Quanten-Sichere Kommunikation im Internet der Dinge (QUASIKOM)

Endbericht | Call 11 | Projekt ID 1886

Inhalt

Einleitung

Innerhalb der nächsten fünf Jahre wird das Internet der Dinge (engl. Internet of Things, kurz IoT) bis zu 30 Milliarden intelligente Geräte vernetzen, welche vollkommen autonom miteinander interagieren und Daten an Server in einer Cloud senden bzw. von dort aus gesteuert werden. Laut dem Ericsson Mobility Report vom November 2017 wird nur etwa ein Drittel dieser 30 Milliarden Geräte aus klassischen Computern bestehen (z. B. PCs, Notebooks, Smartphones), während die übrigen zwei Drittel verschiedene Arten von intelligenten Geräten sein werden (d.h. Computer, die nicht wie Computer aussehen) wie z.B. Fahrzeuge, Maschinen, Messgeräte, Sensoren, Point-of-Sale-Terminals und Unterhaltungselektronik. Eine der größten Herausforderungen für praktische IoT Anwendungen ist die Sicherheit und Zuverlässigkeit der Daten, die von den Geräten gesammelt, übertragen und ausgewertet werden. So gut wie alle heute verwendeten kryptografischen Algorithmen und Protokolle (z.B. TLS) werden in nicht allzu ferner Zukunft total unsicher sein weil sie mit Hilfe eines Quantencomputers relativ einfach geknackt werden können. Im Rahmen dieses Projekts wurde ein Prototyp eines TLS-ähnlichen sicheren Kommunikationsprotokolls für das IoT entwickelt, welches kryptografischen Angriffen mit zukünftigen Quantencomputern standhält. Neben herkömmlichen PCs und Smartphones ist der Prototyp auch auf Kleingeräten mit 8, 16 und 32-bit Prozessoren lauffähig.

Viele namhafte Firmen wie z.B. Google, IBM oder Microsoft arbeiten an der Realisierung von Quantencomputern, wobei zurzeit ein regelrechter Wettlauf um den Quantencomputer mit der größten Zahl an Qubits stattfinden. Qubits sind die grundlegenden Recheneinheiten eines Quantencomputers, vergleichbar mit Bits bei einem normalen Computer. Ein "normales" Bit kann einen der beiden Zustände 0 und 1 einnehmen, ähnlich einem Schalter, der entweder aus- oder eingeschaltet sein kann. Qubits können neben diesen beiden Zuständen auch noch einen fragilen Mischzustand einnehmen, der als Überlagerung (engl. Superposition) bezeichnet wird und den man sich als 0 und 1 simultan vorstellen kann. Quantencomputer nutzen noch eine zweite Eigenschaft von Qubits, nämlich dass mehrere Qubits miteinander durch die sogenannte Quantenverschränkung (engl. Quantum Entanglement) agieren können. Dabei lässt sich der kombinierte Gesamtzustand dieser Qubits beschreiben, jedoch nicht die individuellen Zustände der einzelnen Qubits. Durch diese quantenmechanischen Effekte verdoppelt sich die Zahl der Überlagerungszustände mit jedem zusätzlichen Qubit, wodurch sich auch die Rechenleistung verdoppelt. Somit steigt die Leistungsfähigkeit eines Quantencomputers exponentiell mit der Anzahl der Qubits. Die Zahl von 50 Qubits wird allgemein als die Schwelle gesehen, ab der Quanten-Überlegenheit oder Quanten-Vorherrschaft (engl. Quantum Supremacy) erreicht wird. Es wird nämlich angenommen, dass ein Quantencomputer mit 50 Qubits gewisse Berechnungen schneller durchführen kann als die zurzeit leistungsfähigsten Supercomputer, selbst wenn diese über Hunderttausende Prozessoren verfügen. Zu diesen Berechnungen zählt das Lösen von gewissen hochkomplexen Optimierungsproblemen wie sie z.B. in der Materialforschung oder Medikamentenentwicklung vorkommen.

