen zu können, muss man, wie zuvor beschrieben, das Problem der diskreten Logarithmen lösen. Hier sollen nun einige Methoden gezeigt werden, die den diskreten Logarithmus lösen. Jedoch sind sie immer noch nicht effizient.

4.1 Pollard Rho

Pollard Rho ist eine Methode mit einer der geringsten Laufzeiten (aber immer noch nicht schnell). Gesucht ist hier eine Zahl x mit $y = g^x$. Sei G eine Gruppe und $f\colon G \times G \to G \times G$ eine Hilfsfunktion. Die ρ -Methode startet mit einem zufälligen Eingabewert (a_0, b_0) und berechnet rekursiv $(a_{i+1}, b_{i+1}) = f(a_i, b_i)$. Die Methode stoppt mit der Eingabe (a, b) und der Ausgabe (a', b') für welche $g^b y^a = g^{b'} y^{a'}$ gilt:

$$g^b y^a = g^{b'} y^{a'} \pmod{p}$$

Mit $y = g^x$:

$$g^{xa+b} = g^{xa'+b'} \pmod{p}$$

und damit:

$$xa + b \equiv xa' + b' \pmod{p}$$

es folgt:

$$x = (a - a')^{-1}(b - b') \pmod{p}$$

und so wird verschiedene x berechnet und für alle diese Lösung x ausprobiert, ob sich $y = g^x$ ergibt.

4.2 Pohlig-Hellman

Mit Hilfe der Primfaktorisierung der Ordnung einer zyklischen Gruppe mod z können wir das Problem des diskreten Logarithmus auf die Primteiler reduzieren.
Sei $\varphi(z) = |G| = n$ die Gruppenordnung:

$$n = \prod_{p|n} p^{e(p)}$$

die Primfaktorzerlegung von $n = |G|$.
Gegeben seien $y, g \in G$, gesucht ist eine Zahl x mit: $y = g^x$. wir setzen:

$$n_p = \frac{n}{p^{e(p)}}, \;\; g_p = g^{n_p}, \;\; y_p = y^{n_p}$$

Wenn $h \in G$ ein erzeugendes Element von G ist, dann bilden die Elemente $G_p = \{h^{n_p} | h \in G\}$ eine Untergruppe von G.

Satz 4.1 *Für alle Primzahlen p mit $p \,|\, n$ sei $x_p \in \mathbb{N}$ mit $y_p = g_p^{x_p}$ gegeben. Wenn $x \equiv x_p \, mod \, p^{e(p)}$ für alle Primzahlen $p \,|\, n$ gilt, dann ist $y = g^x$ [2, S.102].*

Dieser Satz zeigt, dass es ausreichend ist, den diskreten Logarithmus in den Gruppen G_p zu lösen. "Wir vereinfachen das Problem für G_p weiter, indem wir den diskreten Logarithmus auf Gruppen der Ordnung p reduzieren"[2, S.102]. Sei $|G| = p^e$ für eine Primzahl p. Wir wissen dass, $x < p^e$ gelten muss. Dann kann man x in der Form:

$$x = x_0 + x_1 p + ... + x_{e-1} p^{e-1}, \;\; 0 \leq x_i < p, \;\; 0 \leq i \leq e-1$$

schreiben. Wir potenzieren die Gleichung $y = g^x$ mit p^{e-1} :

$$y^{p^{e-1}} \;=\; g^{p^{e-1} x}.$$

Angenommen $x_0, ..., x_{i-1}$ für $0 \leq i$ sei bereits bekannt, dann gilt:

$$g^{x_i p^i + ... + x_{e-1} p^{e-1}} \;=\; y g^{-(x_0 + x_1 p + ... + x_{i-1} p^{i-1})}.$$

Wir potenzieren diese Gleichung mit p^{e-i-1} :

$$(g^{p^{e-1}})^{x_i} \;=\; y_i^{p^{e-i-1}}, \quad 0 \leq i \leq e-1.$$

Das Element $g^{p^{e-1}}$ hat die Ordnung p und die Zahl x_i ergibt sich durch die Lösung des diskreten Logarithmus innerhalb dieser Gruppe mit der Ordnung p.

