Seminar IT-Sicherheit
Sommersemester 2018

Post quantum key exchange - a new hope

Quantumresistenter Schlüsselaustausch mit ring learning with errors

Eingereicht von:
Andreas Hirschberger
its103191@fh-wedel.de

Betreuer:
Prof. Dr. Gerd Beuster

1 Einführung

Polynomial-Zeit Algorithmen auf Quantencomputern bedrohen die freie und geheime Kommunikation. Die auf der Schwere der Primfaktorzerlegung basierenden Verfahren werden nach Auffassung einiger Experten in wenigen Jahrzehnten keinen kryptographischen Wert mehr haben. Durch Massenspeicherung verschlüsselter Kommunikation wird es dann auch möglich sein rückwirkend Nachrichten zu entschlüsseln die unter der Annahme der Vertraulichkeit übermittelt wurden.

Angesichts dieser Bedrohung hat das *National Institute of Standards and Technology* (NIST) im Jahr 2016 ein Projekt zur Standardisierung Post-Quantum-Kryptographischer Verfahren ins Leben gerufen[1]. Im Winter 2016 startete der *call for submissions* und im November 2017 endete die Einreichfrist. Kryptographen und Mathematiker aus aller Welt reichten ihre Entwürfe ein von denen 69 über die nächsten Jahre evaluiert und diskutiert, angegriffen und verbessert werden. Eine dieser Vorschläge ist New Hope, ein auf dem *ring learning with errors* Problem basiertes Schlüsselaustauschprotokoll, das sich besonders durch die Performanz gegenüber anderen Post-Quantum Methoden abhebt.

New Hope wurde von Erdem Alkim, Léo Ducas, Thomas Pöppelmann und Peter Schwabe[2] in ihrem 2015 veröffentlichten Paper [Alkim et al., 2015] als eine Weiterentwicklung einer Referenzimplementierung eines Protokolls auf ähnlichen mathematischen Grundlagen vorgestellt. Es gelang den Autoren dabei das durch Bos, Costello, Naehring und Stebila [Bos et al., 2014] präsentierte Verfahren erheblichen in Bezug auf Performanz und Sicherheit zu verbessern. In ihrem Paper führen sie ihre Überlegungen aus und argumentieren ihre Entscheidungen ausführlich.

Dieses Dokument soll eine Übersicht über die Hintergründe der Quantum-Unsicherheit in der Kryptographie bieten und die von den Autoren angebrachten Argumente für spezifische Parameter ihres Verfahrens einfach zusammenfassen.

[1] https://csrc.nist.gov/Projects/Post-Quantum-Cryptography
[2] Im Folgenden „die Autoren“ genannt

2 Quanten-Unsicherheit

Quantencomputer mögen sich immer noch nach Science-Fiction anfühlen und doch schreitet die Forschung auf diesem Gebiet langsam aber stetig voran, bemüht das theoretische Konzept zu einer Wirklichkeit zu machen. Quantencomputer bieten dabei faszinierende Möglichkeiten, insbesondere würden Such- und Optimierungsaufgaben mit erheblich höherer Effizienz berechenbar. Doch genau diese Entwicklung betrachten Sicherheitsexperten mit Sorge, lassen sich doch einige der Probleme auf denen moderne Kryptographie basiert auf eben solche Optimierungsprobleme zurückführen.

Die klassische Kryptographie[3] lässt sich für diese Betrachtung in zwei Klassen gliedern. Zum einen gibt es **symmetrische Verfahren**, bei denen der gleiche Schlüssel zum ver- und entschlüsseln benutzt wird und der allen Kommunikationspartnern bekannt sein muss. AES [NIST, 2001] der *Advanced Encryption Standard* ist dabei das am häufigsten genutzte symmetrische Verfahren.

Zum anderen gibt es **asymmetrische Verfahren**, die auf dem Geheimhalten einer Information, dem *private key*, und dem veröffentlichen eines Komplementär-Schlüssels, dem *public key* basiert. RSA [Rivest et al., 1978] und Diffie-Hellman [Diffie and Hellman, 1976] sind hier am häufigsten zu finden.

Für die Echtzeitverschlüsselung nach z.B. TLS wird eine Kombination aus beiden Klassen genutzt. Authentisierung und Schlüsselaustausch geschehen mit asymmetrischen Verfahren – die zumeist erheblich Langsamer sind – ein durch den so erhaltenen gemeinsamen Schlüssel aufgebauter Kanal ist dann durch ein effizienteres symmetrisches Verfahren verschlüsselt.

Asymmetrische Verfahren basieren dabei fast ausschließlich auf der Schwierigkeit der Primfaktorzerlegung großer Zahlen. Ein Problem von dem angenommen wird, dass es keine effiziente Lösung gibt, auch wenn ein formaler Beweis hierfür noch nicht erbracht werden konnte. Die besonderen Eigenschaften des Rechnens mit QuBits jedoch erlauben eine neue Klasse von Algorithmen die die klassische Kryptographie schwächen und im Falle der asymmetrischen Verfahren sogar brechen könnten. Im folgenden werden zwei dieser Algorithmen kurz vorgestellt.

