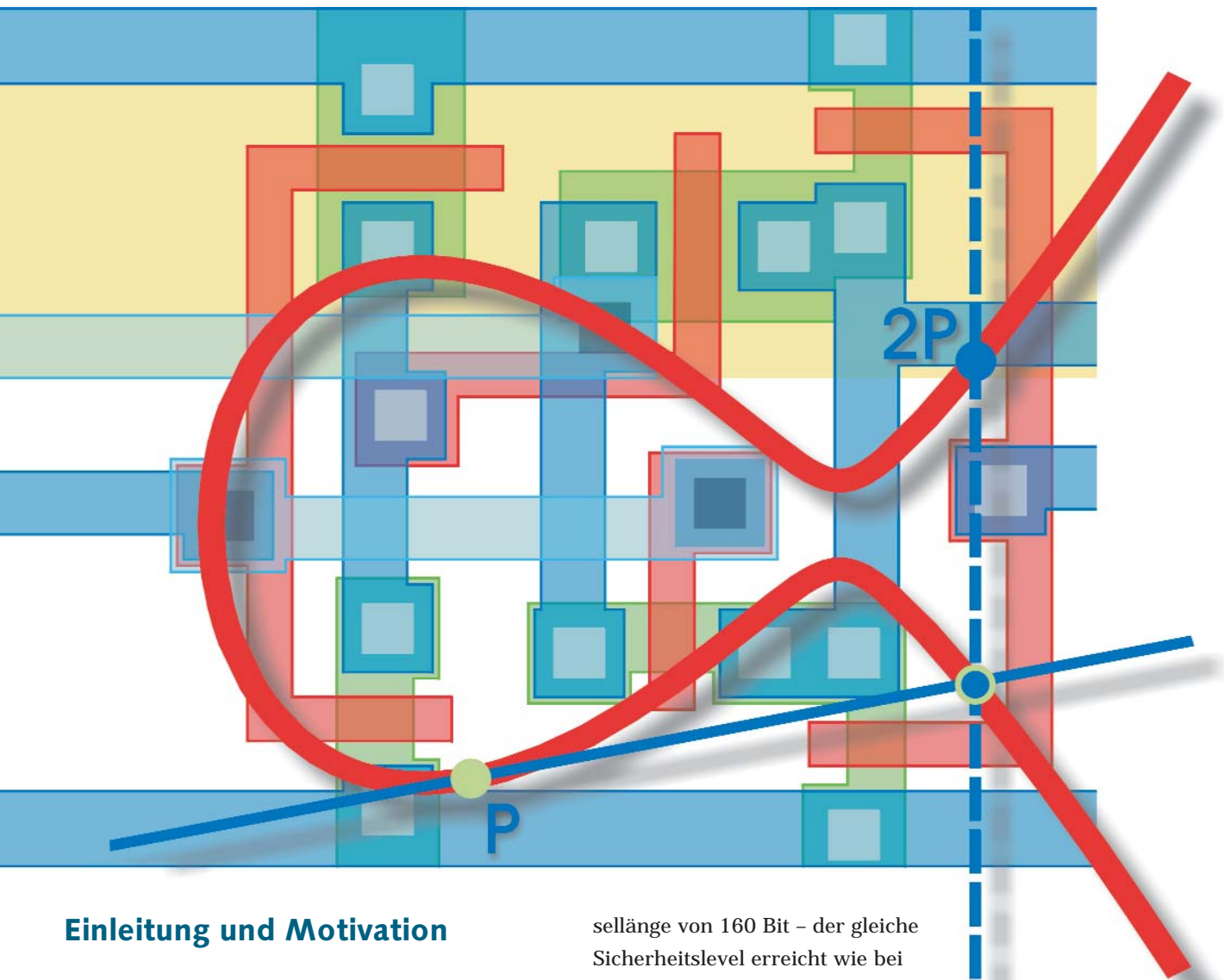


Elliptische Kurven Kryptographie – hocheffizient und portabel

PROF. DR. RER. NAT. JOHANNES BUCHMANN, DIPL.-INFORM. MARKUS ERNST,
DIPL.-MATH. BIRGIT HENHAPL, PROF. DR.-ING. SORIN HUSS



Einleitung und Motivation

Elliptische Kurven haben im Bereich der Public-Key-Kryptographie in jüngster Zeit viel Aufmerksamkeit erregt. Auch wenn vielversprechende Verfahren basierend auf Zahlkörpern quadratischer Ordnung [1] oder auf Gitter-Basis-Transformationen entwickelt werden, so sind die Elliptische-Kurven-Kryptoverfahren die bisher einzig standardisierte Alternative zu RSA. Ihr Einsatz wurde von N. Koblitz [2] und V. Miller [3] erstmals vorgeschlagen. Ihr Vorteil liegt darin, dass es keinen bekannten subexponentiellen Algorithmus gibt, der das zugrunde liegende Diskreter-Logarithmus-Problem über elliptischen Kurven löst. Als Folge daraus wird mit Elliptische-Kurven-Kryptographie – trotz kurzer Schlüs-

sellänge von 160 Bit – der gleiche Sicherheitslevel erreicht wie bei RSA-1024. Im Gegensatz zu den anderen oben genannten Verfahren hat die Elliptische-Kurven-Kryptographie auch in Standards bereits Einzug gehalten, wodurch einheitliche Implementierungen sowie Interoperabilität ermöglicht werden.