Eine große Herausforderung bei der Realisierung von Quantencomputern ist die Fragilität bzw. Empfindlichkeit von Quantenzuständen gegen äußere Einflüsse. Die zurzeit für Qubits verwendeten Teilchen verlieren bei Wechselwirkung mit Materie oder bei elektromagnetischer Strahlung ihre Information. Deshalb besteht die zuvor erwähnte Superposition nur für wenige Mikrosekunden, in denen dann die Berechnungen durchgeführt werden müssen. Konkret kann der von IBM vorgestellte Quantencomputer seine 50 Qubits nur für einen Zeitraum von 90 Mikrosekunden stabil halten. Leider reicht das nicht aus um wirklich nützliche Berechnungen durchzuführen und es werden wohl noch einige Jahre vergehen, bis die Stabilität bzw. Fehler-Toleranz-Mechanismen soweit fortgeschritten sind, dass Quantencomputer tatsächlich bei der Entwicklung neuer Medikamente helfen können. Das hat aber auch seine positiven Seiten. Ein leistungsfähiger Quantencomputer kann nämlich nicht nur komplexe Optimierungsprobleme effizient lösen, sondern wäre auch in der Lage, alle zur Zeit verwendeten asymmetrischen kryptografischen Algorithmen zu knacken. So gut wie alle modernen Sicherheitsprotokolle, insbesondere TLS, verwenden asymmetrische Kryptografie für Schlüsselaustausch bzw. digitale Signaturen. Es wird angenommen, dass ein Quantencomputer mit einigen hundert Qubits ausreichen würde, um den weit verbreiteten RSA Algorithmus zu brechen. Ein Quantencomputer wäre auch in der Lage, alle kryptografischen Algorithmen die auf Elliptischen Kurven basieren zu knacken, jedoch sind dafür einige tausend Qubits notwendig. Die Schätzungen, wann solche leistungsfähigen Quantencomputer Realität werden, gehen sehr weit auseinander. Es ist jedoch nicht unwahrscheinlich, dass es bereits in 10 bis 15 Jahren so weit sein könnte.

Zum Glück bedeutet das Aufkommen von hoch-entwickelten Quantencomputern nicht automatisch das Ende von asymmetrischen kryptografischen Algorithmen und auch nicht das Ende von Sicherheitsprotokollen wie TLS. Es gibt nämlich gewisse mathematische Probleme, die selbst für den leistungsfähigsten Quantencomputer nicht lösbar sind, und damit lassen sich kryptografische Algorithmen entwickeln, die absolut unangreifbar sind. Post-Quanten-Kryptographie (engl. Post-Quantum Cryptography, kurz PQC) ist ein Teilgebiet der Kryptografie dass sich mit der Erforschung von neuartigen kryptografischen Algorithmen beschäftigt, die selbst mit einem Quantencomputer nicht geknackt werden können. So gut wie alle großen Internetkonzerne beschäftigen sich intensiv mit PQC um für den Tag X vorbereitet zu sein, ab dem herkömmliche kryptografische Algorithmen wie RSA als nicht mehr sicher erachtet werden müssen. Google experimentiert schon seit längerer Zeit mit PQC und hat auch bereits ein post-quanten-sicheres Verfahren für kryptografischen Schlüsselaustausch in den Chrome Webbrowser integriert. Microsoft arbeitet ebenfalls intensiv daran, seine Produkte und Services gegen kryptografische Angriffe mit Quantencomputern abzusichern.

Projektbeschreibung

Ein leistungsfähiger Quantencomputer stellt eine ernste Gefahr für die Sicherheit der heute verwendeten asymmetrischen kryptografischen Algorithmen (z.B. RSA, Diffie-Hellman, oder