2.1 Grovers Algorithmus

Der von Grover 1996 [Grover, 1996] veröffentlichte Algorithmus erlaubt die Suche in einem unsortierten Datensatz mit N Einträgen in $O(\sqrt{N})$ Schritten. Dies ermöglicht eine effizientere Lösung von NP-Vollständigen Problemen. Da die Suche nach z.B. einem Schlüssel eines symmetrischen Verfahrens in diese Klasse fällt werden mit Grovers Algorithmus auf einem ausreichend starken Quantencomputer *brute force* Angriffe auf symmetrische Verschlüsselung mit Aufwand $\sqrt{N}$ möglich.

[3] Es gibt bereits Ansätze Kryptographie mittels Quantencomputer umzusetzen was gänzlich andere Kryptographische Konzepte ermöglicht.

2.2 Shors Algorithmus

Der von Shor 1995 [Shor, 1995] vorgestellte Quantencomputer-Algorithmus ermöglicht die Primfaktorzerlegung einer Natürlichen Zahl in polynomieller Laufzeit. Zum Zerlegen einer Zahl n wird dabei ein Quantencomputer mit mindestens $\log n$ QuBits benötigt. Zur Zeit liegt der Rekord bei der Faktorisierung mit Shors Algorithmus bei $21 = 3 \cdot 7$ [Martin-Lopez et al., 2011]. Dennoch ist mit rascher Entwicklung in den nächsten Jahren zu rechnen, nicht nur technisch sondern auch in Algorithmen. Es existiert auch eine Variante von Shors Algorithmus zum Lösen des diskreten Logarithmus.

2.3 Was bedeutet das für die Kryptographie?

Während Grovers Algorithmus die Sicherheitsbits eines symmetrischen Verschlüsselungsverfahrens halbiert, bricht Shors Algorithmus die asymmetrischen Verfahren gänzlich. Experten rechnen mit einem ausreichend starken Quantencomputer zum Jahr 2030. Dieser wird dann auch rückwirkend die Entschlüsselung gespeicherter Kommunikation ermöglichen. Angesichts der zunehmenden Datensammlung durch Geheimdienste[4] ist es im Sinne eines freien Internets baldmöglichst auf Methoden umzusteigen die Robust gegen Quantenalgorithmen und auf konventioneller Hardware lauffähig sind.

Für einen erheblichen Teil der verschlüsselten Kommunikation ist dabei Performanz oft wichtiger als Vertrautheit. Viel der alltäglichen Kommunikation ist für die Kommunikationspartner nur kurzzeitig Sicherheitsrelevant und zusätzliche Latenz oder Serverlast kann unter Umständen für einen Betreiber nicht hinnehmbar sein. Da Patchwork Lösungen nicht wünschenswert sind, ist eine allgemein akzeptable Standardisierung erforderlich. Ein neues Verfahren wird sich daher nicht nur als Sicherheitsrelevant beweisen lassen müssen, sondern auch als unerheblicher zusätzlicher Aufwand bei einer eventuellen Umstellung.

[4] Allein das 2013 eröffnete Rechenzentrum der NSA in Utah speichert bis zu 20TB an Internetkommunikation pro Minute.

3 Schlüsselaustausch

Wie bereits behandelt sind symmetrische Verfahren wie AES auch mit rechenstarken Quantencomputern noch sicher[5]. Allerdings sind diese gleichzeitig auf das Geheimhalten des Schlüssels und das Verbreiten des Schlüssels an Entschlüsselungsberechtigte angewiesen.

Im Falle der Echtzeit-Kommunikation über einen öffentlichen, unsicheren Kanal bedeutet dies, ein gemeinsames Geheimnis zu etablieren, ohne Zuhörern Informationen über den später zu verwendenden Schlüssel zukommen zu lassen. Das Erzeugen eines gemeinsamen Geheimnisses als Schlüssel für ein symmetrisches Verfahren nennt man Key Exchange oder Schlüsselaustausch. Verschiedene asymmetrische Public-Key Methoden sind verfügbar, wobei das Schlüsselaustauschprotokoll nach Diffie-Hellman, in verschiedenen moderneren Varianten[6], dabei heute Standard ist[7].

Der Ablauf[8] eines Schlüsselaustauschs nach dem Diffie-Hellman Protokolls ist in der ursprünglichen Formulierung dabei recht einfach:

Alice möchte einen Schlüssel mit Bob austauschen um einen sicheren Kanal aufzubauen. Hierzu wählt sie zunächst ein geheimes a und sendet eine Nachricht mit Inhalt $A = g^a \mod p$ an Bob, wobei p eine große Primzahl und g eine Primitivwurzel modulo p ist (beide Werte sind öffentlich und bekannt).

Bob seinerseits wählt ein geheimes b und sendet $B = g^b \mod p$ an Alice.

Alice berechnet nun s aus B mit $s = B^a \mod p$ und

Bob berechnet $s = A^b \mod p$.

Beide erhalten damit das gleiche gemeinsame Geheimnis $s = g^{ab} \mod p$.

Die Berechnung von a oder b aus A oder B ist für einen Zuhörer Eve nicht effizient machbar und s lässt sich somit nicht aus A und B berechnen ohne Kenntnis des jeweils Gegenseitigen a oder b.

Alice		Bob
choose a		
	— $A = g^a \mod p$ —>	
		choose b
	<— $B = g^b \mod p$ —	
$s = B^a \mod p$		$s = A^b \mod p$

Tabelle 1: Schlüsselaustausch nach Diffie-Hellman

[5] Die Halbierung der Sicherheitsbits lässt sich im Zweifelsfall durch Erhöhung der Schlüsselgröße ausgleichen. Verfahren wie AES mit 256 Bit bleiben sicher genug.

