Evaluation der Kryptografiebibliothek NaCl im Kontext drahtloser Sensornetze

Markus Jung
Institut für Telematik
Karlsruher Institut für Technologie
E-Mail: markus.jung@kit.edu

Anton Hergenröder*
E-Mail: anton.hergenroeder@kit.edu

Katharina Männle
Karlsruher Institut für Technologie
E-Mail: katharina.maennle@student.kit.edu

***Zusammenfassung*—Als Teil des Internet of Things erfassen, verarbeiten und kommunizieren drahtlose Sensorknoten potentiell sensible und schutzbedürftige Daten. Klassische Schutzziele wie Vertraulichkeit, Integrität und Authentizität können durch entsprechende kryptographische Primitive gewährleistet werden. In dieser Arbeit untersuchen wir die nach aktuellem Stand der Forschung als modern und sicher geltenden Primitive der Kryptographiebibliothek NaCl auf ihre Eignung für den Einsatz in drahtlosen Sensornetzen. Unsere Ergebnisse zeigen, dass insbesondere die authentifizierte Verschlüsselung mit NaCl deutlich effizienter als eine vergleichbare Kombination von AES-128-ECB und SHA-256 ist. Es war aber auch festzustellen, dass die Bibliothek verhältnismäßig viel Programmspeicher benötigt.**

I. Einleitung

Im *Internet of Things* spielen drahtlose Sensorknoten eine wichtige Rolle. Als Datenquelle sind sie ausgestattet mit Sensoren zur Erfassung ihrer Umgebung, bilden drahtlos kommunizierend Netzwerke und werden durch Batterien mit Energie versorgt. Dabei unterliegen sie starken Ressourcenbeschränkungen und stellen so eine besondere Herausforderung für die Entwicklung von Anwendungen und Protokollen für drahtlose Sensornetze dar. Insbesondere der energieeffizienten Kommunikation wird in diesem Kontext eine wichtige Bedeutung beigemessen.

Der Austausch von oftmals sensiblen Daten erfordert die Gewährleistung von Kommunikationssicherheit, gleiches gilt auch für Aspekte der persönlichen Sicherheit (Safety). Da klassische Sicherheitsmechanismen bisher den Aspekt der Energieeffizienz ungenügend berücksichtigen, ist es essentiell Wege zur energieeffizienten Realisierung von Kommunikationssicherheit unter den gegebenen Ressourceneinschränkungen zu erforschen.

Es existieren bereits einige für Mikrocontroller optimierte Kryptographiebibliotheken, welche eine Sammlung von kryptographischen Primitiven bereitstellen. Die im Bereich von drahtlosen Sensornetzen noch nicht untersuchte Bibliothek *NaCl* (ausgesprochen *Salt*) [1] verfolgt dagegen einen anderen Ansatz. Um zu vermeiden, dass einzelne sichere Primitive durch eine ungünstige Kombination zu einem unsicheren Gesamtsystem kombiniert werden, bietet NaCl unter anderem eine anwenderfreundliche Schnittstelle, welche eine authentifizierte Verschlüsselung auf sichere Weise realisiert.

*Bei Entstehung dieser Arbeit tätig am Institut für Telematik des Karlsruher Instituts für Technologie.

In dieser Arbeit wird der Einsatz der Kryptographiebibliothek NaCl für die sichere Kommunikation in Sensornetzen evaluiert. Hierzu wurde eine für Atmel AVR Mikrocontroller portierte Version von NaCl namens *AVRNaCl* [2], in *TinyOS*, einem Betriebssystem für drahtlose Sensorknoten, eingebunden und für die Verschlüsselung und Authentifizierung von Nachrichten verwendet. Als Sensornetzplattform kamen die weit verbreiteten *MICAz*-Sensorknoten zum Einsatz. Die Performanz und Energieeffizienz von NaCl wurde mit dem realitätsnahen Simulator *Avrora+* [3] evaluiert. Weiterhin wurden Experimente mit den Kryptographiebibliotheken *TinyECC* und *AVR-Crypto-Lib* durchgeführt und die jeweiligen Ergebnisse zum Vergleich mit NaCl herangezogen.