Elliptische-Kurven-Kryptografie) dar. Der Übergang von den heute verwendeten kryptografischen Algorithmen auf Post-Quanten-Kryptografie ist zweifellos eine der größten Herausforderungen für das Internet seit dem Übergang von IPv4 auf IPv6, da praktisch die gesamte Public-Key Infrastruktur erneuert werden muss. Zum Glück gibt es einige mathematische Probleme, die selbst für den leistungsfähigsten Quantencomputer nicht effizient lösbar sind, und basierend auf solchen Problemen kann Post-Quanten-Kryptografie realisiert werden. Sogenannte gitterbasierte kryptografische Algorithmen bieten einerseits hohe Resistenz gegen Angriffe mit einem Quantencomputer und sind andererseits auch sehr effizient, d.h. sie erfordern relativ geringen Rechenaufwand und zeichnen sich außerdem durch einigermaßen kurze Schlüssel- und Signaturlängen aus. Die beiden wichtigsten Vertreter von gitterbasierter Kryptografie sind Ring-LWE und NTRU, wobei das letztgenannte Kryptosystem weiter ausgereift ist, was letztendlich den Ausschlag gab, NTRU im QUASIKOM Projekt zu verwenden. Ein wesentliches Ziel des Projekts bestand darin, NTRU so zu implementieren, dass es auch auf Kleinstgeräten mit sehr wenig Rechenleistung lauffähig ist und akzeptable Ausführungszeiten erzielt. Um dies zu gewährleisten, wurden zwei wichtige Komponenten von NTRU, nämlich die Polynom-Arithmetik und die Hashfunktion SHA, nicht nur in der Programmiersprache C sondern auch in Assembler realisiert. Diese Komponenten stellen das Herzstück des Projekts dar und werden im Folgenden näher erklärt.

Die im Rahmen des Projekts entwickelte Assembler-Implementierung der Polynom-Multiplikation orientiert sich am aktuellen Stand der Forschung und beinhaltet alle wichtigen Optimierungen die in der Fachliteratur beschrieben sind. Insbesondere wurde berücksichtigt, dass wegen der speziellen Wahl der Parameter für den Ring der Faltungspolynome sowohl die Modulo-Reduktion der Koeffizienten-Produkte als auch die Faltung der Polynom-Produkte mit einfachen arithmetischen Operationen durchgeführt werden kann. Zusätzlich wird in der Implementierung noch ein dritter Trick eingesetzt um die Effizienz der Polynom-Multiplikation weiter zu steigern. Es ist nämlich möglich, eines der beiden Polynome, die miteinander multipliziert werden sollen, so zu wählen, dass seine Koeffizienten nur die Werte 0, 1 und -1 haben. Dadurch lässt sich eine Polynom-Multiplikation über Additionen bzw. Subtraktionen von Koeffizienten realisieren, d.h. es müssen keine Multiplikationen von Koeffizienten ausgeführt werden. Das ist natürlich besonders nützlich, wenn man NTRU für kleine 8 oder 16-bit Mikrocontroller implementiert, da diese normalerweise nur mit einem langsamen Multiplizierer ausgestattet sind, und somit für die Exekution einer Multiplikation relativ viele Taktzyklen benötigen. Eine Implementierung der Polynom-Multiplikation mit den drei erwähnten Tricks befindet sich im Github Repository, und zwar in Form der Funktion ring_mul_sparse(), welche sich in der Datei ring_arith.c im Unterverzeichnis src/ntru befindet. Da der Source-Code nicht sonderlich kompliziert aussieht, sollte man meinen, dass ein optimierender C Compiler in der Lage ist, effizienten Binärcode daraus zu generieren. Nach einer Inspektion des vom Compiler erzeugten Assembler-Outputs wurde jedoch sehr schnell klar, dass dies leider nicht der Fall ist. Deshalb wurden alle performance-kritischen Teile der Polynom-Multiplikation auch in Assembler implementiert, wodurch sich die Ausführungszeit um fast 30% verringert hat. Die Assembler-Source-Codes für 8-bit AVR Prozessoren befinden sich

im Unterverzeichnis avrasm. Auf einem 8-bit AVR ATmega128 Prozessor beträgt die Ausführungszeit einer Polynom-Multiplikation mit den Parametern N = 439 und q = 2048 etwas mehr als 250.000 Taktzyklen, was bei einer typischen Taktfrequenz von 8 MHz ungefähr 32 Millisekunden entspricht.