[6] insbesondere die ephemerale Variante von Diffie-Hellman auf elliptischen Kurven

[7] Das klassische DH ist kein akzeptiertes Verfahren mehr in TLS 1.3

[8] Hier ohne Authentisierung. In der Praxis würde DH in Verbindung mit z.B. RSA eingesetzt.

Da dieses Schlüsselaustausch-Verfahren auf dem diskreten Logarithmus Problem basiert ist es mit Shors Algorithmus auf einem ausreichend starken Quantencomputer als gebrochen zu betrachten. Andere Varianten von Diffie-Hellman basieren auf ähnlichen mathematischen Problemen, die zwar konventionelle Angriffe besser widerstehen, jedoch keine nennenswert höhere Resistenz gegen quantencomputerbasierte Angriffe vorweisen können.

4 New Hope

Mit New Hope [Alkim et al., 2017] haben Alkim, Avanzi, Bos, Ducas, de la Piedra, Pöppelmann, Schwabe und Stebila im Nov 2017 dem NIST eine Spezifikation für einen neuen, und vor allem quantenresistenten, Schlüsselaustauschverfahren vorgelegt, das im Folgenden vorgestellt werden soll[9]. Das Protokoll ist dabei besonders auf Performanz und Effizienz ausgerichtet und bietet dennoch errechnete 233 Bit Sicherheit.

Der Nachrichtenaustausch in New Hope erinnert stark an DH, basiert jedoch auf gänzlich anderen mathematischen Problemen: namentlich *ring learning with errors*. Der Ablauf ist auch hier einfach: Die Mathematische Struktur ist ein Polynomring mit Ordnung $n = 1024$ und modulo $q = 12289$. Ψ_{16} ist eine Funktion die *kleine*[10] zufällige Polynome erzeugt.

Alice wählt einen zufälligen 256 Bit *seed* und erzeugt mit diesem ein Generatorpolynom a. Sie zieht aus Ψ_{16} ein Geheimnis s sowie einen kleinen Fehler e und sendet $b = as + e$ sowie den *seed* an Bob.

Bob erzeugt mit dem erhaltenen *seed* das Generatorpolynom a. Bob zieht dann seinerseits ein geheimes s' sowie zwei Fehlerpolynome e' und e'' aus Ψ_{16}. Er berechnet $u = as' + e'$ und $v = bs' + e''$ sowie einen Korrekturhinweis r und sendet u und r an Alice.

Alice berechnet nun $v' = us$ und korrigiert das Ergebnis unter Verwendung von r zu ν während

Bob sein v zu ν korrigiert.

Die Ringelemente $v = us = as's + e's$ und $v' = bs' + e'' = as's + e'se''$ werden Maßgeblich von $as's$ bestimmt, unterscheiden sich jedoch in den verhältnismäßig kleinen Fehlern $e's$ bzw. $es' + e''$. Dieser Fehler kann bei der Abbildung des Polynoms auf das 256 Bit Geheimnis ν mit sehr hoher Wahrscheinlichkeit ausgeglichen werden. ν wird dann nochmal zu einem 256 Bit Schlüssel μ gehashed.

[9] Beschreibungen des Protokolls basieren überwiegend auf [Alkim et al., 2015]
[10] Klein gemessen an der Supremumsnorm. d.h. die Koeffizienten sind kleine Integer.

Alice		Bob
seed $\leftarrow \{1,0\}^{256}$		
a $\leftarrow$ Parse(SHAKE-128(seed))		
s, e $\leftarrow \Psi_{16}^n$		s', e', e" $\leftarrow \Psi_{16}^n$
b $\leftarrow$ as + e	– (b,seed) –>	a $\leftarrow$ Parse(SHAKE-128(seed))
		u $\leftarrow$ as' + e'
		v $\leftarrow$ bs' + e"
v' $\leftarrow$ us	<– (u,r) ——	r $\leftarrow$ HelpRec(v)
$\nu \leftarrow$ Rec(v',r)		$\nu \leftarrow$ Rec(v,r)
$\mu \leftarrow$ SHA3-256(ν)		$\mu \leftarrow$ SHA3-256(ν)

Tabelle 2: Schlüsselaustausch nach New Hope

Auf Details zum Sampler Ψ_{16} sowie zu den Funktion 'Parse()', 'HelpRec()' und 'Rec()' wird an geeigneter Stelle eingegangen.

New Hope basiert auf dem *ring learning with errors* Problem nach dem es nicht effizient möglich ist s aus b oder s' aus u zu berechnen. Auch ist es nicht möglich ohne Kenntnis des jeweils gegenseitigen s bzw. s' das gemeinsame Geheimnis ν zu berechnen.

Aufgrund der Ähnlichkeit zu Diffie-Hellman wurden Vorgänger-Verfahren auch als *Diffie-Hellman with noise* bezeichnet.

Zur Illustration soll New-Hope an einem vereinfachten Beispiel demonstriert werden. Die Ordnung wird zu $n = 2$ reduziert und der modulo auf $q = 53$. Es soll ein Schlüssel-Bit ausgetauscht werden das in die beiden verfügbaren Koeffizienten eingebettet ist. Das Generatorpolynom wird als $a = 11x + 32$ festgelegt.

Alice generiert mit dem Sampler Ψ das Geheimnis $s = 1$ und das Fehlerpolynom $e = 51x$. Sie berechnet ihren öffentlichen Schlüssel $b = as + e = (11x + 32) \cdot (1) + (51x) = 9x + 32$ und sendet b an Bob.