4.3 Beispiel

Wir lösen:

$$5^x \equiv 3 \bmod 2017.$$

Die Gruppenordnung der primen Restklassengruppe mod 2017 ist:

$$n = 2016 = 2^5 * 3^2 * 7.$$

Es wird $x(2) = x \bmod 2^5, x(3) = x \bmod 3^2, x(7) = x \bmod 7^1$ bestimmt. Es wird für den Primteiler 2 von 2016

$$\begin{aligned} n_p = \frac{n}{p^{e(p)}} = \frac{2016}{2^{e(5)}} &= 3^2 * 7 \\ g_p = g^{n_p} &= 3^{3^2*7} \\ y_p = y^{n_p} &= 5^{3^2*7} \end{aligned}$$

gesetzt.

Wir erhalten $x(2)$ als Lösung von:

$$(5^{3^2*7})^{x(2)} \equiv 3^{3^2*7} \bmod 2017.$$

Es ergibt:

$$500^{x(2)} \equiv 913 \bmod 2017.$$

Wie schreiben $x(2)$ in Form:

$$x(2) = x_0(2) + x_1(2) * 2 + x_2(2) * 2^2 + x_3(2) * 2^3 + x_4(2) * 2^4.$$

$x_i(2)$ sind die Lösung von:

$$(g^{p^{e-1}})^{x_i(2)} \ = \ y_i^{p^{e-i-1}}, \ \ 0 \leq i \leq e-1.$$

Wir erhalten:

$$x_0(2) = 0, \ x_1(2) = 1, \ x_2(2) = 1, \ x_3(2) = 0, \ x_4(2) = 0.$$

Und es ergibt:

$$x(2) = 6.$$

Schließlich erhalten wir auch :

$$x(3) = 4, \ x(7) = 1.$$

Nun berechnen wir x von:

$$x \equiv x_p \bmod p^{e(p)}$$

$$x \equiv 6 \bmod 32, \ \ x \equiv 4 \bmod 9, \ \ x \equiv 1 \bmod 7$$

Mit dem chinesischen Restsatz ist $x = 1030$.

4.4 Kleine Untergruppen-Angriffe

Eine andere Angriffsform auf DH sind die *Small Subgroup Attacks*, die sich, wie der Name schon andeutet, auf die Untergruppen beziehen. Wie bei der Pohlig-Hellman-Methode muss für diesen Angriff eine kleine Untergruppe vorhanden sein. Damit der Angreifer den geheimen Schlüssel finden kann, braucht er die Zahl der möglichen Schlüssel. Je kleiner die Untergruppe ist, desto einfacher ist es diese Zahl zu finden. Diese Methode funktioniert bei den Protokollen nicht, die für jeden Schlüsselaustausch einen neuen public key einrichten. Dieses Verfahren funktioniert so:

$$n = \prod_{p|n} p^{e(p)}$$

- Der Angreifer wählt eine $z = p^{e(p)}$.
- Alice schickt an Bob ihren public key A.
- Der Angreifer bekommt A und er berechnet A^z und schickt es an Bob als A.
- Bob schickt an Alice seinen public key B.
- Der Angreifer bekommt B und er berechnet B^z und schickt es an Alice als B.

Der Angreifer berechnet ohne a (private key von Alice) und b (private key von Bob) zu kennen k:

$$k = (A^z)^a = (B^z)^b$$

k ist in der Untergruppe. Der Angreifer probiert für jedes Element (mögliche Schlüssel) in der Untergruppe die obige Gleichung aus, bis er den geheimen Schlüssel findet. Je kleiner demnach die Untergruppe ist, desto einfacher ist es k zu finden.

5 Backdoor Diffie-Hellman

5.1 Backdoor in den primen Restklassengruppen

Ein einfacher Versuch, um eine backdoor zu kreieren, wäre ungeeignete Parameter auszusuchen, sodass der diskrete Logarithmus lösbar ist. Einen kleinen Modulus zu haben ermöglicht es den diskreten Logarithmus zu berechnen. Diese Idee hat den Vorteil, dass man den diskreten Logarithmus einfacher lösen kann, aber es lässt keine geheimen Schlüssel einfließen, um eine effiziente backdoor möglich zu machen. Außerdem sind sie unpraktisch. Da jeder diese backdoor öffnen kann, bzw. den Schl üssel berechnen kann, ist diese backdoor keine NOBUS-backdoor.

Eine andere Idee wäre einen Generator einer kleineren Untergruppe zu verwenden, so dass Pollard Rho den diskreten Logarithmus am effizientesten lösen kann. Jedoch ist diese backdoor einfach und jeder Angreifer kann sie ohne weiteres Wissen berechnen und es bleibt kein geheimer Schlüssel (keine NOBUS-backdoor) übrig.