Neben der Polynom-Arithmetik ist vor allem die Hashfunktion SHA256 performance-kritisch und hat einen erheblichen Einfluss auf die Gesamt-Laufzeit von NTRU. Die im Rahmen des Projekts realisierte Implementierung von SHA-256 stellt sowohl eine High-Level- als auch eine Low-Level-Programmierschnittstelle (API) zur Verfügung, wobei Erstere nur aus einer einzigen Funktion besteht, nämlich sha256_hash(). Auf der anderen Seite realisiert das Low-Level-API eine sogenannte I-U-F Schnittstelle (d.h. besteht aus den üblichen sha256_init(), sha256_update() und sha256_final() Funktionen) und wird im Allgemeinen zum Hashen großer oder fragmentierter Daten bevorzugt. Die High-Level-Funktion sha256_hash() benötigt die gesamten zu hashenden Daten im RAM, was in gewissen Situationen problematisch sein kann. Zum Beispiel ist es mit der High-Level-Funktion nicht möglich, eine Datei zu hashen, deren Größe die verfügbare RAM-Kapazität übersteigt. Bei Verwendung des Low-Level-API kann jedoch eine große Datei in kleineren Teilen verarbeitet werden, indem die Teile einfach ins RAM geladen und sha256_update() aufgerufen wird, bis die gesamte Datei gehashed ist. Die sogenannte Kompressionsfunktion ist die wichtigste Komponente von SHA-256 und besteht im Wesentlichen aus einer Schleife, die 64-mal durchlaufen wird. In jedem Durchlauf werden sechs logische Operationen durchgeführt, die in der SHA-256 Spezifikation als Ch, Maj, Sigma0, Sigma1, Rho0 und Rho1 bezeichnet werden. Vier dieser Operationen, nämlich Sigma0, Sigma1, Rho0 und Rho1, führen Rotationen und Verschiebe-Operationen auf 32-bit Worten aus, welche in einer Programmiersprache wie C oder C++ nicht wirklich effizient realisiert werden können. Deshalb macht es Sinn, diese Funktionen in Assembler zu implementieren. Konkret zeigen die Simulationsergebnisse auf einem 8-bit AVR Prozessor dass die C Implementierung der Kompressionsfunktion eine Ausführungszeit zwischen 70.000 und 90.000 Taktzyklen hat, wobei die genaue Zahl von der Optimierungsstufe des Compilers abhängt. Auf der anderen Seite benötigt die Assembler-Implementierung nur etwas mehr als 33.000 Taktzyklen, ist also mindestens doppelt so schnell. Der C Source-Code befindet sind im Github-Repository im Verzeichnis src/ntru und der Assembler-Code der Kompressionsfunktion im Unterverzeichnis avrasm.

Verlauf der Arbeitspakete

3.1 Arbeitspaket 1 - *Projekt-Management, Administration und Öffentlichkeitsarbeit*

Wie im Projektplan ausgeführt, besteht das Projekt aus insgesamt vier Arbeitspaketen (APs). Das erste Arbeitspaket (AP1) umfasst diverse Tätigkeiten im Ausmaß von insgesamt 120 Stunden in den Bereichen Projektmanagement, Administration, sowie Öffentlichkeitsarbeit (wozu insbesondere das Verfassen von Blogeinträgen zählt). Zum Zeitpunkt der Erstellung dieses vorläufigen Endberichts sind ungefähr 90% der im AP1 insgesamt geplanten Tätigkeiten realisiert, wobei sich das Verfassen von Blogeinträgen,

diverse administrative Tätigkeiten (z.B. Projektplanung und Reporting), und das Vorbereiten von Präsentationen bei Fachkonferenzen als besonders aufwändig herausstellten.

Eine Abweichung vom ursprünglichen Projektplan betrifft die Anzahl der verfassten Blogeinträge. Im Berichtzeitraum wurden sieben Blogeinträge geschrieben, jedoch sind drei dieser sieben Einträge ziemlich umfangreich und haben daher relativ viel Zeit in Anspruch genommen. Zwei weitere Blogeinträge sind noch geplant, wobei der Erste die Ergebnisse des Fachartikels über die Hashfunktion zusammenfasst und Eindrücke von der SECITC-Konferenz vermitteln wird. Im zweiten noch geplanten Blogeintrag werden dann die Ergebnisse des anderen Fachartikels über die Polynom-Arithmetik zusammengefasst.