II. Grundlagen

Sichere Kommunikation in drahtlosen Sensornetzen stellt, speziell vor dem Hintergrund der beschränkten Ressourcen, eine große Herausforderung dar. Insbesondere asymmetrische Kryptoverfahren sind sehr rechen- und damit auch energieintensiv. Eine mögliche Maßnahme zur Steigerung der Energieeffizienz ist die Auswahl geeigneter Algorithmen und optimierter Implementierungen. Auch Verwendung dedizierter Kryptoprozessoren [4] kommt dabei in Betracht.

Die AVR-Crypto-Lib [5] stellt kompakte Implementierungen von kryptographischen Funktionen für Atmel AVR Mikrocontroller zur Verfügung. Der Schwerpunkt der Bibliothek liegt dabei auf symmetrischen Kryptoverfahren wie DES oder AES, weiterhin umfasst sie *Keyed-Hash Message Authentication Codes* (*HMAC*) wie beispielsweise HMAC-SHA-256.

Für die asymmetrische Kryptographie auf Basis elliptischer Kurven existiert mit TinyECC [6] eine für Sensorknoten geeignete Bibliothek. Sie implementiert unter anderem das *Elliptic Curve Integrated Encryption Scheme* (*ECIES*), ein hybrides Verfahren. Dieses nutzt zum Austausch eines gemeinsamen Geheimnisses elliptische Kurven, die eigentliche Verschlüsselung erfolgt symmetrisch.

Die Bibliothek NaCl [1] bietet sowohl auf symmetrischer als auch auf asymmetrischer Kryptographie basierende Verschlüsselungs- und Signaturverfahren zur sicheren Netzwerkkomunikation. Bei ihrer Implementierung wurde großer Wert auf Sicherheit, Effizienz und Performanz gelegt. Insbesondere wurde darauf geachtet, Kontroll- und Datenflussabhängigkeiten zu geheimen Daten zu vermeiden um so einen Schutz vor Timing-Angriffen zu gewährleisten.

Tabelle I. ZUR EVALUATION VERWENDETE BIBLIOTHEKS- UND ANWENDUNGSVERSIONEN

µNaCl / AVRNaCl	20140813
TinyECC	2.0
AVR-Crypto-Lib	SVN rev. 5989
avr-gcc	4.8.1
avr-libc	1.8.0
TinyOS	2.1.2
Avrora+	1.7.117

A. Die NaCl Programmierschnittstelle

Mit `crypto_box` und `crypto_secretbox` bietet NaCl zwei Schnittstellen zur Absicherung der am häufigsten in der Netzwerkkommunikation geforderten Schutzziele: Vertraulichkeit, Integrität und Authentizität [7]. Beide kapseln eine vorgegebene, sichere Kombination an Primitiven, die eine authentifizierte Verschlüsselung (*Authenticated Encryption with Associated Data, AEAD*) bereitstellt. Implementierungsfehler wie eine unsichere Komposition kryptographischer Primitive von Seiten des Anwenders sind damit ausgeschlossen.

Die `crypto_secretbox` basiert auf symmetrischer Kryptographie und setzt daher voraus, dass beide kommunizierende Parteien einen gemeinsamen geheimen Schlüssel teilen. Sie kombiniert die Stromchiffre *Salsa20* [8] mit dem MAC *Poly1305* [9].

Ein asymmetrisches Kryptosystem wird durch die `crypto_box` realisiert. Sie beruht auf einem hybriden Verfahren bei dem die eigentliche Verschlüsselung von Daten ebenfalls durch die `crypto_secretbox` erfolgt. Der dazu erforderliche symmetrische Schlüssel wird mit dem *Elliptic-Curve Diffie-Hellmann Schlüsselaustauschverfahren* (*ECDH*) erzeugt. Dieses operiert bei NaCl auf der elliptischen Kurve *Curve25519* [10], welche als schnell und sicher gilt.