Unser Ziel ist es, der Allgemeinheit sichere und schnelle Kryptographie zur Verfügung zu stellen. Daher entschieden wir uns für elliptische Kurven. Neben der Sicherheit und der hohen Geschwin-

digkeit ist es außerdem wichtig, diese Kryptographie auch anwendbar zu machen. Dazu gehört, dass einerseits der Anwender zu Hause an seinem PC signieren und verschlüsseln kann – zum Beispiel mit Outlook – und andererseits die Leistung des Providers den hohen Anforderungen im High-End-Bereich genügt. Wir werden im Weiteren vorstellen, wie der Provider konzipiert und umgesetzt wurde.

Elliptic Curve Cryptography – highly efficient and portable

Public Key Cryptography is the basic technology for secure internet transactions with confidential information. Presently, security for most systems is maintained by the RSA procedure. Because computers are becoming more and more efficient, utilized keys require at least 1024 Bit in order to guarantee the necessary security. However, future developments are unforeseeable. Thus, it is very important to create alternatives to RSA in order to prevent possible security risks. At this time, we introduce a provider for cryptographic protocols, which has been developed at the Institute for Cryptography and Computer Algebra. This provider, which amongst other things utilizes the ECDSA procedure (a digital signature with elliptic curves), offers an equally high level of security as does RSA, however, with keys of 160 Bit instead of 1024 Bit. Also by using the programming language Java, this provider is flexible, user-friendly, and very efficient. Furthermore, this provider meets the highest demands for use with servers: On the basis of the Cryptoprocessor, developed at the Integrated Circuits and Systems Lab., 200 signatures/second can be verified with the implemented hardware acceleration. Thus, this provider can be set up for high-end environments as found in online banking businesses for instance.

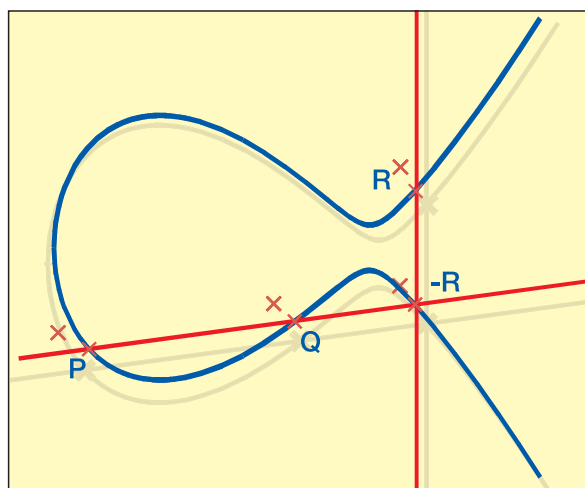


Abbildung 1:
 $R = P+Q$

Elliptische Kurven

Im Folgenden werden wir die elliptischen Kurven und die darauf möglichen Operationen bildlich veranschaulichen, um auch dem Leser ohne Vorkenntnisse der Algebra und der Kryptographie das Verständnis zu ermöglichen. Eine elliptische Kurve E ist eine kubische Gleichung mit zwei Variablen x und y . Als Beispiel betrachten wir die elliptische Kurve $E: y^2 = x^3 - 7x$ über dem Körper K . Das Paar (x, y) nennt man einen Punkt $P = (x, y)$ auf der elliptischen Kurve E . Die Punkte einer elliptischen Kurve können addiert oder voneinander subtrahiert werden. Abbildung 1 zeigt die Addition eines Punktes P zum Punkt Q auf der Kurve E , also $R = P+Q$.

Um die Punkte P und Q zu addieren, legt man eine Gerade durch sie hindurch. Diese schneidet die Kurve an einem weiteren Punkt $-R$. Eine Punktaddition ist also keine triviale Operation, sie benötigt mehrere Rechenschritte, die Additionen, Multiplikationen und Quadrierungen umfassen.

Nimmt man zu der Menge der Punkte noch einen Punkt hinzu, den *Punkt im Unendlichen*, genannt O , erhält man eine additive Gruppe. Der Punkt O ist für die Gruppenstruktur notwendig. Er ergibt sich immer dann, wenn die

Gerade senkrecht liegt, das heißt, wenn sie die Kurve nicht an einer dritten Stelle schneidet. Eine Gruppe ist eine Menge, die sich durch die folgenden drei Eigenschaften – hier veranschaulicht an elliptischen Kurven – auszeichnet:

1. Das Ergebnis einer Operation in der Gruppe ist wieder ein Element der Gruppe: liegen P, Q auf der Kurve E , dann liegt auch $P+Q$ auf E .
2. Es existiert ein *neutrales Element*, der Punkt im Unendlichen O mit $P+O = P$.
3. Jedes Element besitzt ein dazu *inverses Element* mit $P + (-P) = O$.

Die Punktaddition kann wiederholt angewandt werden, die wiederholte Punktaddition nennen wir eine *Punktvervielfachung*. So ist $R = P+P+\dots+P+P = k \cdot P$, wenn P k -mal zu sich selbst addiert wird.

Man nennt k den *diskreten Logarithmus von R zur Basis P* . Hat man ein P und ein R gegeben, ist es schwierig, den diskreten Logarithmus k zu berechnen.

