

Viele E-Commerce Anwendungen sind u.a. dadurch gekennzeichnet, daß vertrauliche Informationen über öffentliche Kommunikationsnetze (z.B. das Internet) ausgetauscht werden. Diese Informationen müssen vor dem unberechtigten Zugriff Dritter geschützt werden. Die Basistechnologie, um diesen Schutz zu gewährleisten, ist die Public-Key Kryptographie. Neben dem weit verbreiteten RSA-Verfahren gewinnen Public-Key-Verfahren auf der Basis Elliptischer Kurven (EC) zunehmend an Bedeutung.

Die Sicherheit dieser Verfahren hängt im wesentlichen von der verwendeten Schlüssellänge ab (1024 RSA-Bits entsprechen ca. 160 EC-Bits). Im Vergleich zu RSA bedeutet dies, daß beim Einsatz von EC-Verfahren das gleiche Sicherheitsniveau bereits mit wesentlich kürzeren Schlüsseln erreicht werden kann. Die geringere Schlüssellänge ermöglicht wiederum effizientere Implementierungen und somit höheren Durchsatz.

Die Operation $K \cdot P$, d.h. die Multiplikation eines Punktes $P(x,y)$ auf der Kurve mit einer natürlichen Zahl K , stellt die Basisoperation im Bereich der EC-Kryptographie dar. Sie ist sehr aufwendig und entsprechend rechenzeitintensiv. Die zur Berechnung von $K \cdot P$ benötigte Zeit bestimmt im wesentlichen das Laufzeitverhalten von EC-Implementierungen. Der am Institut für Integrierte Schaltungen und Systeme (ISS) der TU Darmstadt entwickelte *Elliptic-Curve* Kryptoprozessor führt genau diese Operation in Hardware durch. Die Performanz von EC-Kryptosystemen (z.B. *Online-Banking* Server) kann durch den Einsatz des Prozessors um mehrere Größenordnungen gesteigert werden.

Das Verfahren

Das Diskrete Logarithmus Problem (DL-Problem) über der Punktgruppe Elliptischer Kurven (alle Punkte $P(x,y)$, die die Gleichung $y^2+xy=x^3+ax^2+b$ erfüllen) bildet die Grundlage für die Sicherheit von *Elliptic-Curve* Kryptoverfahren. Am Beispiel einer natürlichen Zahl K und eines Punktes $P_1(x,y)$ auf der Kurve beschreibt das DL-Problem im Grunde folgenden Sachverhalt:

1. Es ist **relativ leicht**, den Punkt $P_2 = K \cdot P_1$ zu berechnen.
2. Es ist **nahezu unmöglich**, den Schlüssel K allein aus P_1 und P_2 zu berechnen.

Aufgrund dieser Eigenschaften ist es möglich, effiziente und sichere Kryptoverfahren auf der Basis Elliptischer Kurven zu implementieren.

Die Operation $K \cdot P$ hat allerdings nichts mit der *normalen* Multiplikation ganzer Zahlen zu tun, sondern sie ist das K -malige Aufaddieren von P und kann mit dem *Double-and-Add* Algorithmus (siehe Abbildung 1) berechnet werden. Dieser Algorithmus verwendet die Operationen Punktverdopplung (*EC-Double*) und Punktaddition (*EC-Add*) der darunterliegenden *Elliptic-Curve* Arithmetik.

Das Laufzeitverhalten des Algorithmus ist abhängig von der Schlüssellänge N und dem Hamminggewicht H , d.h. der Anzahl der 1-Bits in der Binärdarstellung von K . Zur Berechnung von $K \cdot P$ müssen N *EC-Double* und H *EC-Add* Operationen durchgeführt werden. Diese EC-Operationen basieren wiederum auf einer N -Bit breiten *Finite-Field* Arithmetik mit den Operationen *FF-Mult*, *FF-Add*