Bob erzeugt sein Geheimnis $s' = x$ und die Fehlerpolynome $e' = x + 51$ und $e'' = x$. Er berechnet seinen öffentlichen Schlüssel $u = as' + e' = (11x + 32) \cdot x + (x + 51) = 33x + 40$, das Ringelement $v = bs' + e'' = (9x + 32) \cdot x + (x) = 33x + 44$ und den Korrekturhinweis[11]

$$\vec{r} = \begin{pmatrix} -0,12 \\ -0,33 \end{pmatrix}$$

. Bob sendet u und r an Alice.

Alice berechnet $v' = us = (33x + 40) \cdot (1) = 33x + 40$ und berechnet unter Verwendung des Korrekturhinweises $\nu = 1$.

Bob korrigiert seinerseits v zu $\nu = 1$

[11] Berechnung des Korrekturhinweises wird in 4.4.5 erläutert.

Beide kennen nun das gemeinsame Schlüssel-Bit $\nu = 1$. Ein Zuhörer kann dieses nicht ohne Kenntnis von s oder s' berechnen.

4.1 Ring learning with errors

Ring learning with errors[12] ist die Übertragung des *learning with errors* Problems, einem Problem aus dem Maschinellen Lernen, auf die mathematische Struktur der Polynomringe. Es gibt keine bekannten Angriffe auf *rlwe* die auf der Ringstruktur basieren, so dass im Allgemeinen die Härte-Beweise des *learning with errors* Problems als auch für das *ring learning with errors* Problem geltend betrachtet werden.

Ring learning with errors existiert in zwei Formulierungen:

- Es ist nicht möglich ein zufälliges Ringelement r von einem solchen zu Unterscheiden das als $as + e$ berechnet wurde[13]
- Es ist nicht effizient möglich s aus dem Tupel (a, r) mit $r = as + e$ zu berechnen.

Es wurde bewiesen [Lyubashevsky et al., 2010], dass *ring learning with errors* eine *Schwierigkeit* aufweist die einer *worst case* Lösung des *shortest vector* Problems in einem idealen Gitter gleicht. SVP in einem lattice sind unter bestimmten Umständen NP-Hart, es ist jedoch nicht sicher, dass dies auch für ideale Gitter gilt.

Ein erheblicher Vorteil der Verwendung von Polynomringen ist die Einfachheit der zugrundeliegenden Rechenoperationen[14]. Vektor- und Matrixadditionen mit verhältnismäßig kleinen Zahlen reichen im Allgemeinen aus.

Einem *ring learning with errors* basierten Verfahren stehen dabei die folgenden Parameter zur Verfügung:

- Grad n des Polynomrings
- Koeffizienten-Modulo q
- Definition der Fehlerfunktion χ[15]

4.2 Peikerts Key Encapsulation Mechanism

New Hope ist eine Instanz des von Peikert vorgestellten *key encapsulation mechanism*. Der Mechanismus beschreibt die 4 Algorithmen Setup, Gen, Encaps und Decaps. Nach erfolgreicher Anwendung des folgenden Protokolls teilen sich Alice und Bob einen gemeinsamen ephemeralen Schlüssel μ.

Alice	Bob
Setup():	

[12] Eigentlich *learning with errors over rings*
[13] a ist ein Generatorpolynom, s und e sind kleine Polynome.
[14] Bei geschickter Wahl des Polynomrings
[15] in New Hope ist das die vereinfachte Fehlerfunktion Ψ_k

Alice		Bob
$a \leftarrow R_q$		
Gen(a):		Encaps(a,b):
s, e $\leftarrow \chi$		s', e', e" $\leftarrow \chi$
b $\leftarrow$ as + e	– b –>	u $\leftarrow$ as' + e'
		v $\leftarrow$ bs' + e"
		$\bar{v} \leftarrow$ dbl(v)
Decaps(s, (u,v')):	<– u, v' –	v' = $\langle \bar{v} \rangle_2$
$\mu \leftarrow$ rec(2us, v')		$\mu \leftarrow \lfloor \bar{v} \rceil_2$

Tabelle 3: Peikert's key encapsulation mechanism

4.3 Bos et al. Proposal

Bos, Costello, Naehrig, und Stebila demonstrierten die Praktikabilität eines Schlüsselaustauschs mit *Peikert's Key Encapsulation Mechanism* indem sie es erfolgreich in OpenSSL integrierten[Bos et al., 2014]. Hierzu wählten sie die Konkreten Parameter

- $n = 1024$
- $q = 2^{32} - 1$
- χ = Normalverteilung $D_{Z,\sigma}$ mit Standardabweichung $\sigma = 8/\sqrt{2\pi}$

4.4 Parameter und Ideen von New Hope

Als *sofortiger* Ersatz für Diffie-Hellman ist die oben beschriebene Implementierung jedoch nicht performant genug. Die Autoren nennen zwei primäre Gründe für diese Ineffizienz; Zum einen der große Modulo von $q = 2^{32} - 1$ zum Anderen halten sie die Verwendung einer diskreten Normalverteilung für den *Sampler* χ für überdimensioniert. Auch ist die Nachrichtengröße mit 32kBit größer als Notwendig.