Eine weitere Abweichung vom ursprünglichen Projektplan betrifft die Ausgaben für Sach- und Reisekosten. Anfänglich waren Ausgaben in der Höhe von 3000 € für Software-Entwicklungs-Tools für 8, 16, und 32-bit Mikrocontroller vorgesehen. Im Rahmen der Entwicklungsarbeiten an der NTRU-Implementierung stellte sich jedoch heraus, dass die gratis verfügbaren “Kickstart-Versionen” der Tools ausreichend sind. Eine “Kickstart-Version” hat denselben Funktionsumfang wie die Voll-Version des Tools, ist aber beschränkt bezüglich der maximalen Größe des exekutierbaren Programms, üblicherweise auf 32 kB. Diese 32 kB waren jedoch vollkommen ausreichend für die Entwicklung von NTRU, selbst dann wenn noch Testfunktionen in das Programm integriert wurden. Bei der Integration von NTRU in den TinyDTLS-Protokoll-Stack wurde hauptsächlich unter dem freien Betriebssystem Linux gearbeitet und die frei verfügbare GNU Compiler Collection (GCC) verwendet. Somit sind entgegen der ursprünglichen Projektplanung keine Ausgaben für Software-Entwickler-Tools angefallen. Auf der anderen Seite sind die Ausgaben für Reisekosten höher als ursprünglich geplant. Neben drei Fahrten nach Wien zu Netidee-Veranstaltungen (ca. 600 Euro zusammen) ist in diesem Zusammenhang noch die Teilnahme an der Fachkonferenz SECITC 2018 in Bukarest zu erwähnen, bei der ein wissenschaftlicher Artikel über die Hashfunktion SHA präsentiert wurde. Zusätzlich ist noch geplant, einen weiteren Artikel bei der Konferenz WISTP 2018 im Dezember in Brüssel zu präsentieren. Die Reisekosten für SECITC beliefen sich auf ca. 1050 € und jene für den Besuch von WISTP werden auf ca. 850 € geschätzt.

Insgesamt haben die erwähnten Tätigkeiten im AP1 ungefähr 110 Stunden in Anspruch genommen, was den oben erwähnten 90% der Gesamtstundenzahl des AP1 entspricht. Die verbleibenden 10% werden für das Verfassen von zwei weiteren Blogeinträgen verwendet.

3.2 Arbeitspaket 2 - *Implementierung des NTRU Public-Key Kryptosystems*

Das mit Abstand aufwändigste Arbeitspaket des Projekts ist AP2, welches laut Projektplan insgesamt 540 Stunden umfasst. Im Rahmen des AP2 wurde das NTRU Public-Key Kryptosystem vollständig in C und in Assembler für 8-bit AVR sowie 16-bit MSP430 Prozessoren implementiert und getestet. Zu den konkret ausgeführten Tätigkeiten zählen z.B. die Implementierung der NTRU Polynom-Arithmetik in C und Assembler, die Implementierung der restlichen NTRU Komponenten (insbesondere der Hashfunktion

SHA256), das Testen all dieser Komponenten (auf verschiedenen Prozessoren), und das Verfassen der Dokumentation. Die Dokumentation erfolgte in Form eines 20-seitigen wissenschaftlichen Fachartikels, der zur Präsentation bei einer Fachkonferenz eingereicht wurde und zurzeit begutachtet wird. Ein weiterer Fachartikel über die Hashfunktion SHA256 wurde zur Präsentation bei der internationalen IT-Sicherheitskonferenz SECITC 2018 angenommen, welche im November in Bukarest stattfinden wird. Der Konferenzband wird in der renommierten Buchreihe "Lecture Notes in Computer Science" des Springer Verlags veröffentlicht. Alle Source-Codes sind auf GitHub verfügbar, wobei sich die Assembler-Files für die AVR-Plattform im Unterverzeichnis src/avrasm befinden, und jene für MSP430 in src/mspasm. Insgesamt wurden im Rahmen des AP1 ungefähr 3000 Lines-Of-Code (LOC) geschrieben, etwa die Hälfte davon in Assembler.

Im Vergleich zum Projektplan gab es in AP2 eine geringe Abweichung bezüglich der Plattformen, für die Assembler-Code entwickelt wurde. Ursprünglich war geplant, die performance-kritischen Teile von NTRU für MSP430 und ARM Prozessoren in Assembler zu implementieren. Im Laufe der Projektarbeit stellte sich jedoch heraus, dass der verwendete C-Compiler für ARM sehr effizienten Binärcode erzeugt, der mit selbst mit Hilfe einer hoch-optimierten Assembler-Implementierung kaum verbessert werden konnte. Deshalb entschied der Fördernehmer, 8-bit AVR Prozessoren anstatt 32-bit ARM Prozessoren mit optimierten Assembler-Code zu unterstützen. Somit wurden im AP2 Assembler-Implementierungen für AVR und MSP430-Prozessoren entwickelt, während auf ARM-Prozessoren gewöhnlicher C-Code zum Einsatz kommt.