In der Kryptographie verwendet man elliptische Kurven über endlichen Körpern. Das bedeutet, dass die Koeffizienten a und b der Kurve $E: y^2 = x^3 - ax + b$ sowie die der Punkte x und y Elemente des Körpers K sind und die Operationen, im Folgenden *Add*, *Square* und *Mult* genannt, ebenfalls in diesem Körper erfolgen. Ein Beispiel für einen Körper sind die *Primkörper $GF(p)$* : Für die Primzahl p bilden die Zahlen $\{0, \dots, p-1\}$ zusammen mit der Addition und Multiplikation modulo p einen Körper. Ein weiteres Beispiel ist der Körper mit 2^n Elementen $GF(2^n)$, der n -dimensionale Vektorraum über der Menge $\{0, 1\}$. Auf ihn werden wir uns bei der Erläuterung der Hardwareimplementierung beziehen.

Die Anzahl der Punkte einer Kurve E ist abhängig von dem zu

Grunde liegenden Körper K . Je mehr Elemente der Körper K besitzt, desto mehr Punkte können auf E liegen. Natürlich ist das Problem, den diskreten Logarithmus über elliptischen Kurven zu lösen, um so größer, je mehr Punkte auf der Kurve liegen. Die Anzahl der Punkte nennt man die *Ordnung* von E (abgekürzt $ord(E)$).

ECDSA

ECDSA ist ein digitales Signatur-Verfahren für elliptische Kurven. Digitale Signaturen gehören zur Public-Key-Kryptographie und ersetzen im IT-Bereich herkömmliche Unterschriften: Sie sollen die Authentizität des Signierers und die Integrität des Dokuments gewährleisten. Die Grundidee ist folgende: Jeder Teilnehmer besitzt einen privaten geheimen Schlüssel, dazu gibt es einen passenden öffentlichen „Gegenschlüssel“, den jeder kennt und verwenden kann. Ein mit dem privaten Schlüssel verschlossenes Dokument kann nur mit Hilfe dieses Gegenschlüssels wieder geöffnet werden. Verschließt also Person A ein Dokument mit dem privaten Schlüssel S_{PA} , kann das Dokument nur mit A's öffentlichen S_{oA} wieder geöffnet werden. Öffnen kann das Dokument jeder, da der Schlüssel öffentlich ist. Dabei, kann man sicher sein, dass A das Dokument verschlossen hat, da nur A den Schlüssel S_{PA} besitzt und weil kein anderer Schlüssel als Gegenstück funktioniert. So wird die Authentizität von Person A bewiesen.

Auch die Integrität des Dokuments ist sichergestellt, da das Verschließen und Öffnen des Dokuments aus Operationen besteht, die den Inhalt des Dokuments mit einbeziehen.

Aber auch wenn die Praxis auf den ersten Blick anders aussieht,

so ist die Grundidee doch noch vorhanden. So wird das Dokument nicht wirklich „verschlossen“. Vielmehr wird eine separate Signatur, die von dem zu signierenden Dokument abhängig ist, mit dem privaten Schlüssel erzeugt und zusammen mit dem Originaldokument verschickt. Anhand der Signatur und des öffentlichen Schlüssels kann dann festgestellt werden, ob der Absender der Signatur entspricht und ob der Inhalt des Dokuments noch original ist. Ist dies nicht der Fall, muss natürlich von einer Fälschung ausgegangen werden.

Die Abhängigkeit der Signatur von dem Dokument wird durch eine sogenannte Hashfunktion hergestellt: Eine *Hashfunktion* h bildet einen String m beliebiger Länge auf einen String fester Länge, den *Hashwert* $f = h(m)$, ab. Eine sichere Hashfunktion erfüllt die Eigenschaften, dass sie nicht umkehrbar ist, und dass man zu einem gegebenen Hashwert $f = h(m)$ keinen String m' findet, mit $h(m') = f$. Für ECDSA ist die Hashfunktion SHA1 fest vorgeschrieben.

Im Folgenden werden wir ECDSA grob skizzieren: Im Vorfeld der Schlüsselerzeugung und des Signierens beziehungsweise Verifizierens müssen die *EC Domain Parameter* bekannt sein: Der Körper K , die elliptische Kurve E mit ihrer Ordnung $ord(E)$, eine große ganze Zahl r , die ein Primteiler von $ord(E)$ ist, und weiterhin ein Punkt G auf E , für den gilt, dass $r \cdot G = O$.

Der *private Schlüssel* ist eine zufällige ganze Zahl s mit $1 < s < r$, der öffentliche Schlüssel ist $W = s \cdot G$. Die Schlüssellänge ergibt sich aus der Größe des zugrundeliegenden Körpers K : Für $K = GF(2^n)$ ist die Schlüssellänge n , für $K = GF(p)$ ist die Schlüssellänge $\log_2 p$.

Die **Signatur** wird von einem Paar ganzer Zahlen (c, d) , $0 < c, d < r$, gebildet. Gegeben sei nun eine Nachricht m , zusammen mit ihrem Hashwert f . Der Signierer muss zur Erzeugung seiner Signatur folgende Schritte ausführen:

1. Erzeugung eines Einmal-Schlüsselpaars (u, V) mit $1 < u < r$ und $V = u \cdot G = (x_V, y_V)$.
2. Berechnung: $c = x_V$ modulo r .
3. Berechnung: $d = u^{-1}(f + sc)$ modulo r . Ist $d = 0$: Neustart bei 1.