4.4.1 Auswahl des Modulo

Die Autoren wählen den Modulo q des Polynomrings mit $q = 12289$ als die kleinste Primzahl für die gilt $q \equiv 1 \mod 2n$ was eine effizient umsetzbare *number-theoretic transform* (NTT) und damit ein effizientes Rechnen mit den Polynomen ermöglicht. Zudem steigt die Sicherheit des *ring learning with errors* mit dem Verhältnis des Fehlers zum Modulo, sodass gleichzeitig eine Verkleinerung der Nachrichten und eine Sicherheitsverbesserung erreicht werden können.

Ein Polynom p hat dann die Form:

$$p = p_0 + p_1 x + p_2 x^2 + .. + p_{1023} x^{1023}$$

Mit $q = 12289$ lässt sich jeder Koeffizient mit $\log_2 q \approx 14$ Bit darstellen, so dass sich eine *public key* Größe von $n \cdot \ log_2 q = 14336$ Bit oder 1792 Byte ergibt.

4.4.2 NTT

Die *number-theoretic transform* ist ein, in Implementierungen von lattice Kryptographie, häufig eingesetztes Werkzeug welches eine schnelle Multiplikation von Polynomen ermöglicht mit:

$$ab = NTT^{-1}(NTT(a) \circ NTT(b))$$

NTT ist eine Variante der Diskreten Fourier Transformation mit endlichen Körpern. Es existieren semi-logarithmische Algorithmen für NTT solange der Polynomring einen Grad n hat der eine Zweierpotenz ist und für den modulo gilt $q \equiv 1 \mod 2n$.

In New Hope sind n und q genau so gewählt und somit lässt sich der folgende Algorithmus zur Transformation eines Polynoms $g = \sum_{i=0}^{1023} g_i X^i$ verwenden:

$$NTT(g) = \hat{g} = \sum_{i=0}^{1023} \hat{g}_i X^i, \text{ wobei}$$
$$\hat{g}_i = \sum_{j=0}^{1023} \gamma^j g_j \omega^{ij}$$

ω ist auf 49 festgelegt und $\gamma = \sqrt{\omega} = 7$.

Alice		Bob
$seed \leftarrow \{1,0\}^{256}$		
$\hat{a} \leftarrow$ Parse(SHAKE-128($seed$))		
s, e $\leftarrow \Psi_{16}^n$		s', e', e" $\leftarrow \Psi_{16}^n$
$\hat{s} \leftarrow$ NTT(s)		
$\hat{b} \leftarrow \hat{a} \circ \hat{s} +$ NTT(e)		
$m_a =$ encodeA($seed$,$\hat{b}$)	– m_a –>	($\hat{b}$, $seed$) $\leftarrow$ decodeA(m_a)
		$\hat{a} \leftarrow$ Parse(SHAKE-128($seed$))
		$\hat{t} \leftarrow$ NTT(s')
		$\hat{u} \leftarrow \hat{a} \circ \hat{t} +$ NTT(e')
		v $\leftarrow NTT^{-1}(\hat{b} \circ \hat{t}) + e''$
		r $\leftarrow$ HelpRec(v)
($\hat{u}$,r) $\leftarrow$ decodeB(m_b)	<– m_b –	$m_b =$ encodeB($\hat{u}$,r)
v' $\leftarrow NTT^{-1}(\hat{u} \circ \hat{s})$		
$\nu \leftarrow$ Rec(v',r)		$\nu \leftarrow$ Rec(v,r)
$\mu \leftarrow$ SHA3-256(ν)		$\mu \leftarrow$ SHA3-256(ν)

Tabelle 4: Detaillierter Ablauf des Schlüsselaustauschs mit Encoding und NTT

4.4.3 Generatorpolynom

Das Generatorpolynom wird in New Hope im Gegensatz zu anderen *ring learning with errors* Schlüsselaustauschprotokollen nicht festgelegt um *Backdoor-* und *All-for-the-price-of-one* Angriffe von vornerein auszuschließen.

Backdoor Angriffe sind Beispielsweise möglich wenn das Generatorpolynom a aus gewählten kleinen Polynomen f, g berechnet wird, so dass $a = gf^{-1}$. Das Verfahren ist dann immer noch Sicher außer gegen die Instanz mit Kenntnis von f und g. Dieser ist es dann möglich s zu berechnen.

Für **All-for-the-price-of-one Angriffe** sind möglich wenn die Sicherheit auf einem der gewählten Parameter basiert. Mit einem nicht-manipulierten, festgelegten Generatorpolynom könnten sich Schwachstellen finden lassen die nicht auf andere Generatorpolynome übertragbar sind. Es ist daher im Allgemeinen besser die Sicherheit des Verfahrens nicht auf eine konkrete Problemstellung zu basieren.

Aus diesen Gründen wird das Generatorpolynom anhand eines zufällig gewählten *seed* berechnet und nur kurzzeitig gespeichert und wiederverwendet. Konkret wird ein zufällig erzeugter 256 Bit seed mittels SHAKE-128[16] gehashed und zu einem Polynom geparst.

Hierzu wird der Output von SHAKE-128 als Stream von 16 Bit Integer aufgefasst, die mittels modulo-Rechnung auf 14 Bit reduziert werden. Die so entstandenen Koeffizienten können größer als q (12289) ausfallen. Sollte dies der Fall sein wird der Wert verworfen. Das so erzeugte Generatorpolynom a liegt damit auch bereits in der NTT Domäne vor.

