Effiziente Implementierung von Public-Key-Algorithmen für Sensornetze

Erik-Oliver Blaß
Holger Junker
Martina Zitterbart

[blass|junker|zit]@tm.uka.de

Institut für Telematik, Universität Karlsruhe
April 2005

Abstract: Sicherheit in Sensornetzen ist ein schwieriges Problem, da diese winzigen Sensoren extrem ressourcenbeschränkt sind. Typischerweise basieren Sensoren auf batteriebetriebenen 8 Bit Mikrokontrollern, wie die bekannten MICA-Motes der Universität Berkeley. Die Entwicklung neuer sicherer Protokolle stellt sich so als äußerst schwierig dar, weil im Allgemeinen unklar ist, ob die dafür notwendigen Sicherheitsbausteine, kryptographische Algorithmen, überhaupt auf der limitierten Hardware einsetzbar sind. Gerade *asymmetrische* Algorithmen gelten bisher als zu teuer und unmöglich auf der schwachen Hardware zu realisieren. Im Gegensatz dazu stellt diese Arbeit effiziente Implementierungen einiger asymmetrischer Verfahren vor.

1 Einleitung

Die zunehmende Bedeutung von Sensornetzen wirft neue Fragen nach deren Sicherheit auf. Klassische Ansätze zum Schutz sicherheitskritischer und äußerst sensibler Daten gegen beispielsweise Abhören oder Verändern, die aus der LAN-Welt bekannt sind, scheitern jedoch in Sensornetzen: Sensoren müssen sich weitestgehend eigenständig und autonom organisieren und können sich nicht auf feste Infrastrukturkomponenten verlassen, so daß zentrale Einheiten wie Schlüsselverteilungs-Server oder *Certificat Authorities* (CA) undenkbar sind. Desweiteren ensteht ein großes Problem durch die extreme Hardware-Beschränktheit von Sensorknoten. Kleine, mit Batterien ausgerüstete Einheiten wie die MICA-Motes oder BTNodes verfügen nur über 4 KByte Hauptspeicher, 128 KByte Flash-ROM und leisten etwa 7 MIPS. Eine Entwicklung geeigneter Sicherheitsprotokolle ist extrem schwierig, da Sicherheitsbausteine wie kryptographische Chiffren, Hash- und Signatur-Funktionen dafür notwendig, diese jedoch in Bezug auf Energie- und Speicherverbrauch als äußerst ressourcenhungrig bekannt sind. Gerade asymmetrische Public-Key-Algorithmen gelten deshalb in der Literatur als in Sensornetzen unmöglich zu realisieren.

Im Gegensatz dazu stellt diese Arbeit die erfolgreiche und effiziente Implementierung

mehrerer asymmetrischer Algorithmen für den Atmel Atmega 128L Mikrokontroller, Basis u. a. der MICA-Sensoren und BTNodes, vor. Im Gegensatz zur allgemein geläufigen Meinung ist es gelungen, Public-Key Kryptographie *effizient* auf der Sensor-Hardware umzusetzen – damit werden komplexere und elegante Protokollkonstrukte wie Signaturen oder Zero-Knowledge-Beweise auch in Sensornetzen möglich. Zusätzlich zur Implementierung mehrerer Verfahren beschreibt diese Arbeit in Ergänzung zu [BZ05] die jeweils notwendigen *Kosten*, wie RAM- und ROM-Verbrauch sowie CPU-Zeit und damit den Energieverbrauch. Der ohne größeren Aufwand auf andere Plattformen portierbare Quellcode ist unter einer Open-Source-ähnlichen Lizenz erhältlich, so daß die Forschungsgemeinschaft z. B. zur Realisierung eigener Sicherheitsprotokolle davon profitieren kann.