Die **Verifikation** gibt **true** zurück, wenn die Signatur korrekt ist, sonst **false**:

1. Liegen c und d nicht im Intervall $[1, r - 1]$: gib **false** zurück.
2. Berechnung: $h = d^{-1}$ modulo r ; $h_1 = fh$ modulo r und $h^2 = ch$ modulo r .
3. Berechnung: $P = h_1 \cdot G + h_2 \cdot W = (x_P, y_P)$. Ist $P = O$, gib **false** zurück. Ist $c = x_P$, gib **true** zurück.

Es ist klar, dass sowohl beim Signieren als auch beim Verifizieren die Punktvervielfachung die mit Abstand aufwendigste Operation ist. Genau diese gilt es zu beschleunigen.

KryptoProzessor

Software Implementierungen kryptographischer Verfahren haben generell den Nachteil, dass die n Bit breiten Körperoperationen auf einen Prozessor mit fester Wortlänge (z.B. 32-Bit Intel Pentium) abgebildet werden, was mit einem beträchtlichen Verlust an Performanz verbunden ist. Zur effizienten und auf hohen Datendurchsatz optimierten Implementierung der Punktvervielfachung $k \cdot P$ ist eine auf die Schlüssellänge n angepasste Hardwareimplementierung zwingend erforderlich. Durch die, bei einer Hardwarerealisierung mögliche, parallele Ausführung mehrerer Körperoperationen kann die Per-

formanz des Verfahrens nochmals erheblich gesteigert werden. Im Folgenden werden der am ISS entwickelte KryptoProzessor und die zugehörige Entwurfsmethodik vorgestellt. Für detailliertere Informationen verweisen wir auf [4].

Hardwareplattform

Die Implementierung des *Elliptic-Curve* KryptoProcessors basiert auf der Standard PCI-Karte *microEnable* (siehe Abbildung 2) der Fa. Silicon Software GmbH. Das Kernstück der Karte ist ein rekonfigurierbarer Logikbaustein (FPGA) der Fa. Xilinx Inc. In diesem FPGA wird die eigentliche Funktionalität des KryptoProcessors implementiert. Darüber hinaus stehen ein programmierbarer Taktgenerator (bis 120 MHz), statisches RAM und externe Schnittstellen zur Verfügung, so dass sich ganze Hardware-Subsysteme auf der Karte implementieren lassen. Die Einbindung ins Gesamtsystem erfolgt auf einfache Weise über das PCI-Interface und die zugehörige C-Programmierschnittstelle. Die PCI-Karte *microEnable* gibt es mit unterschiedlicher SRAM-Bestückung sowie mit FPGA-Bausteinen unterschiedlicher Komplexität.

VHDL Generator

Für den Entwurf und die Implementierung der KryptoProcessoren wird ein VHDL-gestützter Entwurfsablauf eingesetzt. VHDL ist der *de-facto* Standard für die abstrakte Modellierung digitaler Schaltungen. Zur funktionalen Modellierung synchroner, digitaler Schaltungen mit klaren Randbedingungen bzgl. Schaltungsgröße und Performanz sind VHDL-Beschreibungen auf *Register Transfer Level (RTL)* bestens geeignet. Dennoch reichen sie für den hier vorgesehenen Anwendungsbe- reich nicht aus.

Die Architektur des KryptoProcessors ist variabel in Bezug auf die Schlüssellänge *n* und den Parallelisierungsgrad *g*. Für eine konkrete Implementierung müssen diese beiden Parameter vom Systementwickler festgelegt werden. Zur automatisierten Erzeugung von parametrisierten Instanzen des KryptoProcessors wurde am ISS ein dediziertes Generatorprogramm entwickelt (siehe Abbildung 3). Das automatisch generierte VHDL-Modell des Prozessors bildet die Grundlage für die anschließende Hardware-Synthese und die weiteren Entwurfsschritte zur Er-

zeugung der FPGA-Konfigurationsdatei.

Der Generator-basierte Ansatz zur Erzeugung der VHDL-Beschreibung wurde gewählt, da die ausschließliche Modellierung der Funktionalität mit synthese-fähigen VHDL-Sprachkonstrukten nicht möglich ist. Dies liegt hauptsächlich an der speziellen Struktur des verwendeten Massay-Omura Multiplizierers, der im Wesentlichen aus einem sehr großen XOR-Baum besteht [5]. Zwei *n* Bit Eingangsvektoren werden auf diese Weise auf ein einzelnes Bit abgebildet, was sich besonders effizient in Hardware implementieren lässt. Da sich die Struktur dieses XOR-Baumes mit dem zugrundeliegenden endlichen Körper bzw. der Schlüssellänge *n* ändert, ist es nicht möglich, ein entsprechendes generisches und gleichzeitig synthese-fähiges VHDL-Modell anzugeben. Im rechten Teil von Abbildung 3 ist ein Ausschnitt des generierten VHDL-Modells der Massay-Omura Architektur für die Schlüssellänge *n=191* Bit dargestellt.