Eine andere Idee ist, die Pohlig-Hellman-Methode zu variieren. Die Idee ist einen primen modulo p auszuwählen. Da p prim ist, ist die Gruppenordnung von der primen Restklassengruppe modulo p: $\varphi(p) = p - 1$. Natürlich haben wir die möglichen public keys in dieser Gruppe.

Mit Hilfe der Primfaktorisierung der Ordnung der Gruppe, wie es bei Pohlig-Hellman gesagt wird, können wir das Problem des diskreten Logarithmus auf die Primteiler reduzieren.
Sei $|G| = n = \varphi(p)$ die Gruppenordnung. Wenn diese Zahl faktorisiert wird, dann können wir die möglichen Untergruppen finden und wenn genügende kleine Untergruppen vorhanden sind, ist es möglich mit dem chinesischen Restsatz den richtigen key zu finden. Natürlich ist das Problem hier, dass wenn wir Pohlig-Hellman nutzen, und genügend kleine Untergruppen vorhanden sind, jeder die Faktorisierung und eben den key finden kann. Deswegen ist diese backdoor gescheitert.

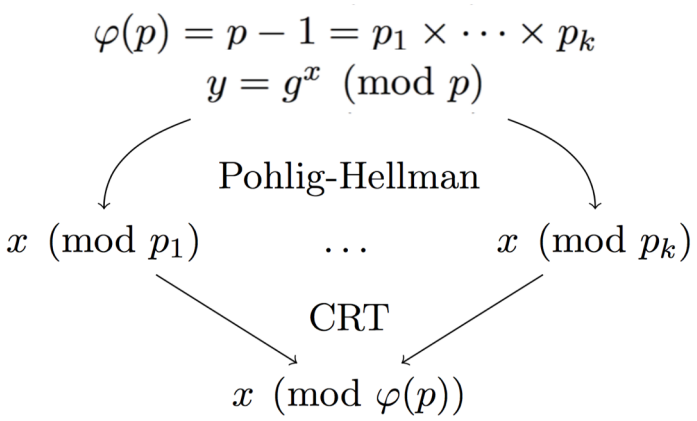


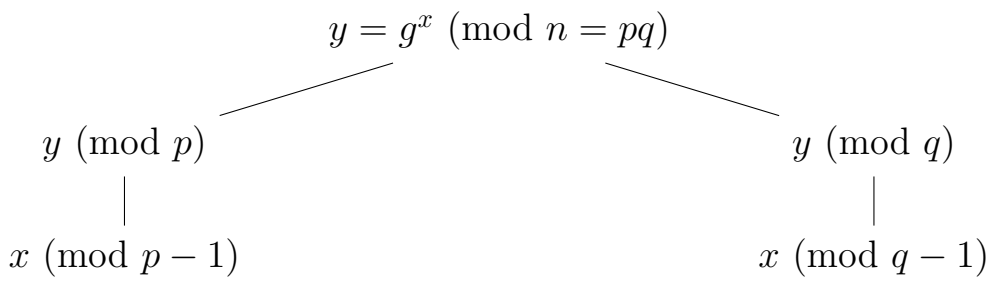
Abbildung 5.1: Backdoor in den primen Restklassengruppen[1]

[1]David Wong: How to Backdoor Diffie-Hellman, NCC Group, June 2016, `https://eprint.iacr.org/2016/644.pdf` S. 7. (Zugriff 20.06.2018)

5.2 Zusammengesetzter Modulus für eine NOBUS Backdoor

Die zweite backdoor baut auf der vorherigen Methode auf. Wir wählen einen modulo $n = pq$ aus, wobei p und q Primzahlen und groß genug sind, sodass eine Faktorisierung von n unterbleibt. Deswegen bleiben die Untergruppen versteckt. Hier ist n keine Primzahl mehr.

Wenn wir die Faktorisierung von n kennen, kann das Problem des diskreten Logarithmus auf modulo p und q reduziert werden und wieder mit Hilfe des chinesischen Restsatzes gelöst werden.



mit chinesischem Restsatz $\Rightarrow x \pmod{(p-1)(q-1)}$.