B. µNaCl / AVRNaCl

Das Projekt *µNaCl* [2] stellt eine Portierung von NaCl für Mikrocontroller bereit. Bisher unterstützt werden Atmel AVR Mikrocontroller der ATmega-Familie, Implementierungen für weitere Architekturen sind in Arbeit. µNaCl gibt es in zwei verschiedenen Varianten, die auf Geschwindigkeit respektive geringen Speicherbedarf hin optimiert sind.

III. EVALUATION

Im Rahmen der Evaluation wird die Implementierung von NaCl hinsichtlich der Energieeffizienz, der erforderlichen Rechenzeit und dem Speicherbedarf untersucht. Ein wesentlicher Augenmerk liegt dabei auf einem Vergleich mit den Bibliotheken TinyECC und AVR-Crypto-Lib. Die verwendeten Versionen sind in Tabelle I aufgeführt

A. Methodik

Zur Evaluation der Bibliotheken wurden TinyOS-Anwendungen implementiert, welche die zu untersuchenden kryptographischen Operationen verwenden. Bei AVRNaCl handelte es sich dabei um die auf Geschwindigkeit optimierten Funktionen. Als Hardwareplattform wurden MICAz-Sensorknoten gewählt. Um äußere Störeinflüsse auszuschließen und identische Vergleichsbedingungen zu schaffen, erfolgte die Evaluation im Simulator Avrora+ [3]. Dieser erzielt durch die Emulation der einzelnen Bestandteile drahtloser Sensorknoten realitätsnahe Ergebnisse. Insbesondere entspricht das Laufzeitverhalten des emulierten Prozessorkerns dem echter Hardware.

Tabelle II. AVRORA+ MODELLPARAMETER FÜR MICAZ

Parameter	Wert
Taktfrequenz	7,3728 MHz
Versorgungsspannung	3,309 V
Stromaufnahme	9,357 mA

Tabelle III. RECHENAUFWAND FÜR DIE VER- UND ENTSCHLÜSSELUNG MIT `CRYPTO_SECRETBOX` (SYMMETRISCH) UND `CRYPTO_BOX` (ASYMMETRISCH)

Verfahren	**Anzahl Zyklen**	**Dauer**	**Energiebedarf**
Sym. Verschlüsseln	92'563	0,013 s	0,5 mJ
Sym. Entschlüsseln	126'972	0,017 s	0,7 mJ
Asym. Verschlüsseln	22'904'034	3,105 s	125,6 mJ
Asym. Entschlüsseln	22'938'445	3,110 s	125,8 mJ

Für jede der Anwendungen wurde die zur Ausführung der einzelnen Operationen erforderlichen Zyklenzahl erfasst. Aus dieser können, aufgrund der Architektur des bei MICAz eingesetzten Atmel AVR Mikrocontrollers, mit den in Tabelle II angegebenen Modellparametern die entsprechenden Werte für Rechenzeit und Energiebedarf direkt ermittelt werden.

B. crypto_secretbox und crypto_box im Vergleich

Für die Gegenüberstellung von `crypto_secretbox` und `crypto_box` wurde eine Nachricht von 128 Byte verbeziehungsweise entschlüsselt. Zum Vergleich: Die maximale Größe einer IEEE 802.15.4-Dateneinheit beträgt 127 Byte.

Wie die Ergebnisse in Tabelle III zeigen, führt die mit asymmetrischer Kryptographie realisierte `crypto_box` deutlich aufwändigere Berechnungen durch als `crypto_secretbox`. Da beide Nutzungsmodi die Daten nach dem gleichen Verfahren verschlüsseln, ergibt sich der Unterschied von über Faktor 200 aus dem bei `crypto_box` genutzten Schlüsselaustauschverfahren auf Basis elliptischer Kurven.