Da das gleiche Generatorpolynom a so von jedem erzeugt werden kann ist nur die Übertragung des *seed* notwendig also 32 Byte die einfach an die 1792 Byte der Polynomencodierung von Alice' Schlüssel angehängt werden können. Somit hat Alices Nachricht an Bob eine Größe von 1824 Byte.

4.4.4 Errorsampling – Ψ_k

Learning with errors ist auf einen *guten* Sampler angewiesen der *gute*, *kleine* Polynome erzeugen kann. Kleine Polynome sind hier solche bei denen der Betrag einer 0-zentrierten[17] Darstellung jedes Koeffizienten nahe an 0 liegt.

In anderen Verfahren werden dabei üblicherweise entweder gleichverteiltes sampling aus dem Intervall $\{-b, .., b\}$ benutzt[18] oder diskrete Normalverteilung mit einer geeignet gewählten kleinen Standardabweichung σ. Für die Normalverteilung lassen sich dabei Beweise der Sicherheit der entsprechenden Algorithmen führen.

New Hope Verwendet statt dessen die zentrierte binomial Verteilung Ψ_k mit:

$$\Psi_k = \sum_{i=0}^{k} b_i - b'_i \text{ mit } b_i, b'_i \in \{0, 1\}$$

[16] Eine extendable-output function von Keccak
[17] Koeffizienten liegen dann im Bereich $\{-(q-1)/2, .., (q-1)/2\}$
[18] Singh schlägt Beispielsweise $b = 5$ vor [Singh, 2015]

Die zentrierte binomial Verteilung ist eine Approximation der Normalverteilung die sich erheblich einfacher und effizienter implementieren lässt. Die Autoren führen einen Beweis der zeigt, dass die Wahl eines solchen *Samplers* niedrigerer Präzision die Sicherheit des Verfahrens nicht Maßgeblich negativ beeinflusst und nennen Timing-Angriffe als eine Schwäche üblicher Implementierungen von *Samplern* auf Basis der Normalverteilung.

Unter Verwendung von Ψ_k werden dann Polynome e der Ordnung n erzeugt, so dass:

$$e = e_0 + e_1 x + e_2 x^2 + .. + e_{n-1} x^{n-1}$$

wobei jedes e_i im Bereich $\{-k, k\}$ entsprechend des *Samplers* verteilt ist.

In der portablen Demonstrationsimplementierung wählten die Autoren zur Erzeugung der zufälligen 16 Bit Blöcke ChaCha20 als *pseudo random number generator* mit einer *true random number* als seed.

4.4.5 Error recovery

Wir erinnern uns, dass die sich aus der Anwendung des Protokolls ergebenen Ringelemente $v = us = as's + e's$ und $v' = bs' + e'' = as's + e's + e''$ von dem vom Generatorpolynom abhängigen Teil $as's$ dominiert werden, sich aber in dem verhältnismäßig kleinen Fehlern $e's$ bzw. $es' + e''$ unterscheiden. Dieser Fehler lässt sich durch einen Korrektur-Hinweis von Bob[19] auf Seite von Alice nahezu Vollständig ausgleichen ohne einem Zuhörer Eve Informationen über den Schlüssel zu geben.

Das in New Hope verwendete Korrekturverfahren nutzt den großen Nachrichtenraum von 1024 Koeffizienten geschickt aus, so dass mit einem Korrektur-Hinweis von 256 Byte die Wahrscheinlichkeit einer Abweichung der errechneten Schlüssel unter 2^{-60} liegt.

Key Embedding In dem hier vorgestellten Verfahren wird der Schlüssel in die Koeffizienten des Polynoms eingebettet. Da $n = 1024$ Koeffizienten zur Verfügung stehen aber nur 256 Bit zum erzeugen des gemeinsamen Geheimnisses ausgetauscht werden sollen, wird jedes Bit auf 4 Koeffizienten abgebildet.

Ein einzelnes Bit lässt sich einfach in einen Integer kodieren z.B. mit $b = a/q$, für zwei Koordinaten lässt sich die folgende abgeleitete Idee verwenden: Die beiden Koeffizienten a_0 und a_1 werden durch a_i/q auf einen Bereich von $(0, 1)$ abgebildet. Das Ergebnis wird nun als Vektor $\vec{v}$ in einem zweidimensionalen Gitter betrachtet.

Den Gitterpunkten {(0,0),(0,1),(1,0),(1,1)} wird der Wert 0 zugeordnet, dem Gitterpunkt (1/2,1/2) der Wert 1. Nun kann überprüft werden welchem Vektor $\vec{v}$ am nächsten liegt, was gleichbedeutend ist mit der Frage in welche Voronoi-Zelle der Vektor zeigt.

Dies lässt sich nun mit mehr Koordinaten auf Gitter höherer Dimension übertragen.

[19] Bob ist der erste der v' berechnet. Alice könnte die Berechnung eines Korrektur-Hinweises auch durchführen allerdings wäre dann eine weitere Nachricht erforderlich.