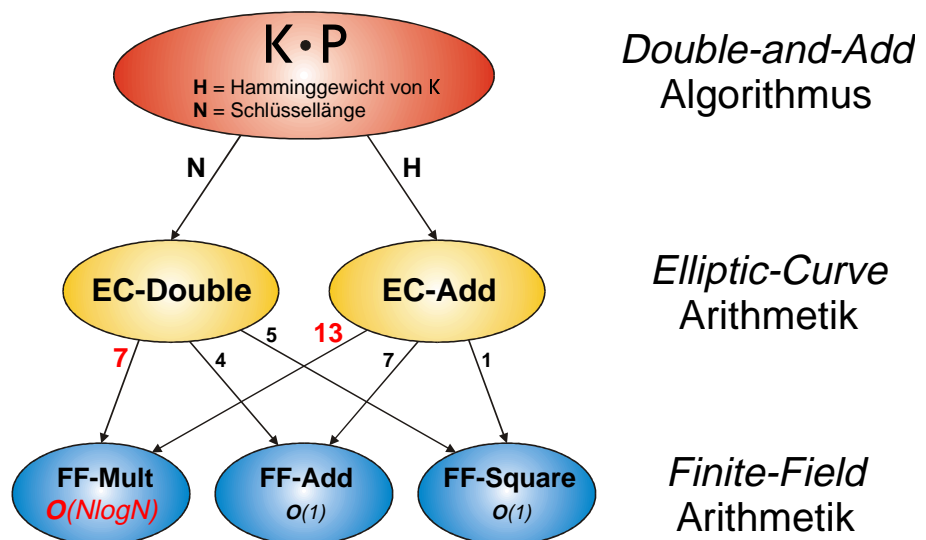


Abbildung 1: *Double-and-Add* Algorithmus

und *FF-Square*, wobei *FF-Mult* die aufwendigste¹ und auch meist verwendete Operation² ist. Letztlich ist die Ausführungsgeschwindigkeit der Operation *FF-Mult* der ausschlaggebende Faktor für die zur Berechnung von $K \cdot P$ benötigte Rechenzeit. Die *Finite-Field* Multiplikation besteht wiederum aus N *Finite-Field* Elementaroperationen, wobei mit jeder Elementaroperation ein Bit des Ergebnisvektors erzeugt wird. Für eine Berechnung von $K \cdot P$ müssen insgesamt $N(7N+13H)$ *Finite-Field* Elementaroperationen durchgeführt werden.

Die Umsetzung des Verfahrens ist sowohl in Software als auch in Hardware möglich. Software-Implementierungen haben allerdings den Nachteil, daß die N -Bit breiten *Finite-Field* Elementaroperationen 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 Implementierung des Verfahrens ist ein auf die Schlüssellänge N angepaßter Kryptoprozessor zwingend erforderlich. Durch die, bei einer Hardwarerealisierung mögliche, parallele Ausführung mehrerer *Finite-Field* Elementaroperationen kann die Performanz des Verfahrens nochmals erheblich gesteigert werden.

Beispiel: Zur Berechnung von $P_2 = K \cdot P_1$ mit $N=270$ und $H=141$ werden **3.723** *FF-Mult* Operationen und somit **1.005.210** *Finite-Field* Elementaroperationen benötigt.

$$\begin{aligned}
 K &= 2c9347124cad8faab85e3d8f7c15585b4dde4c853a6d597909f4fa70b36e565ab0f2 \\
 P_1(x,y) &= (3cba001300ed3cb212e35fd700cc52f84e7970ee3b697a093bfc35891e9d7ac30dc1, \\
 &\quad 1b8d059e7b438d8dfccef5bfbcb0f4a866fd591bc5c5e7af20afc1beee1451e35eb60) \\
 P_2 = K \cdot P_1 &= (12c606897bbfaab38eaea84c2b2a1bcb31d4065cc9646f9584c5c070bbbde1b45bdf, \\
 &\quad 219644f15009229e299fc7227fa9cae1f5cdb63fe92015b7bc87baca20905a04a91a)
 \end{aligned}$$

Die Hardwareplattform

Die Implementierung des *Elliptic-Curve* Kryptoprozessors 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 Kryptoprozessors 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 unter Verwendung von speziellen Konfigurations- und Kommunikationsfunktionen. *microEnable* gibt es in mehreren Varianten: mit unterschiedlicher SRAM-Bestückung und mit FPGA-Bausteinen unterschiedlicher Komplexität.

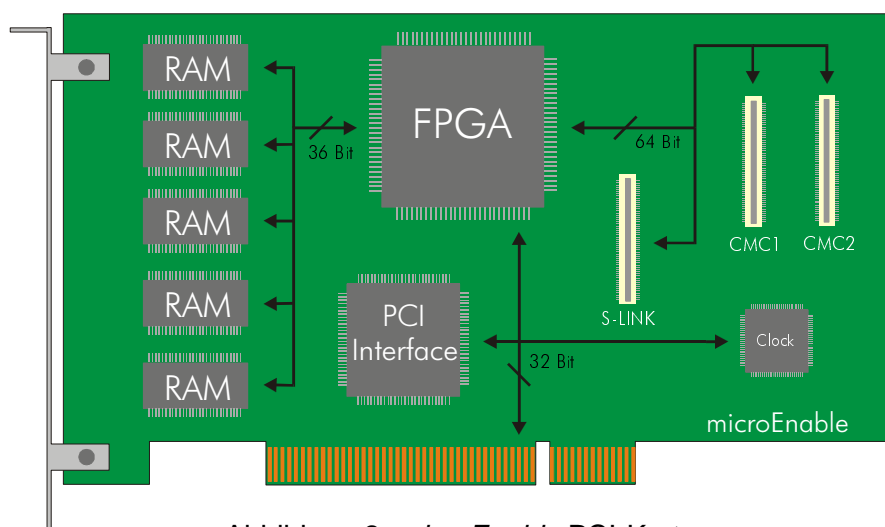


Abbildung 2: *microEnable* PCI-Karte

¹ Der Berechnungsaufwand für *FF-Mult* ist $O(N \log N)$, also abhängig von der verwendeten Schlüssellänge. Der Aufwand zur Berechnung von *FF-Add* und *FF-Square* ist dagegen $O(1)$, also konstant.