C. Vergleich der symmetrischen kryptographischen Primitive aus NaCl

Neben der authentifizierten Verschlüsselung mit der `crypto_secretbox` ermöglicht NaCl auch den direkten Zugriff auf die einzelnen Primitive. Zur Verschlüsselung kommt Salsa20 zum Einsatz, als Hashfunktion wird SHA-512 genutzt. Zum Vergleich der einzelnen Funktionen untereinander und zur Untersuchung des Skalierungsverhaltens wurden diese auf verschiedene Datenblöcke von 8 bis 256 Byte angewandt. Grafik 1 zeigt die Taktzyklen, die zur Bildung eines SHA-512 Hashwerts, eines HMAC-SHA-512-256 Authentifikators sowie zu dessen Prüfung erforderlich waren. Weiterhin dargestellt ist der Berechnungsaufwand für die Ver- und Entschlüsselung mit Salsa20 sowie mit der `crypto_secretbox`.

Auffällig ist zum einen der große Geschwindigkeitsunterschied zwischen den SHA-512-basierten Funktionen und der mit Salsa20 und Poly1305 arbeitenden `crypto_secretbox`. Obwohl letztere Authentifizierung und Verschlüsselung bietet benötigt sie dafür eine deutlich

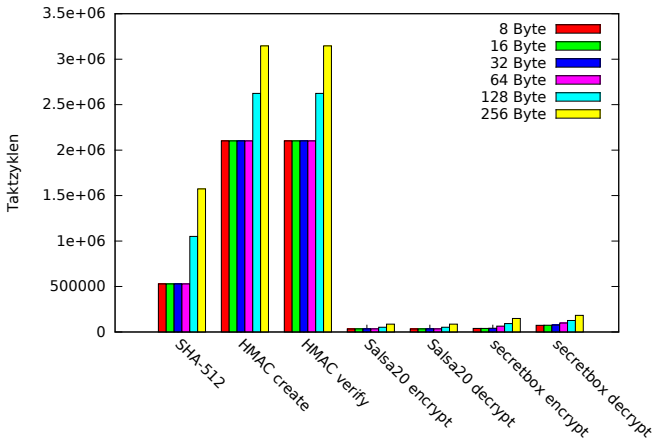

Abbildung 1. Skalierungsverhalten verschiedener symmetrischer kryptographischer Primitive aus NaCl

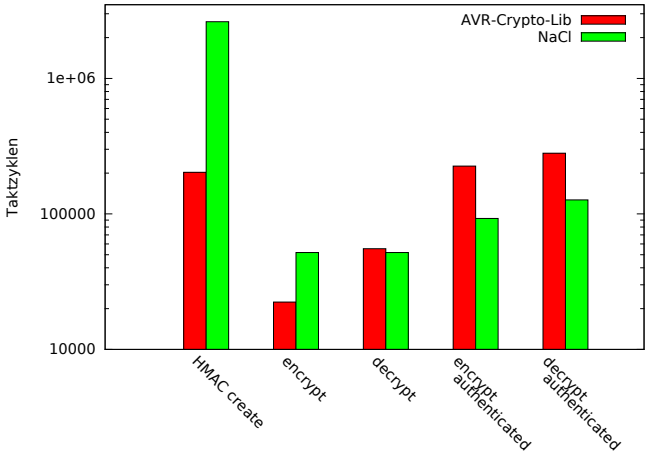

Abbildung 2. Rechenaufwand für die symmetrische Verschlüsselung von 128 Byte mit NaCl und der AVR-Crypto-Lib

geringere Anzahl an Taktzyklen und ist damit für den Einsatz in drahtlosen Sensornetzen besser geeignet. Weiterhin zeigt sich, dass die `crypto_secretbox` gut mit der Nachrichtenlänge skaliert. Der Berechnungsaufwand für die auf SHA-512 aufbauenden Funktionen steigt, durch die Arbeitsweise der Hashfunktion, in diskreten Stufen. SHA-512 verarbeitet Eingaben in Blöcken von je 1024 Bit. Aufgrund von Padding-Mechanismen erfolgt der Sprung von einem zu zwei Blöcken, mit einer entsprechenden Zunahme der Berechnungszeit, nicht erst bei einer Nachrichtengröße von 256 Byte sondern bereits bei 128 Byte. Diese Eigenschaft ist auch anhand der Graphen nachzuvollziehen.