Hier reduzieren wir das Problem auf modulo p und modulo q. Da p und q Primzahlen sind, ist die Ordnung der primen Restklassen modulo p, $(p-1)$ und die Ordnung der primen Restklassen modulo q, $(q-1)$. Der diskrete Logarithmus wird in diesen Untergruppen gelöst und wir haben die möglichen private keys.
Aber um dieses Problem einfacher zu lösen, suchen wir einen Generator g, sodass g mod p und g mod q kleine Untergruppen erzeugen und uns den diskreten Logarithmus in zwei kleinen Untergruppen statt in einer großen Gruppe berechnen lässt.
Wir wählen p und q so:

$$p-1 \;=\; 2p_1p_2 \text{ und } q-1 \;=\; 2q_1q_2$$

wobei p_1 und q_1 zwei kleine Primfaktoren und p_2 und q_2 zwei große Primfaktoren sind.

Der Satz von Lagrange (nach dem italienischen Mathematiker Joseph-Louis de Lagrange) besagt, dass wenn G eine endliche Gruppe ist, dann teilt die Ordnung der Untergruppe die Ordnung von G.

Wenn g ein Element von G ist, dann ist dessen Ordnung die Ordnung der Untergruppe, die von diesem Element erzeugt wird.
Es heißt hier, dass dieses Element g die Ordnung p_1q_1 verfügt und das ist der gesuchte Generator.

Durch Reduzieren das diskreten Logarithmus $y \;=\; g^x$ mod p und mod q und mit dem neuen, durch die backdoor hergestellten Generator, kann x mod $(p-1)$ und x mod $(q-1)$ einfacher in diesen kleinen Untergruppen berechnet werden. Jetzt sind die möglichen private keys vorhanden und schliesslich wird mit dem chinesischen Restsatz der secret key mod $((p-1)(q-1))$ berechnet.

Da die Untergruppen so gut versteckt sind, dass ein Dritter sie nicht finden kann und nur die Kommunikationspartner von ihnen wissen, bezeichnete man diese backdoor als $NOBUS$. Sie ist jedoch noch nicht effizient, weil eine Angriffsmethode auf diese backdoor entwickelt wurde.

Die nächste backdoor ist die Fortentwicklung der letzten Methode, die NOBUS-Eigenschaften hat, die vorherige nicht mehr. Diese backdoor kann ein Dritter nicht finden. Es heisst wieder $n = pq$ und die Ordnung der Gruppe ist wie gesagt:

$$\varphi(n) = (p-1)(q-1)$$

Aber es werden p und q ausgewält, sodass:

$$p - 1 = p_1 \times ... \times p_k \times 2$$

und

$$q - 1 = q_1 \times ... \times q_l \times 2$$

mit $ggt(p-1, q-1) = 2$

ist.

Es wird ein großer Faktor in beide, $p-1$ und $q-1$ eingefügt und sie heissen jeweils p_{big} und q_{big}.

Wie bei der letzten backdoor wird der public key y auf mod p und q reduziert und mit dem Pohlig-Hellman-Algorithmus werden wieder die möglichen private keys berechnet. Am Ende wird mit Hilfe des chinesischen Restsatzes der private secret key mod $\frac{(p-1)(q-1)}{2}$ berechnet.