Eine weitere Abweichung vom ursprünglichen Projektplan ergab sich im Zusammenhang mit den beiden Fachartikeln, die als Teil der Dokumentation in AP2 verfasst wurden (das Vorbereiten der Präsentationen und das Erstellen der Foliensätze zählen jedoch zu Öffentlichkeitsarbeit und sind somit Teil von AP1). Der aktualisierte Projektplan enthält einen Task T2.7, der im ursprünglichen Projektplan nicht enthalten war und das Überarbeiten (d.h. Verbessern und Korrigieren) der beiden Fachartikel zum Ziel hat. Die Arbeiten an diesem Task wurden in den letzten beiden Monaten (d.h. September und Oktober) durchgeführt und nahmen ungefähr 40 Stunden in Anspruch. Dadurch erhöhte sich der Gesamtaufwand für AP2 von 540 auf 580 Stunden.

3.3 Arbeitspaket 3 - *Integration von NTRU in einen Open-Source DTLS Stack*

Das AP3 hat laut Projektplan einen Umfang von 360 Stunden und ist damit das zweitgrößte Arbeitspaket des gesamten Projekts. Der erste Schritt im AP3 umfasste die Evaluierung von verschiedenen Open-Source DTLS-Implementierungen hinsichtlich Ihrer Eignung für die Integration von Post-Quanten-Kryptografie, sowie Ihrer Eignung zum Einsatz auf IoT-Kleingeräten mit geringer Rechenleistung und wenig RAM. Im Rahmen dieser Tätigkeit wurden sechs verschiedene DTLS-Implementierungen untersucht und dann entschieden, dass TinyDTLS am besten für das QUASIKOM Projekt geeignet ist da es nicht nur mit relativ wenig RAM auskommt, sondern auch bezüglich der Größe der exekutierbaren Datei am besten abschnitt. Nachdem die Wahl von TinyDTLS feststand war der nächste Schritt die Definition und Implementierung einer sogenannten Cipher-Suite, welche im

Wesentlichen eine Kollektion von kryptografischen Algorithmen darstellt, die beim DTLS Handshake-Subprotokoll und beim Record-Subprotokoll zum Einsatz kommt. Zu diesem Zweck wurde eine RSA-basierte Cipher-Suite so modifiziert, dass NTRU anstelle von RSA zum Schlüsselaustausch verwendet wird. Danach wurde diese neue Post-Quanten Cipher-Suite in TinyDTLS eingebaut, was sich als relativ einfache Tätigkeit herausstellte da DTLS ein Algorithmen-agiles Protokoll ist, d.h. so spezifiziert wurde dass es mit wenig Aufwand möglich ist, einen der verwendeten kryptografischen Algorithmen durch einen alternativen Algorithmus zu ersetzen.

Die einzige nennenswerte Abweichung vom ursprünglichen Projektplan betrifft die Anzahl der für AP3 aufgewendeten Stunden. Leider musste festgestellt werden, dass im Source-Code von TinyDTLS recht viele Programmierfehler enthalten sind, die in mühsamer Arbeit ausgemerzt werden mussten. Durch diese Bug-Fixes, die nicht NTRU betrafen sondern die originale TinyDTLS Implementierung, vergrößerte sich die Zahl der im AP3 aufgewendeten Arbeitsstunden von 360 auf 380.