D. Vergleich mit der AVR-Crypto-Lib

Im Folgenden werden SHA-256 als HMAC und AES-128-ECB als Verschlüsselungsverfahren aus der AVR-Crypto-Lib mit dem HMAC-SHA-512-256 und Salsa20 aus NaCl verglichen. Die `crypto_secretbox` aus NaCl realisiert eine authentifizierte Verschlüsselung. Um die gleichen Schutzziele mit der AVR-Crypto-Lib zu erreichen wurde AES mit dem HMAC-SHA256 kombiniert (*Encrypt-then-MAC*). Verarbeitet wurden jeweils Nachrichten mit einer Länge von 128 Byte.

Die für die verschiedenen Operationen benötigte Anzahl an Taktzyklen ist in Abbildung 2 dargestellt. Auffällig ist ein großer Geschwindigkeitsunterschied im Zusammenhang mit der HMAC-Implementierung in NaCl. Ursächlich dafür ist die Nutzung von SHA-512 als zugrundeliegende kryptographische Hashfunktion. HMAC-SHA512-256 verwirft eine Hälfte des Berechnungsergebnisses und entspricht damit, bei deutlich höherem Rechenaufwand, dem Sicherheitsniveau von HMAC-SHA-256. Auch Salsa20 schneidet bei der Verschlüsselung gegenüber AES-128-EBC schlechter ab und benötigt ungefähr die doppelte Berechnungszeit. Die Entschlüsselung mit AES erfolgt aber deutlich langsamer als die Verschlüsselung [11], Salsa20 zeigt dieses Verhalten nicht und liegt hier gleichauf. Bei der authentifizierten Ver- und Entschlüsselung ist die `crypto_secretbox` von NaCl im Vorteil, der zusätzliche Authentifikator ist mit Poly1305 bei nur geringem Mehraufwand zu berechnen.

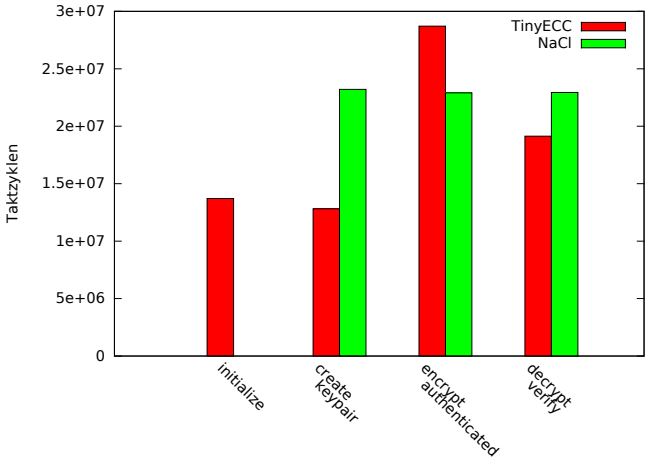

Abbildung 3. Rechenaufwand für die asymmetrischer Verschlüsselung von 128 Byte mit NaCl und TinyECC

E. Vergleich mit TinyECC

Für den Vergleich mit der `crypto_box` wurde die ECIES-Implementierung von TinyECC herangezogen. Diese arbeitet ähnlich wie die `crypto_box` in NaCl mit einem hybriden Schema und entspricht durch die Nutzung des HMAC-SHA1 dem Schutzniveau der durch NaCl realisierten authentifizierten Verschlüsselung. Für den Vergleich wurden Nachrichten mit einer Länge von 128 Byte verarbeitet.

In Abbildung 3 dargestellt ist der Berechnungsaufwand zur Initialisierung der Bibliothek, ein Schritt der nur bei TinyECC einmalig erfolgt. Weiterhin zeigt die Grafik die Anzahl der Taktzyklen, welche zum Erzeugen des Schlüsselpaares beziehungsweise zum Ver- und Entschlüsseln einer Nachricht erforderlich sind. Im direkten Vergleich sind die Unterschiede zwischen TinyECC und NaCl geringer ausgeprägt als bei NaCl und der AVR-Crypto-Lib. Der Aufwand für Initialisierung der Bibliothek und Erzeugung des Schlüsselpaares ist bei TinyECC rund 15% höher als bei NaCl. Auch die Verschlüsselung benötigt ungefähr 25% mehr Rechenzeit, im Gegensatz dazu ist der Aufwand zur Entschlüsselung und Authentifizierung bei NaCl aber knapp 20% größer.