Mit unserem VHDL-Generator sind wir in der Lage, VHDL-Beschreibungen des KryptoProcessors für unterschiedliche Schlüs-

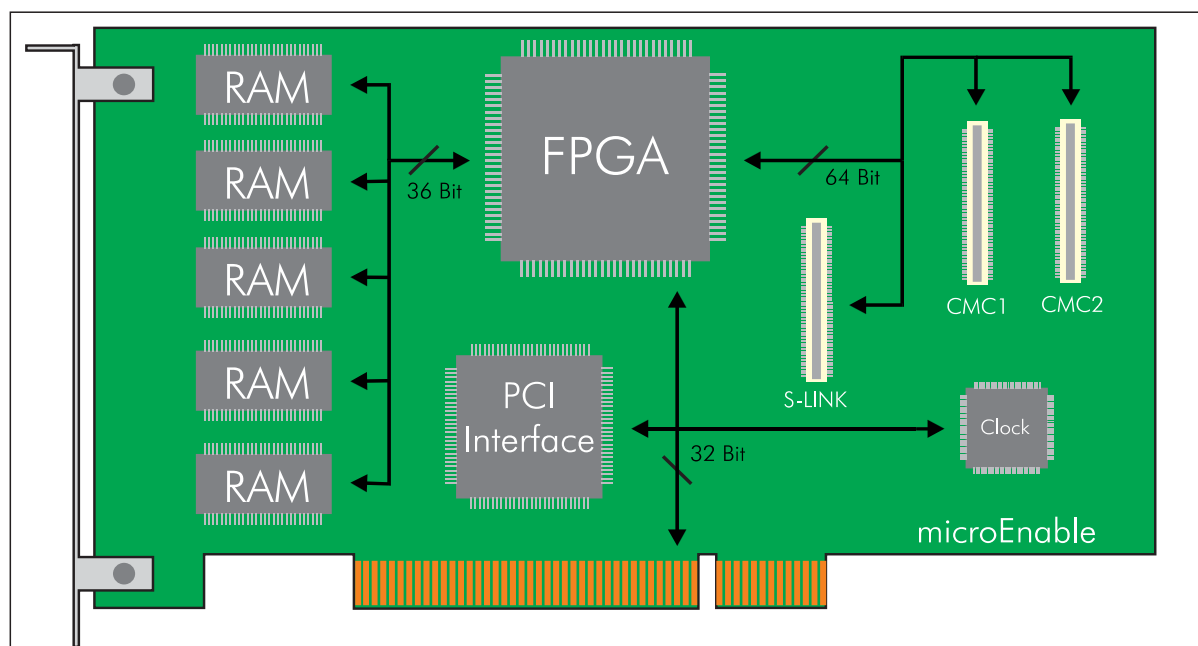
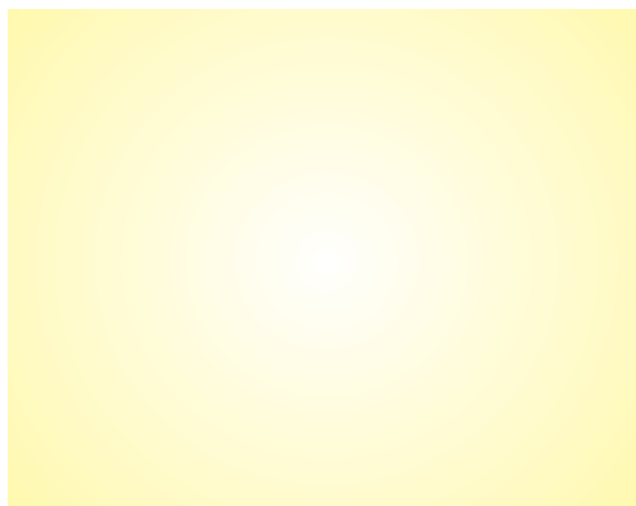
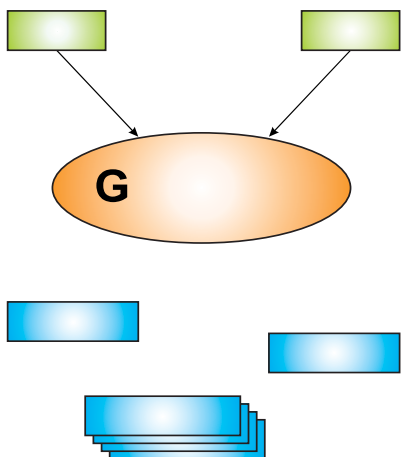


Abbildung 2: *microEnable* PCI-Karte

Abbildung 3:
VHDL Generator



sellängen zu erzeugen. Hierdurch ist es leicht möglich, zukünftig steigende Sicherheitsanforderungen, d.h. größere Schlüssellängen, zu unterstützen. Speziell für FPGA-basierte Zielplattformen ergeben sich weitere Vorteile durch den Generator gestützten Ansatz: Da auch der Parallelisierungsgrad parametrisierbar und somit variabel ist, lassen sich die jeweils verfügbaren FPGA-Ressourcen optimal ausnutzen, was wiederum die Voraussetzung zum Erreichen der maximalen Performance ist. Da die generierten VHDL-Modelle nicht auf eine spezielle FPGA-Familie zugeschnitten sind, ist es somit möglich den technologischen Fortschritt bei FPGAs (größere und schnellere

Bausteine) direkt in bessere Performance umzusetzen.

Auch in Bezug auf Qualitätssicherung und Validierung ergeben sich Vorteile durch die Generatorbasierte Vorgehensweise. Implementierungen für kleine Schlüssellängen (z.B. $n=18$ Bit) können für erschöpfende, funktionale Tests verwendet werden. Dies ist notwendig um sicherzustellen, dass der VHDL-Generator selbst richtig funktioniert. Anhand von praxisrelevanten KryptoProcessoren mit Schlüssellängen $n \leq 160$ sind wirklich erschöpfende Tests nicht durchführbar.

Prozessorarchitektur

Auf dem *Elliptic-Curve* Krypto-Processor ist der komplette Algo-

rithmus zur Berechnung der Punktvervielfachung in Hardware implementiert. Die Architektur des Prozessors ist in Abbildung 4 dargestellt und besteht im Wesentlichen aus drei funktionalen Blöcken.