3.4 Arbeitspaket 4 - *Integrations- und Systemtests*

Das AP4 stellt den Abschluss der praktischen Arbeiten am Projekt dar und umfasst laut Projektplan 180 Stunden. In diesem AP wurde die entwickelte Software ausführlichen Tests auf Komponenten-, Modul- und Systemebene unterzogen. Zusätzlich wurde der gesamte Source-Code mit Werkzeugen zur statischen Code-Analyse, insbesondere Splint, untersucht und potentielle Speicherfehler (z.B. ungültige Speicherzugriffe, inkorrekte Freigabe von dynamisch-bereitgestelltem Speicher, Memory Leaks) mit Hilfe von Valgrind aufgespürt. Bei den zuvor erwähnten Tests auf Komponenten-, Modul- und Systemebene wurden jeweils drei verschiedene Eigenschaften untersucht, nämlich Funktionalität, Performance, und Robustheit. Funktionalitätstests sind der übliche Ansatz zur Überprüfung der Korrektheit von Software, wobei die zu testenden Funktionen und Module nur mit gültigen Eingaben (d.h. Parametern die der Spezifikation entsprechen) "gefüttert" werden. Im Gegensatz dazu kommen bei Robustheitstests absichtlich ungültige Eingabewerte bzw. Parameter zum Einsatz, um zu prüfen, ob diese erkannt und abgefangen werden, bevor sie Unheil anrichten können. Bei Performancetests wird die Ausführungszeit von Funktionen in Abhängigkeit der Größe der Parameter (z.B. Ordnung der Polynome bei NTRU) bestimmt. Diese Tests spielen eine wesentliche Rolle bei kryptografischer Software, da mit ihrer Hilfe Schwachstellen aufgespürt werden können, die eine Implementierung anfällig für sogenannte Timing-Attacken machen

Leider war es auch in diesem Arbeitspaket notwendig, etliche Fehler im originalen TinyDTLS Source-Code zu korrigieren, was den Gesamtaufwand von den geplanten 180 Stunden auf 200 Stunden erhöhte. Abgesehen von diesem Mehraufwand gab es keine nennenswerten Abweichungen vom ursprünglichen Projektplan.

Liste Projektendergebnisse

1	Source-Code der NTRU Implementierung (die Assembler-Optimierungen befinden sich in den Unterverzeichnissen avrasm und mspasm). Die im ersten Fachartikel beschriebene Assembler-Optimierung der Hashfunktion SHA befindet sich in der Datei sha256_compress.S. Das Unterverzeichnis avrasm enthält zusätzlich die Assembler-Dateien der im zweiten Fachartikel beschriebenen Polynom-Arithmetik (macros.S, mod3.S, ring_mul_cfadd.S, und ring_mul_cfsub.S).	GPLv3	http://github.com/grojoh/quasikom/tree/master/src/ntru
2	Source-Code der DTLS-Implementierung TinyDTLS mit den für die Unterstützung von NTRU notwendigen Änderungen und Erweiterungen.	Eclipse Public License (EPL) v1.0	http://github.com/grojoh/quasikom/tree/master/src
3	14-seitiger wissenschaftlicher Fachartikel über die Assembler-Implementierung der Hashfunktion SHA-2. Dieser Artikel wurde bei der Fachkonferenz “International Conference on Information Technology and Communication Security (SECITC 2018)” präsentiert, welche im November in Bukarest stattfand. Der Fachartikel sowie der bei der Präsentation verwendete Foliensatz sind online verfügbar.	CC-BY-SA	http://www.netidee.at/quasikom/secitc2018
4	14-seitiger wissenschaftlicher Fachartikel über die Assembler-Implementierung der Polynom-Arithmetik. Dieser Artikel wurde zur Präsentation bei der Fachkonferenz “International Conference on Information Security Theory and Practice (WISTP 2018)” angenommen, welche im Dezember in Brüssel stattfindet. Der Fachartikel sowie der bei der Präsentation verwendete Foliensatz nach der Konferenz online zur Verfügung gestellt.	CC-BY-SA	http://www.netidee.at/quasikom/wistp2018
5	Diverse Blogeinträge auf der Projekt-Webseite. Ein weiterer Blogeinträge ist noch geplant, in dem die Ergebnisse des Fachartikels über die Polynom-Arithmetik zusammenfasst und Eindrücke von der SECITC-Konferenz zum Inhalt haben wird. Im zweiten noch geplanten Blogeintrag werden dann die Ergebnisse des anderen Fachartikels über die Polynom-Arithmetik zusammengefasst.	CC-BY-SA	http://www.netidee.at/quasikom