Tabelle IV. SPEICHERBEDARF VON AVR-CRYPTO-LIB, NACL UND TINYECC

Bibliothek	Programmspeicher	Arbeitsspeicher
NaCl small	22 948 Byte	1613 Byte
Nacl fast	31 142 Byte	1485 Byte
AVR-Crypto-Lib	5482 Byte	1831 Byte
TinyECC	18 850 Byte	2249 Byte

F. Speicherbedarf der drei Bibliotheken

Eine Testanwendung zur Ver- und anschließenden Entschlüsselung einer Nachricht von 128 Byte wurde mit allen drei Bibliotheken implementiert. Dabei ergab sich der in Tabelle IV aufgeführte Bedarf an Programm- und Arbeitsspeicher. Letztere Angabe bezieht sich auf den statisch allozierten Speicher, zur Laufzeit werden weitere Variablen dynamisch auf dem Stack abgelegt. Bei NaCl wurden sowohl die `crypto_box` als auch die `crypto_secretbox` in der Testanwendung aufgerufen, der Unterschied zur getrennten Betrachtung war vernachlässigbar gering.

Im Vergleich mit den anderen Bibliotheken fällt auf, dass NaCl einen etwas geringeren Arbeitsspeicherbedarf hat, aber mehr Programmspeicher benötigt. Gegenüber der AVR-Crypto-Lib belegt NaCl annähernd die sechsfache Menge an Programmspeicher, der Unterschied zu TinyECC ist vergleichsweise gering.

IV. ZUSAMMENFASSUNG UND AUSBLICK

Die große Stärke von NaCl ist, von Implementierungsaspekten wie der gezielten Vermeidung von Seitenkanälen abgesehen, vor allem die hohen Geschwindigkeit bei der authentifizierten symmetrischen Verschlüsselung. Insbesondere benötigt diese weniger Zeit als die bloße Berechnung einer HMAC-SHA-256 mit der AVR-Crypto-Lib. Gegenüber TinyECC gestalteten sich die Ergebnisse ausgeglichen. Gleichzeitig ist aber auch festzustellen, dass NaCl deutlich mehr Programmspeicher erfordert als die beiden anderen betrachteten Bibliotheken.

Gegenwärtig befindet sich mit *ChaCha20* [12] eine weiterentwickelte Form von Salsa20 im Standardisierungsprozess als Verschlüsselungsverfahren für das Protokoll *Transport Layer Security* (*TLS*). Auch für Curve25519 gibt es entsprechende Bemühungen. Ferner ist für TLS 1.3 vorgesehen, nur noch Chiffren mit authentifizierter Verschlüsselung zuzulassen. Im Hinblick auf die Evaluationsergebnisse erscheinen Salsa20 (beziehungsweise ChaCha20 als Nachfolger) und Poly1305 für TLS 1.3 als gut geeignet.

TLS ist von einem zuverlässigen Transportprotokoll abhängig. Für den Einsatz in drahtlosen Sensornetzen ist *Datagram Transport Layer Security* (*DTLS*) daher besser geeignet. Beide Protokolle sehen eine Vielzahl von Verschlüsselungs- und Signaturverfahren vor, welche aber teilweise veraltet sind und nicht gezielt für Sensorknoten ausgewählt wurden. Die Arbeitsgruppe *DTLS in Constrained Environments* (*DICE*) der *Internet Engineering Taskforce* (*IETF*) beschäftigt sich daher unter anderem mit der Spezifikation eines auf ressourcenbeschränkte Systeme zugeschnittenen DTLS-Profils.