2 Literatur

Ein klassischer Diffie-Hellman (DH) Schlüsselaustausch mit nur 512 Bit Schlüsseln dauert laut [Wat03] auf der Atmel Plattform insgesamt 315 Sekunden. In [GPW03] benötigt die einfache Verifikation einer Signatur mit RSA bei 1024 Bit Schlüssellänge schon 83 Sekunden – die Zeit für eine deutlich teurere Signatur-Generierung wird erst gar nicht gemessen. Die Arbeit [KSW] kommt bei 1024 Bit Schlüsseln immerhin mit 54,9 Sekunden für einen DH aus. In [Mal04] findet moderne elliptische Kurven Kryptographie Anwendung – gleichwohl fallen für einen DH immer noch 34 Sekunden an. Das Resultat dieser Arbeiten ist, daß aufgrund der enormen Rechenbelastung asymmetrische Verfahren als zu *teuer* für Sensornetze betrachtet werden. Dies hat zu der allgemeinen Meinung geführt, daß solche Algorithmen prinzipiell *unmöglich* in drahtlosen Sensornetzen zu realisieren sind. Daher werden in vielen neuen Protokollen Probleme, die eigentlich elegant mit asymmetrischer Kryptographie zu lösen sind, über komplizierte Umwege durch rein symmetrische Verfahren emuliert [ASW+02, EG02].

3 Asymmetrische Algorithmen

Die folgenden Abschnitte beschreiben einige technische Details der implementierten asymmetrischen Algorithmen, die für eine effiziente Umsetzung notwendig sind.

Eine reine Gegenüberstellung der Ergebnisse ist vorweg in Tabelle 1 dargestellt. Für den Atmega Prozessor gilt zudem, daß der Stromverbrauch direkt proportional zur Rechenzeit ist und in keiner Weise mit dem ausgeführten Code zusammenhängt. Der gemessene Stromverbrauch liegt bei etwa 30 mA.

Implementiert worden sind die Verfahren RSA [RSA78], elliptische und hyperelliptische Kurven [Men93], XTR [LV00] sowie NTRU [O'R02]. RSA mit einer Schlüssellänge von 1024 Bit dient dabei als Referenzpunkt: Die Parameter bzw. Schlüssellängen aller anderen Algorithmen sind so gewählt, daß sie die selbe Sicherheit wie RSA mit 1024 Bit bieten. Somit sind die unterschiedlichen Kosten aus Tabelle 1 miteinander vergleichbar.

Algorithmus	Parameter	Operation	Zeit	ROM	RAM
RSA	$\|n\| = p^2q = 1024$	Signatur	18	30,53	3,70
ECC	$\|p\| = 160$	Punktmult.	5,55	59,35	0,79
HECC	$\|g\| = 2$, $\|q\| = 80$	Divisormult.	9,1	61,37	0,76
XTR	$\|p\| = 170$, $\|q\| = 160$	$S_n(c)$	0,35	60,94	0,90
NTRU	$\|N\| = 167, \|p\| = 3, \|q\| = 128$	Polynommult.	0,40	5,71	0,13

Tabelle 1: Zeit- und Speicherkosten in s bzw. $KByte$

Die in der Tabelle aufgeführten *Operationen* der einzelnen Verfahren sind diejenigen, welche notwendig und jeweils am Teuersten sind, um beispielsweise eine Signatur, einen DH-Schlüsselaustausch oder einen Ver- bzw. Entschlüsselungsschritt durchzuführen.

Als Basis der Implementierung ermöglicht das freie Projekt MIRACL [Sco04] durch seine extreme Modularität und Flexibilität ein *Zuschneidern* der für die Algorithmen benötigten Langzahlarithmetik bzw. Arithmetik in endlichen Körpern.

3.1 RSA

Der Zeitbedarf der für RSA notwendigen modularen Exponentiation mit dem typischerweise zum Einsatz kommenden Verfahren von Montgomery [Mon85] benötigt bei einer für den praktischen Einsatz sicheren Schlüssel- bzw. Operandenlänge von 1024 Bit mehrere Minuten. Darüberhinaus stellt sich schon alleine die Größe der Operanden auf Sensoren als Speicherproblem dar. Daher empfiehlt sich der Einsatz einer hochoptimierten Variante, Multi-Exponent-RSA. Dieses verwendet als Modulus z. B. $n = p^2q$ und erzielt zusammen mit der Anwendung des chinesischen Restesatzes, Garners Algorithmus sowie Hensel-Lifting [MvOV97] eine Laufzeitverbesserung um Faktor 13 verglichen mit Standard-RSA. In bestimmten Fällen leidet dadurch jedoch die Sicherheit von RSA, Details dazu siehe [Lab].