Körper Arithmetik:

Hier sind die oben beschriebenen Körperoperationen implementiert. Für *Add* und *Square* steht jeweils eine Komponente zur Verfügung. Zur Berechnung von *Mult* können, wie bereits erwähnt, mehrere Massay-Omura Multiplizierer instanziiert werden. Dadurch lassen sich die jeweils verfügbaren FPGA-Ressourcen optimal ausnutzen und somit die maximale Performance erzielen.

Register File:

Der Prozessor verfügt über 16 n Bit Register zur Speicherung von Körperelementen bzw. von positiven ganzen Zahlen der entsprechenden Bitbreite.

Controller:

Im Controller ist der eigentliche Algorithmus zur Berechnung von $k \cdot P$ implementiert. Er ist als endlicher Automat realisiert und stellt das Steuerwerk des Krypto-Processors dar. Die in Abbildung 4 angegebenen externen Signale dienen zur Kommunikation mit dem KryptoProcessor und werden von der PCI-Schnittstelle bedient.

Abbildung 4:
Architektur des
KryptoProcessors
*Architecture
of the
CryptoProcessor*

FlexiECProvider

Wie bereits in der Motivation erläutert, ist es unser Ziel, sichere und effiziente Kryptographie für die Allgemeinheit zugänglich und anwendbar zu machen. Doch was heißt eigentlich „effizient“?

Ein typisches Anwenderszenario: Der normale Endbenutzer zu Hause an seinem PC, der mit Outlook oder Netscape Emails verschlüsselt und signiert verschicken will. Für ihn ist eine reine Softwarelösung ausreichend. Der Zeitaufwand von bis zu 500 ms ist völlig akzeptabel.

Ein anderes, ebenso typisches Anwenderszenario: Der Server einer Bank, der zum Beispiel Kontostands-Abfragen beantwortet. Bei jeder dieser Anfragen muss überprüft werden, ob sie auch berechtigt ist – eine Signatur muss verifiziert werden. Wenn man davon ausgeht, dass 100 Abfragen pro Sekunde eingehen, darf eine Verifikation nicht mehr als 10 ms dauern. Um solche Performanzwerte zu erreichen, ist dedizierte Verschlüsselungshardware unabdingbar. Ein weiterer Unterschied zwischen diesen beiden Szenarien besteht darin, dass sich die Investition in einen KryptoProzessor nur serverseitig (bei einer Bank), nicht aber clientseitig (beim Endbenutzer zu Hause) lohnen würde.

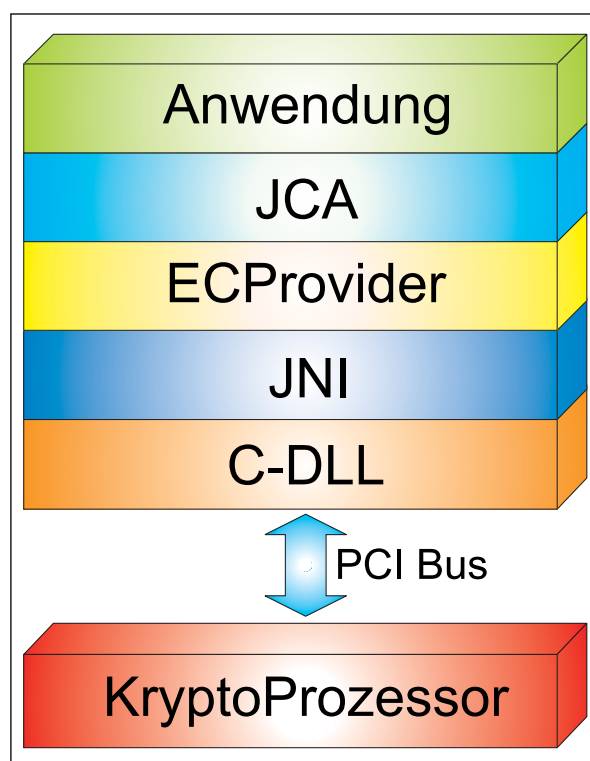
Um beiden Seiten gerecht zu werden, ist ein flexibles System, bestehend aus Software- und Hardwarekomponenten, notwendig. Die Software ist auf beiden Seiten identisch, d.h. Server und Client verwenden den selben Provider. Wird im Server zusätzlich der KryptoProzessor zur Beschleunigung der Punktvervielfachung installiert, dann wird das Erzeugen und Verifizieren von Signaturen in wenigen Millisekunden ermöglicht.

Der FlexiECProvider ist in Java geschrieben und kann auf jeder Plattform angewandt werden. Weiterhin bietet uns Java die *Java Cryptography Architecture (JCA)* und die *Java Cryptography Extension (JCE)*¹, ein Framework zur Benutzung und Entwicklung kryptographischer Verfahren in Java. Während Objekte zur Verschlüsselung, zur Hashwert-Berechnung und zu digitalen Signaturen über das *Application Programming Interface (API)* instanziiert und genutzt werden, ist das Framework selbst von spezifischen Kryptographie-Algorithmen frei. Diese werden von sogenannten *Cryptographic Service Providern* (nachfolgend schlicht *Provider* genannt) angeboten und durch die JCA bzw. die JCE zur Benutzung bereitgestellt. So können eigene oder andere bekannte Algorithmen in Form eines Providers implementiert werden. Die Schnittstellen zur JCA sind über das *Service Provider Interface (SPI)* gegeben.