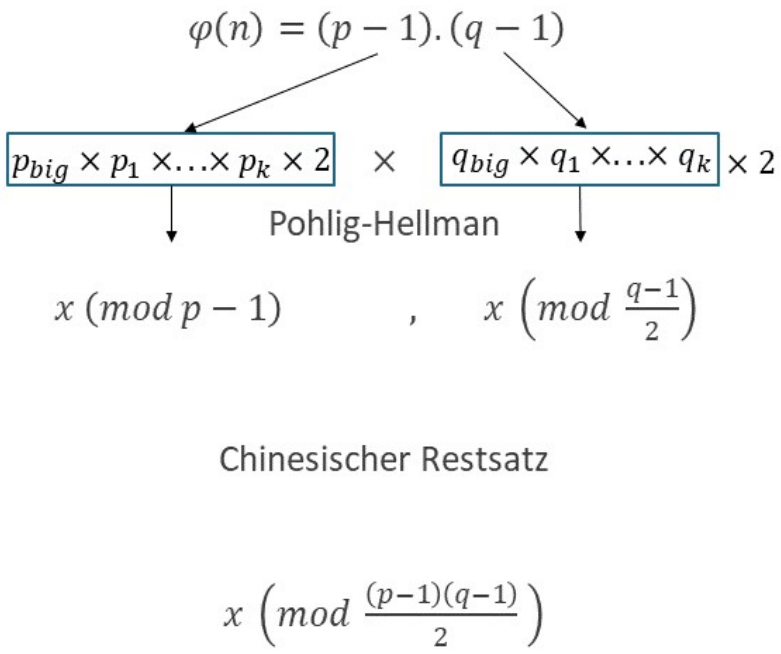


Abbildung 5.2: Zusammengesetzter Modulus für eine NOBUS-backdoor

Diese backdoor hält die Untergruppen so gut versteckt,dass nur derjenige, der die Parameter festgelegt hat, kann den Pohlig-Hellman-Angriff einsetzen. Diese Idee verstärkt die Sicherheit von Diffie-Hellmann, Weil wenn n groß genug ist, ist die Faktorisierung von n schwer. Außerdem sind hier p und q versteckt und solange der Angreifer über p und q nicht Bescheid weiß, kann er $(p-1)(q-1)$ und die Faktorisierung von $((p-1)(q-1)$ mit p_{big} und q_{big} nicht wissen, um die Untergruppen herauszufinden.

6 Fazit

In diesen Zusammenhang wurden viele open source libraries für solche backdoors getestet. Von 4,522,263 in Frage kommenden Diffie-Hellman-Handshakes nutzten nur 30 diesen zusammengesetzten Modulus und obwohl alle diese einen kleinen Faktor hatten, könnte keiner unter 5 Stunden faktorisiert werden. Aber der Prozess ist in einer realen Zeit erfolgreich. Es besteht eine Angriffsgefahr und alle Administratoren wurden über das Problem informiert.

Um einen Angriff zu vermeiden, hilft es natürlich zu prüfen, ob der Prim-Parameter eine geeignete Primzahl ist. Diese Prüfung ist nicht immer einfach. Es gibt Algorithmen, die diesen Test effizient durchführen, aber nicht alle Diffie-Hellman-Implementationen prüfen, ob es sich um eine gute und geeignete eingesetzte Primzahl handelt.

Nun kennen wir das Problem und wir können uns mit Auswahl der richtigen Parameter dagegen verteidigen.

Abbildungsverzeichnis

Literaturverzeichnis

[1] Johannes Buchmann: Einführung in die Kryptographie. 4., erweiterte Auflage. Springer 2008

[2] Volker Diekert, Manfried Kufleitner, Gerhard Rosenberger: Diskrete Algebraische Methoden. Arithmetik, Kryptographie, Automaten und Gruppen. Studium 2013

[3] Stephan Spritz, Michael Pramateftakis, Joachim Swoboda: Kryptographie und IT-Sicherheit. Grundlage und Anwendungen. 2.,überarbeitete AuflageStudium. Studium 2011.

[4] Dr. Annette Werner: Elliptische Kurven in der Kryptographie. Springer 200. S. 77 fff.

[5] Martin Bartosch: Gute Zahlen, Schlechte Zahlen. 27.05.2008. `https://www.heise.de/security/artikel/Gute-Zahlen-schlechte-Zahlen-270078.html`(Zugriff 20.06.2018)

[6] Hanno Böck: Eine Backdoor für Diffie-Hellman. 8.August.2016. `https://www.golem.de/news/schluesselaustausch-eine-backdoor-fuer-diffie-hellman-1608-122552.html` (Zugriff 20.06.2018)

[7] Hanno Böck: Primzahlen knnen Hintertr enthalten. 12.Oktober.2016. `https://www.golem.de/news/dsa-diffie-hellman-primzahlen-koennen-hintertuer-enthalten-1610-12.html`(Zugriff 20.06.2018)

[8] Fabian A. Scherschel: Verflixte Primzahlen: Eine subtile Hintertr im Diffie-Hellman-Schlsselaustausch. 10.08.2016. https://www.heise.de/security/meldung/Verflixte-Primzahlen-Eine-subtile-Hintertuer-im-Diffie-Hellman-Sch
html(Zugriff 20.06.2018)

[9] David Wong: How to Backdoor Diffie-Hellman. NCC Group June 2016. https://eprint.iacr.org/2016/644.pdf

[10] David Wong: How to Backdoor Diffie-Hellman.quick explanation. August 2016. https://www.cryptologie.net/article/360/how-to-backdoor-diffie-hellman-quick-explanation/ (Zugriff 20.06.2018)