3.2 Elliptische Kurven (ECC) und Hyperelliptische Kurven (HECC)

Eine Einführung in die relativ neue Technik der elliptischen Kurven findet sich z. B. bei Menezes [Men93]. Die kostenintensive Operation bei elliptischen Kurven ist die *Punktmultiplikation* – u. a. Grundlage für einen DH-Schlüsselaustausch. Elliptische Kurven der Form $y^2 = x^3 + Ax^2 + B$ über $GF(p)$ können wahlweise mit projektiven oder affinen Koordinaten verwendet werden, projektive Koordinaten haben sich auf dem Atmega 128L als schnellere Variante herausgestellt. Durch eine Normalisierung der projektiven Koordinaten kann die Punktmultiplikation nochmals erheblich beschleunigt werden.

Zudem ist auf Grundlage der Arbeiten von Wollinger [Wol04] eine Implementierung hyperelliptischer Kurven erfolgt. Durch das Verwenden *expliziter Formeln* für die Operatio-

nen Punktaddition und Punktverdopplung auf speziellen hyperelliptischen Kurven wird die erforderliche Arithmetik für die kritische *Divisormultiplikation* realisiert. Die zu RSA-1024 Bit erforderliche Operandenlänge von $q = p^m$ für Zahlen aus $GF(q)$ liegt mit *Genus* $g = 2$ der Kurve bei 80 Bit und produziert die schnellsten Ergebnisse. Für höhere und damit wahrscheinlich noch effizientere Geni $g > 4$ existieren zur Zeit noch keine expliziten Formeln.

3.3 XTR

Die bei XTR kritische Operation ist das Berechnen der Spur $S_n(c)$. Damit kann u. a. ein DH-Schlüsselaustausch durchgeführt werden. XTR verwendet eine besondere Darstellung für Gruppenelemente aus $GF(p^6)^*$ mit Elementen aus $GF(p^2)^*$ durch Ausnutzung der Spurabbildung. Dadurch wird eine Reduktion der Länge der verwendeten Operanden um den Faktor $1/3$ erreicht, so daß zur Implementierung Operanden mit nur etwa 170 Bit Breite zur Äquivalenz mit RSA-1024 Bit ausreichen. Hier bieten sich dann sowohl die Techniken von Montgomery wie auch von Comba [Com90] an, um die Multiplikation schnell zu bewerkstelligen. Eine weitere Optimierung bzw. Vereinfachung der Arithmetik ergibt sich aus der Verwendung bestimmter Primzahlen [LV00].

3.4 NTRU

Als Leitfaden bei der Implementierung des NTRU-Algorithmus dient O'Rourke [O'R02], da der ursprünglich veröffentlichte Quellcode wegen Kommerzialisierung nicht mehr verfügbar ist. Die bei NTRU entscheidende Operation ist die *Polynommultiplikation* zweier Polynome mit Koeffizienten modulo p bzw. q modulo des Polynoms $G = X^N - 1$. Diese Operation wird durch eine effiziente Modifikation von Baileys [BC01] Verfahren realisiert. Die Verwendung von Polynomen mit binären Koeffizienten und der Eigenschaft, daß sich Polynome in drei Teile mit jeweils kleinem Hamming-Gewicht gemäß $f(x) = f_1(x) * f_2(x) + f_3(x)$ aufteilen, beschleunigen ebenfalls die Berechnung. Neben der Polynom-Multiplikation ist es erforderlich, Inverse zu gegebenen Polynomen zu berechnen. Dazu werden Modifikationen der Verfahren zum Invertieren gemäß Silverman [Sil99] verwendet. Darüberhinaus kommen zusätzliche Optimierungen nach [HS00] zum Einsatz.