Zur Klasse dieser Provider gehört der FlexiECProvider. Über die JCA werden von ihm die digitalen Signatur Algorithmen ECDSA und ECNR sowie das Schlüsselaustauschverfahren ECDH mit dem Verschlüsselungsverfahren ECIES zur Verfügung gestellt. Jedes dieser Verfahren ist konform zu dem IEEE Standard P1363 über Primkörpern als auch über Körpern der Charakteristik 2 implementiert und kann unter <http://www.flexiprovider.de> frei herunter geladen werden.

Wir werden nun am Beispiel der ECDSA-Signaturerzeugung und -verifikation skizzieren, wie der Provider zu verwenden ist.

¹ Aufgrund früherer US-Exportbeschränkungen musste die JCA zweigeteilt werden. Frameworks, die Verschlüsselungen enthielten, durften nicht exportiert werden.



Im Folgenden wird angenommen, dass bereits eine Instanz der Klasse *AlgorithmParameterSpec* sowie ein Schlüsselpaar *Key-Pair kp* vorhanden ist.

Signatur Erzeugung:

```
Signature sig =
Signature.getInstance(„ECDSA“);
sig.initSign();
sig.sign(kp.getPrivate());
```

Signatur Verifikation:

```
Signature sig =
Signature.getInstance(„ECDSA“);
sig.initVerify();
sig.verify(kp.getPublic());
```

Das Schema, nach dem die Operationen ablaufen, ist deutlich zu erkennen:

- Erzeugen einer Instanz des jeweiligen Objektes eines kryptographischen Dienstes (Signatur, Schlüsselpaar, Hashalgorithmus,...),
- Initialisieren dieser Instanz mit – hier – dem jeweiligen Schlüssel und
- Ausführen der Operation.

Unser JCA-basierter Provider funktioniert nach dem selben Prinzip. Das Initialisieren des KryptoProzessors sowie die Ausführung der *k•P* Operation in

Abbildung 5: ECD-SA Systemschichten mit Hardwarebeschleunigung
Hardware accelerated ECDSA system layers

FlexiECProvider Ziel Plattform	ECDSA über GF(2ⁿ) 191 Bit Signatur Erzeugung	ECDSA über GF(2ⁿ) 191 Bit Signatur Verifikation
Software- implementierung (AMD Athlon, 1 GHz)	24 ms	105 ms
KryptoProzessor (32 MHz, Paralleli- sierungsgrad 7)	3.4 ms	5.7 ms

Hardware geschieht intern und ist für den Anwender transparent. Da man von Java nicht direkt auf Hardware zugreifen kann und die *microEnable* PCI-Karte über eine vorgegebene C-DLL angesprochen wird, muss die Kommunikation zwischen Provider und Krypto-Prozessor, wie in Abbildung 5 dargestellt, über das Java Native Interface (JNI) erfolgen. In der Tabelle sind die Ergebnisse zusammengefasst.

Die reine Softwareimplementierung wurde in Echtzeit auf einem AMD Athlon-Rechner mit 1 GHz unter Linux getestet. Die Messungen mit dem integrierten Krypto-Prozessor wurden ebenfalls auf einem Standard PC (Intel Pentium III mit 550 MHz) unter MS Windows NT 4.0 durchgeführt. Die Java Quellen wurden mit JDK 1.3 übersetzt, der C/C++ Code mit MS Visual C++ 6.0.

Es ist deutlich zu erkennen, dass die reine Softwareimplementierung auf einem herkömmlichen Rechner den Anforderungen von Heimanwendern vollkommen genügt. Doch wie schon erläutert, ist für Serveranwendungen eine auf die Schlüssellänge n angepasste Hardwareimplementierung zwingend erforderlich. Mit dem FlexiECProvider und dem darin integrierten KryptoProzessor ist es möglich, 175 Signaturen/Sek zu verifizieren.

Damit haben wir einen Provider für Elliptische-Kurven-Kryptogra-

phie, der auch den höchsten Anforderungen bzgl. Performanz und Sicherheit genügt. Darüber hinaus steht durch die Portabilität von Java die Funktionalität des Providers auch für den normalen Benutzer zur Verfügung.

Danksagung

Die Autoren danken der DFG für die Förderung der Arbeit im Rahmen des Projekts „Sicherheit in der Informations- und Kommunikationstechnik“.

Literatur

- [1] <http://www.informatik.tu-darmstadt.de/TI/Forschung/Welcome.html>
- [2] N. Koblitz, Elliptic Curve Cryptosystems, Mathematics of Computation, Vol. 48, 1987.
- [3] V. Miller, Use of Elliptic Curves in Cryptography, Advances in Cryptology, Proc. CRYPTO'85, LNCS 218, H.C. Williams, Ed., Springer-Verlag, 1986.
- [4] M. Ernst, S. Klupsch, O. Hauck and S. Huss, Rapid Prototyping for Hardware Accelerated Elliptic Curve Public Key Cryptosystems, Proc. 12th IEEE Workshop on Rapid System Prototyping (RSP01), Monterey, CA, June 2001.
- [5] J. Massey and J. Omura, Computational Method and Apparatus for Finite Field Arithmetic, U.S. Patent 4.587.627, 1986.