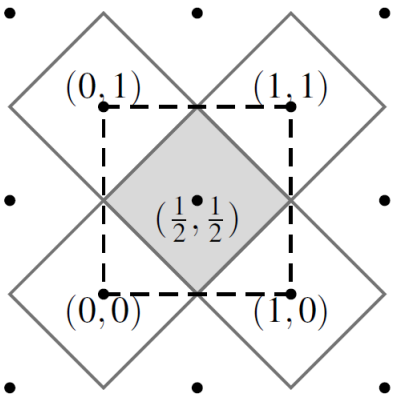

Abbildung 1: 2D Gitter mit Voronoi-Zellen

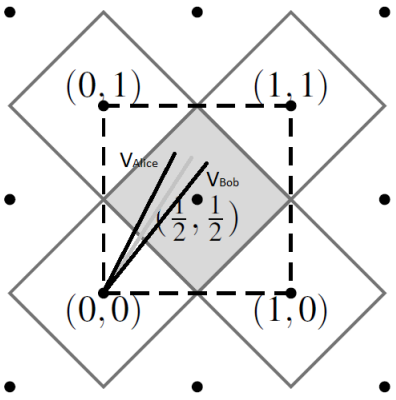

Abbildung 2: Bobs und Alices v zeigen in die gleiche Zelle

Error Reconciliation Da sich die Koeffizienten von Alices und Bobs Ringelement geringfügig unterscheiden ist es möglich, dass die übertragenen Vektoren $\vec{v}$ nicht in die gleiche Voronoi-Zelle zeigen. In dem Fall ist für Alice die Korrekturhilfe von Bob erforderlich.

Die Idee ist hier, dass Bob einen Distanzvektor zu dem nächstgelegenen Gitterpunkt berechnet und diesen Alice als Teil seiner Nachricht zur Verfügung stellt.

Verschiebt Alice ihren eigenen Vektor $\vec{v}$ um diesen Distanzvektor kommt sie (fast) immer Näher an den korrekten Gitterpunkt und die Wahrscheinlichkeit einen gemeinsamen Gitterpunkt zu finden steigt erheblich.

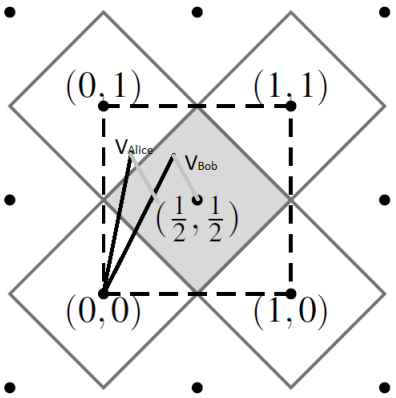

Abbildung 3: Alices Vektor wird durchs Bobs Korrekturhinweis verschoben

Im Falle des in 4 geschilderten Beispiels berechneten Alice und Bob ihre Ringelemente mit $v = 33x + 44$ bzw. $v' = 33x + 40$. Für die Fehlerkorrektur ergibt sich für Bobs Koeffizientenvektor $\vec{v_B} = \begin{pmatrix} 0,62 \\ 0,83 \end{pmatrix}$ was ihn in der grauen Zelle platziert. Somit ergibt sich ein Korrekturvektor von $\vec{r} = \begin{pmatrix} -0,12 \\ -0,33 \end{pmatrix}$ und ein Bitwert von 1. Alice kann mit dem Korrekturhinweis ihren Koeffizientenvektor $\vec{v_A} = \begin{pmatrix} 0,62 \\ 0,75 \end{pmatrix}$ zu einem korrigierten Vektor $\vec{v_A'} = \begin{pmatrix} 0,5 \\ 0,42 \end{pmatrix}$ verrechnen was sie ebenfalls komfortabel in der grauen Zelle – und damit auf einem Bitwert von 1 – platziert.

Der genaue Korrekturvektor kann zur Reduktion der Nachrichtengröße diskretisiert werden. Hierzu wird jede Koordinate in 4 Bereiche eingeteilt, also im 2-Dimensionalen die Zelle in 16 Unterzellen zerlegt. Für jede Unterzelle ergibt sich dann ein Korrekturvektor. Die Information in welcher dieser Unterzellen Bobs $\vec{v}$ zeigt ist zur Korrektur ausreichend.

Auf das 4-Dimensionalen Gitter übertragen reichen dann 8 Bit je Schlüsselbit aus um sehr hohe Korrekturwahrscheinlichkeit zu gewährleisten.

4.5 Performance

Die Autoren führen eine gründliche Betrachtung der Performanz von New Hope an, sowohl in Bezug auf Sicherheit als auch auf Geschwindigkeit. Im Folgenden werden die Ergebnisse kurz zusammengefasst.

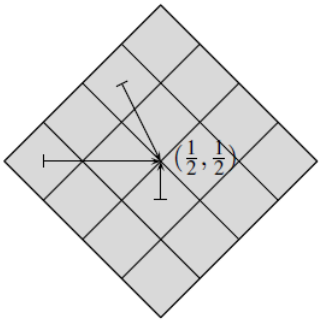

Abbildung 4: Aufteilung der Voronoi-Zelle in 16 Unterzellen und ihr jeweiliger Korrekturvektor

4.5.1 Sicherheit

New Hope verfügt über eine errechnete Sicherheit von 233 Bit. Es liegt also weit über dem Referenzwert von 100 Bit. Im Grunde wäre somit auch eine Variante reduzierter Sicherheit denkbar. Die Autoren präsentieren mit JarJar eine solche Variante mit $n = 512$[20].

Da mit dem Deployment eines standardisierten Verfahrens ein erhöhtes Interesse an Analyse verbunden ist sind die Autoren der Auffassung eine pessimistische Bewertung der Sicherheit sei angemessen und konzentrieren sich daher auf die Optimierung von New Hope. Der Überschuss an Sicherheit kann Verlust durch neue Angriffe besser abfangen. Im Extremfall könnte New Hope auch mit einer Halbierung der Sicherheit noch Kryptographisch eingesetzt werden.