4 Zusammenfassung

Diese Arbeit präsentiert die effiziente Implementierung moderner Public-Key-Algorithmen für Mikrokontroller-basierte Sensoren. Im Gegensatz zur bisherigen Meinung sind solche Verfahren durchaus auf solch beschränkter Hardware einsetzbar und deren Kosten mit Ausnahme von RSA überschaubar gering. Gerade Verfahren wie XTR oder NTRU mit Laufzeiten von deutlich unter einer Sekunde und wenigen hundert Bytes Speicher-

verbrauch ermöglichen den Entwurf neuer Protokolle für Sicherheit in Sensornetzen, die asymmetrische Primitive wie Signaturen nicht mehr über Umwege durch komplexe symmetrische Konstrukte emulieren müssen, sondern Sicherheitsprobleme elegant lösen.

Der entwickelte Quellcode ist komplett portabel in Hochsprache geschrieben und steht unter einer Open-Source-Lizenz zur Verfügung. Bei Interesse können Anfragen hierzu an `blass@tm.uka.de` gerichtet werden.

Literatur

[ASW+02] A.Perrig, R. Szewczyk, V. Wen, D. Culler und J. D. Tygar. SPINS: Security Protocols for Sensor Networks. Wireless Networks Journal, 2002.

[BC01] D. Bailey und D. Coffin. NTRU in Constrained Devices. In *Workshop on Cryptographic Hardware and Embedded Systems*, 2001.

[BZ05] E.-O. Blaß und M. Zitterbart. Towards Acceptable Public-Key Encryption in Sensor Networks. In *The 2nd International Workshop on Ubiquitous Computing*, 2005.

[Com90] P. G. Comba. Exponentiation Cryptosystems on the IBM PC. *IBM Systems Journal*, 1990.

[EG02] L. Eschenauer und V. Gligor. A Key Management Scheme for Distributed Sensor Networks. Computer and Communications Security, 2002.

[GPW03] N. Gura, A. Patel und A. Wander. Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs, 2003. `http://www.research.sun.com/projects/crypto`.

[HS00] J. Hoffstein und J.H. Silverman. Optimizations for NTRU. In *Public Key Cryptography and Computational Number Theory*, 2000.

[KSW] C. Karloff, N. Saastry und D. Wagner. TinySec – Technical Report. `http://www.cs.berkeley.edu/~nks/tinysec`.

[Lab] RSA Laboratoires. A Cost-Based Security Analysis of Symmetric and Asymmetric Key Lengths – Technical Report. `http://www.nullify.org/docs/bulletin13/bulletin13.html`.

[LV00] A.K. Lenstra und E.R. Verheul. The XTR public key system. In *LNCS Crypto*, 2000.

[Mal04] D. Malan. Crypto for Tiny Objects – Technical Report. `http://airclic.eecs.harvard.edu/publications/tr-04-04.pdf`, 2004.

[Men93] A.J. Menezes. *Elliptic Curve Public key Crtyptosystems*. Kluwer, 1993.

[Mon85] P. Montgomery. Modular Multiplication Without Trial Division. *Mathematics of Computation*, 1985.

[MvOV97] A. Menezes, P. van Oorschot und S. Vanstone. Handbook of Applied Cryptography, 1997.

[O'R02] C.M. O'Rourke. Efficient NTRU Implementations. Diplomarbeit, Worcester Polytechnic Institute, 2002.

[RSA78] R. L. Rivest, A. Shamir und L. M. Adelman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM Vol. 21, 1978.

[Sco04] M. Scott. MIRACL library. `http://indigo.ie/~mscott`, 2004.

[Sil99] J.H. Silverman. Almost Inverses and Fast NTRU Key Generation – Technical Report, 1999.

[Wat03] R. Watro. TinyPK – Energy Efficient Security for Wireless Sensor Networks – Technical Report, 2003. `http://www.motelab.org/papers/TinyPK.pdf`.

[Wol04] T. Wollinger. *Software and Hardware Implementation of Hyperelltiptic Cuve Cryptosystems*. Dissertation, Ruhr-Universität Bochum, 2004. `http://crypto.rub.de`.