Verwertung der Projektergebnisse in der Praxis

Die Projektergebnisse bestehen einerseits aus neuen wissenschaftlichen Erkenntnissen über die effiziente Realisierung von Post-Quanten-Kryptografie und andererseits aus dem Source-Code der NTRU-Implementierung und der Erweiterung von TinyDTLS. Die im Rahmen des Projekts entwickelten Methoden zur Software-Optimierung setzten neue Maßstäbe im Bereich der effizienten Implementierung von Hashfunktionen und Polynom-Arithmetik und erweitern daher den aktuellen Stand der Forschung. Da diese Methoden in der Fachliteratur publiziert werden, können andere Wissenschaftler die im Bereich Post-Quanten-Kryptografie arbeiten sie direkt in Ihre Forschungsprojekte übernehmen und darauf aufbauen. Der Source-Code ist auf GitHub frei verfügbar und kann von anderen Software-Entwicklern direkt in Ihre Projekte übernommen werden. In diesem Zusammenhang ist vor Allem die Implementierung der Hashfunktion SHA relevant, da diese zahlreiche Anwendungen außerhalb der Post-Quanten-Kryptografie hat, welche von digitalen Signaturen bis zur Blockchain-Technologie (z.B. Bitcoin) reichen.

Öffentlichkeitsarbeit / Vernetzung

Neben den Blogeinträgen erfolgte bzw. erfolgt die Öffentlichkeitsarbeit in erster Linie über Vorträge bei Fachkonferenzen und Fortbildungsseminaren. Der Fördernehmer hielt einen Vortrag zum Thema “IoT Security” im Rahmen eines “IoT Hackathons”, der zusammen mit dem Excellium IoT Security and Privacy Day am 19. Oktober in Luxemburg stattfand, und präsentierte dabei auch die Ergebnisse des QUASIKOM Projekts. Ein weiterer Vortrag über die Implementierung und Optimierung der Hashfunktion SHA-2 wurde im November bei der Fachkonferenz “International Conference on Information Technology and Communication Security (SECITC 2018)” gehalten. Des Weiteren ist geplant, die Ergebnisse des QUASIKOM-Projekts in einem Vortrag bei der IoT Austria Gruppe Graz zu präsentieren. Diese Präsentation wird im Dezember stattfinden. Ein vierter Vortrag über die Implementierung und Optimierung der Polynom-Arithmetik ist im Rahmen der Fachkonferenz “International Conference on Information Security Theory and Practice (WISTP 2018)” vorgesehen, welche im Dezember in Brüssel abgehalten werden wird.

Geplante Aktivitäten nach netidee-Projektende

Sowohl die NTRU-Implementierung als auch die TinyDTLS-Software werden nach Projektende in zwei Richtungen weiter entwickelt. Im November 2017 wurde im Rahmen einer Initiative des amerikanischen National Institute of Standards and Technology (NIST) zur Standardisierung von Post-Quanten-Kryptografie eine neue Version von NTRU vorgestellt, die sich erheblich von der aktuellen Version unterscheidet und einige Verbesserungen mit sich bringt. Es ist geplant, die zurzeit in TinyDTLS integrierte NTRU-Version durch die neue Version zu ersetzen. Auch bei TinyDTLS selbst gibt es noch viel Potential für Optimierungen, insbesondere um den Speicherverbrauch zu reduzieren und

die Laufzeit weiter zu verbessern. Es ist geplant, NTRU auch in zukünftige Versionen von TinyDTLS zu integrieren und die Schnittstellen, falls nötig, entsprechend anzupassen. Neben NTRU gibt es noch ungefähr ein Dutzend weiterer Post-Quanten-Kryptosysteme, die auf ähnlicher Polynom-Arithmetik basieren. Der Fördernehmer plant, die im Rahmen des QUASIKOM-Projekts entwickelte Polynom-Arithmetik und die Anforderungen einiger alternativer Post-Quanten-Kryptosysteme anzupassen und dann deren Effizienz mit NTRU zu vergleichen.

Anregungen für Weiterentwicklungen durch Dritte

Die um NTRU erweiterte TinyDTLS-Implementierung ist in erster Linie für Entwickler von IoT-Anwendungen interessant und kann im Prinzip in jedem Projekt zum Einsatz kommen, in dem sensible Daten zwischen zwei IoT-Geräten übertragen werden müssen. Typische Beispiele sind Anwendungen in der Medizintechnik oder im Bereich “Home Automation” bzw. “Smart Home”.