4.5.2 Geschwindigkeit

In der Referenzimplementierung hat sich New Hope als erheblich schneller erwiesen als vergleichbare Post-Quantum Verfahren mit ähnlicher Sicherheit. Im Vergleich mit der Referenzimplementierung von Bos et al. erreichte New Hope etwa 9 mal schnelleren Schlüsselaustausch. Die Geschwindigkeit ist damit durchaus mit modernen Implementierungen der Kryptographie auf Elliptischen Kurven vergleichbar.

[20] Andere Parameter bleiben gegenüber New Hope unverändert

5 Fazit

New Hope wurde von den Autoren als schnelles, effizientes und einfaches Verfahren bezeichnet, geeignet als Ersatz für RSA und Elliptic-Curve Kryptographie.

Die Autoren betonen dabei besonders die **Performanz** die in Versuchsimplementierungen vergleichbaren Aufwand zu haben scheint wie moderne *ECC* Verfahren.

Die **Einfachheit** des Verfahrens zeigt sich in verschiedenen Erfolgreichen Implementierungen in verschiedenen Sprachen, auch zur Integration in OpenSSL, Google Chrome und Apache.

Speichereffizienz erreicht das Verfahren durch Verzicht auf große Datenstrukturen und die Nutzung von NTT.

Analysen zur **Sicherheit** sind konservativ und pessimistisch gehalten. Die errechnete Sicherheit erlaubt Raum für zukünftig besseres Verständnis der zugrunde liegenden Mathematik und daraus resultierender Angriffsmöglichkeiten.

Eine Variante von New Hope ist seit 2016 in Google Chrome Canary integriert[21] und Serverseitig von einigen Googledomains unterstützt.

Eine Variante von New Hope wurde versuchsweise auf einer Smart-Card von Infineon implementiert[22]

Die mathematischen Grundlagen der Kryptographie mit *ring learning with errors* sind relativ jung und werden erst seit wenigen Jahren intensiv untersucht. Wohl möglich wird erst praktischer Einsatz der Technologie und Jahre der Sicherheitsforschung (oder ein mathematischer Beweis der Sicherheit) Ungewissheiten beseitigen.

Es ist jedoch fraglich ob Kommunikationssicherheit auf Gewissheit warten kann. Diffie-Hellman und RSA sind nur einen technischen Durchbruch davon entfernt als gebrochen betrachtet werden zu müssen; alle Kommunikation die jetzt aufgezeichnet wird könnte mit einem hinreichend starken Quantencomputer einfach entschlüsselt werden. Ein Ersatz ist somit dringend angeraten. Mit New Hope als einem von 69 zur NIST Ausschreibung[23] eingereichten Vorschlägen wird sich in naher Zukunft zeigen ob die Technologie robust genug zur Standardisierung ist. Ein *Draft Standard* soll 2022/24 veröffentlicht werden.

[21] https://security.googleblog.com/2016/07/experimenting-with-post-quantum.html
[22] https://www.infineon.com/cms/en/about-infineon/press/press-releases/2017/INFCCS201705-056.html
[23] https://csrc.nist.gov/Projects/Post-Quantum-Cryptography

Literatur

[Alkim et al., 2017] Alkim, E., Avanzi, R., Bos, J., Ducas, L., de la Piedra, A., Pöppelmann, T., Schwabe, P., and Stebila, D. (2017). Newhope: Algorithm specification and supporting documentation. Submission to the NIST Post-Quantum Cryptography Standardization Project. https://cryptojedi.org/papers/#newhopenist.

[Alkim et al., 2015] Alkim, E., Ducas, L., Pöppelmann, T., and Schwabe, P. (2015). Post-quantum key exchange - a new hope. *IACR Cryptology ePrint Archive*, 2015:1092.

[Bos et al., 2014] Bos, J. W., Costello, C., Naehrig, M., and Stebila, D. (2014). Post-quantum key exchange for the tls protocol from the ring learning with errors problem. Cryptology ePrint Archive, Report 2014/599. https://eprint.iacr.org/2014/599.

[Diffie and Hellman, 1976] Diffie, W. and Hellman, M. (1976). New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654.

[Grover, 1996] Grover, L. K. (1996). A fast quantum mechanical algorithm for database search.

[Lyubashevsky et al., 2010] Lyubashevsky, V., Peikert, C., and Regev, O. (2010). On ideal lattices and learning with errors over rings. In Gilbert, H., editor, *Advances in Cryptology – EUROCRYPT 2010*, pages 1–23, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Martin-Lopez et al., 2011] Martin-Lopez, E., Laing, A., Lawson, T., Alvarez, R., Zhou, X.-Q., and O'Brien, J. L. (2011). Experimental realisation of shor's quantum factoring algorithm using qubit recycling.

[NIST, 2001] NIST (2001). Fips pub 197, advanced encryption standard (aes). U.S.Department of Commerce/National Institute of Standards and Technology.

[Rivest et al., 1978] Rivest, R. L., Shamir, A., and Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126.

[Shor, 1995] Shor, P. W. (1995). Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer.

[Singh, 2015] Singh, V. (2015). A practical key exchange for the internet using lattice cryptography. Cryptology ePrint Archive, Report 2015/138. https://eprint.iacr.org/2015/138.