Über DICE hinaus geht das Konzept der Delegation aufwändiger Operationen, insbesondere des Schlüsselaustauschs, an leistungsstärkere Systeme im Netz, welches aktuell im Rahmen der IETF Arbeitsgruppe *Authentication and Authorization for Constrained Environments* (*ACE*) verfolgt wird. Inwiefern solche Ansätze tatsächlich den Energiebedarf verringern können ist unklar. Frühere Arbeiten zeigten, dass etwa beim Schlüsselaustauschprotokoll *Rich Uncle* die Wahl effizienter kryptographischer Verfahren wirkungsvoller als die Delegation von Berechnungen war [13].

Ausgehend von den Evaluationsergebnissen erscheint NaCl als gut geeignet für drahtlose Sensornetze. Für weitergehende Untersuchungen im Kontext von DTLS und DCAF sollte NaCl daher in eine DTLS-Implementierung integriert werden. Dabei stellt sich insbesondere die Frage, inwiefern DCAF den Energiebedarf von Sensorknoten zu reduzieren vermag und wie die Abwägung zwischen Kommunikations- und Rechenaufwand durch energieeffiziente Kryptoverfahren beeinflusst wird.

Danksagungen

Teile dieser Arbeit entstanden im Rahmen des, durch das Bundesministerium für Bildung und Forschung geförderten, Kompetenzzentrums für angewandte Sicherheitstechnologie (KASTEL, BMBF Förderkennzeichen 01BY1172)

LITERATUR

[1] D. J. Bernstein, T. Lange, und P. Schwabe, "The security impact of a new cryptographic library," in *Progress in Cryptology–LATINCRYPT 2012*. Springer, 2012, pp. 159–176.

[2] M. Hutter und P. Schwabe, "NaCl on 8-Bit AVR Microcontrollers," in *AFRICACRYPT*. Springer, 2013, pp. 156–172.

[3] C. Haas, J. Wilke, und V. Stöhr, "Realistic simulation of energy consumption in wireless sensor networks," in *Wireless Sensor Networks*. Springer, 2012, pp. 82–97.

[4] C. Haas, S. Munz, J. Wilke, und A. Hergenröder, "Evaluating Energy-Efficiency of Hardware-based Security Mechanisms," in *Proceedings of the 9th IEEE International Conference on Pervasive Computing and Communications Workshops (PerSeNS)*. IEEE, Mär. 2013, pp. 560–565.

[5] das labor, "Avr-crypto-lib." [Online]. Abzurufen unter: http://avrcryptolib.das-labor.org

[6] A. Liu und P. Ning, "TinyECC: A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks," in *Proceedings of the 7th International Conference on Information Processing in Sensor Networks*, ser. IPSN '08. IEEE Computer Society, 2008, pp. 245–256.

[7] D. J. Bernstein, "Cryptography in NaCl," Mar. 2009.

[8] D. J. Bernstein, "The Salsa20 Family of Stream Ciphers," in *New Stream Cipher Designs*, ser. Lecture Notes in Computer Science, M. Robshaw und O. Billet, Eds. Springer, 2008, vol. 4986, pp. 84–97.

[9] D. J. Bernstein, "The Poly1305-AES Message-authentication Code," in *Proceedings of the 12th International Conference on Fast Software Encryption*, ser. FSE'05. Springer-Verlag, 2005, pp. 32–49.

[10] D. J. Bernstein, "Curve25519: New Diffie-Hellman Speed Records," in *Public Key Cryptography - PKC 2006*, ser. Lecture Notes in Computer Science, M. Yung, Y. Dodis, A. Kiayias, und T. Malkin, Eds. Springer, 2006, vol. 3958, pp. 207–228.

[11] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, und N. Ferguson, "Performance comparison of the AES submissions," in *Second AES Candidate Conference*. National Institute of Standards and Technology, 1999, pp. 15–34.

[12] D. J. Bernstein, "ChaCha, a variant of Salsa20," January 2008.

[13] C. Haas, J. Wilke, und F. Knittel, "Evaluating the Energy-Efficiency of the Rich Uncle Key Exchange Protocol in WSNs," in *Proceedings of the 38th IEEE Conference on Local Computer Networks (LCN)*. IEEE, Okt. 2013.