Informationen zum Fachgebiet Kryptographie und Computeralgebra an der TU Darmstadt

Die Lehre von der Verschlüsselung, die Kryptographie und deren Anwendung in der Praxis ist der Forschungsschwerpunkt im Fachgebiet „Kryptographie und Computeralgebra“ unter der Leitung von Professor Dr. Johannes Buchmann. Die Palette der wissenschaftlichen Arbeiten, die an unserem Institut entstehen, reicht von der Entwicklung der Trustcentersoftware FlexiTrust, die den Aufbau von komplexen Public-Key-Infrastrukturen erlaubt, über die Erforschung und Implementierung neuer kryptographischer Verfahren in Soft- und Hardware bis zur Untersuchung der mathematischen Probleme, auf denen die Sicherheit der kryptographischen Algorithmen beruht. So wird die Zahlkörper-Kryptographie und die Gitter-Kryptographie vorangetrieben und im FlexiProvider implementiert. Die im FlexiProvider implementierte Elliptische-Kurven-Kryptographie zieht mit den entstehenden Standards immer gleich und ist in Technik als auch in Performance führend.

Ansprechpartner:

Prof. Dr. Johannes Buchmann
Fachgebiet Kryptographie und
Computeralgebra

Tel.: 0 61 51 / 16-48 89

E-mail:

info@cdc.informatik.tu-darmstadt.de

Informationen zum Fachgebiet Integrierte Schaltungen und Systeme an der TU Darmstadt

Die Forschung am Fachgebiet Integrierte Schaltungen und Systeme (ISS) ist im Bereich der Technischen Informatik angesiedelt und ist der besonderen Rolle gewidmet, die integrierte Schaltungen in komplexen Systemen der Informationsverarbeitung einnehmen. Das Fachgebiet gehört dem Fachbereich Informatik der TU Darmstadt an und wird von Prof. Huss geleitet, der vor 10 Jahren aus der Industrie an die TU Darmstadt berufen wurde. Über die Zweitmitgliedschaft seines Leiters im Fachbereich Elektrotechnik und Informatikstechnik bildet ISS eine Brücke zwischen der Informatik und der Elektrotechnik an dieser Universität.

Die Forschungsarbeiten am ISS lassen sich in drei Kompetenzbereiche einteilen. Der Bereich „Hardware-/Software Koentwurf“ befasst sich mit Fragestellungen einer ganzheitlichen Entwurfsmethodik für eingebettete Systeme. Der Kompetenzbereich „Hochleistungsarchitekturen“ hat die Entwicklung anwendungsspezifischer Koprozessoren zum Ziel, die das in eingebetteten Systemen häufig geforderte Zeitverhalten erst ermöglichen. Der dritte Kompetenzbereich „Modellierung heterogener Systemkomponenten“ erforscht Problemstellungen der Modellbildung und der ausführbaren Modellrepräsentation in standardisierten Hardware-Beschreibungssprachen wie VHDL, VHDL-AMS und SystemC.

Das Fachgebiet verfügt über eine umfangreiche Entwurfs-, Validierungs- und Testausstattung sowohl für integrierte Schaltungen als auch für eingebettete Systeme. Vielfältige Kooperationen mit Forschungseinrichtungen in Europa, USA und China und gemeinsame Projekte mit Unternehmen aus dem In- und Ausland werden für eine praxisrelevante Ausrichtung der Forschungsarbeiten genutzt.

Ansprechpartner:

Prof. Dr.-Ing. Sorin Alexander Huss
Alexanderstrasse 10
D-64283 Darmstadt

Tel.: 06151/16-3980
Fax: 06151/16-4810

E-mail: huss@iss.tu-darmstadt.de
ISS Homepage:
<http://www.vlsi.informatik.tu-darmstadt.de>

NanoPhotonics AG ist ein expandierendes Unternehmen mit weltweitem Erfolg bei der Entwicklung und Fertigung miniaturisierter optischer Messgeräte für die Oberflächenanalytik. Unsere Kunden sind führende Halbleiter- und Chip-Produzenten. Wir suchen eine(n)

Software-Ingenieur(in)

Aufgaben:

- Weiterentwicklung und Pflege der Produktsoftware
- Entwicklung und Programmierung von Testständen zur Qualitätskontrolle
- Projektkoordinierung mit externen Partnern

Qualifikation:

- Abgeschlossenes Studium als Diplom-Ingenieur der Fachrichtung Informatik, Physik, Elektrotechnik oder verwandten Gebieten
- Kenntnisse in objektorientierter Programmierung unter MS Visual C++
- Vorteilhaft sind Kenntnisse in UML und Datenbankprogrammierung (Access, ADO)
- Kenntnisse in Linux, XML und HTML sind wünschenswert
- Mögliche Ausbildungsschwerpunkte in Physik, Bildverarbeitung, Mikroprozessortechnologie, Optik, Messtechnik, Antriebstechnik
- Gute Englischkenntnisse

Wir bieten ein angenehmes Arbeitsklima in einem jungen Team von Ingenieuren und Physikern, modernste Ausstattung, anspruchsvolle und abwechslungsreiche Aufgaben und Freiräume für die Entfaltung Ihrer Kreativität. Nutzen Sie die einmalige Chance, ein zukunftsorientiertes Unternehmen mit anspruchsvollen Technologien mitzugestalten!

NanoPhotonics AG, Karin Prüfert, Galileo-Galilei-Str. 28, 55129 Mainz, Telefon: (06131) 95 85 410, Pruefert@nanophotonics.de, www.nanophotonics.de



NANOPHOTONICS