² Für eine *EC-Double* Operation sind 7 *FF-Mult* notwendig, *EC-Add* benötigt 13 *FF-Mult* Operationen.

Die Prozessorarchitektur

Auf dem am ISS entwickelten *Elliptic-Curve* Kryptoprozessor ist der *komplette Double-and-Add* Algorithmus zur Berechnung von $K \cdot P$ in Hardware implementiert. Die Architektur des Prozessors ist in Abbildung 3 dargestellt und besteht im wesentlichen aus drei funktionalen Blöcken.

Finite-Field Arithmetik:

Hier sind die oben beschriebenen *FF*-Operationen implementiert. Für *FF-Add* und *FF-Square* steht jeweils eine Komponente zur Verfügung. Zur Berechnung von *FF-Mult* können mehrere Komponenten instanziiert werden. Hierdurch lassen sich die jeweils verfügbaren FPGA-Ressourcen optimal ausnutzen, was wiederum die Voraussetzung zum Erreichen der maximalen Performanz ist.

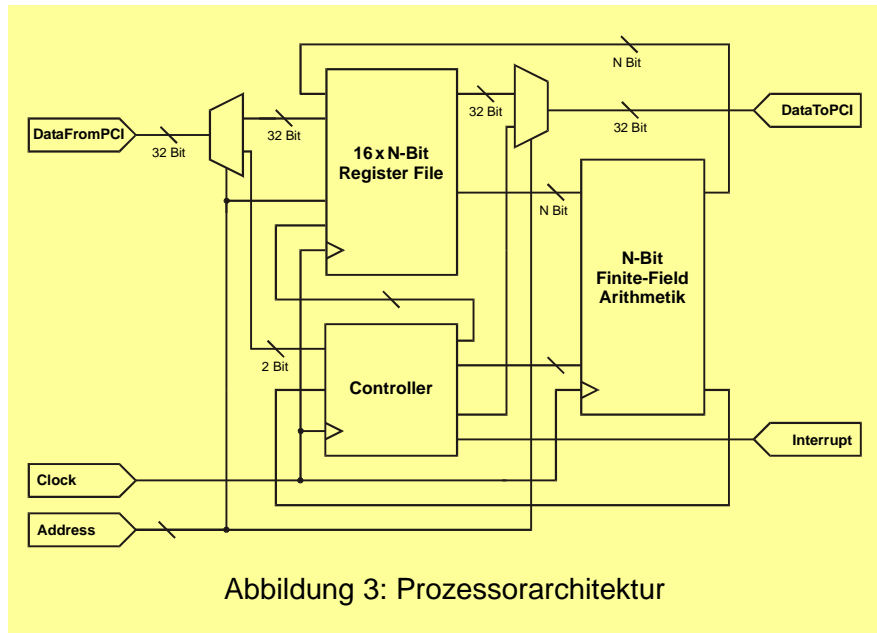


Abbildung 3: Prozessorarchitektur

Register File:

Der Prozessor verfügt über 16 *N*-Bit Register zur Speicherung von *FF*-Körperelementen bzw. von positiven ganzen Zahlen der entsprechenden Bitbreite.

Controller: Im Controller ist der *Double-and-Add* Algorithmus zur Berechnung von $K \cdot P$ implementiert. Er ist als Endlicher Automat realisiert und stellt das Steuerwerk des Kryptoprozessors dar.

Die in Abbildung 3 angegebenen externen Signale dienen zur Kommunikation mit dem Kryptoprozessor und werden von der PCI-Schnittstelle zur Verfügung gestellt.

Der Entwurfsablauf

Für den Entwurf und die Implementierung der Kryptoprozessoren wird ein VHDL-gestützter Entwurfsablauf eingesetzt. VHDL ist der *de-facto* Standard für die abstrakte Modellierung digitaler Schaltungen. Wie bereits oben erläutert, ist die Architektur des Kryptoprozessors variabel in Bezug auf die Schlüssellänge und den Parallelisierungsgrad, d.h. der Ergebnis-Bits, die parallel berechnet werden. Für eine konkrete Implementierung müssen diese beiden Parameter allerdings festgelegt werden. Zur Erzeugung einer solchen spezifischen Variante des Kryptoprozessors wurde am Institut ein spezielles VHDL-Generatorprogramm entwickelt. Das automatisch generierte VHDL-Modell des Prozessors bildet die Grundlage für die anschließende Hardware-Synthese und die weiteren Entwurfsschritte zur Erzeugung der FPGA-Konfigurationsdatei.

Kooperationspartner



Silicon Software GmbH
Seckenheimer Straße 12
D-68165 Mannheim
www.silicon-software.de

Kryptographie und Computeralgebra

Prof. Johannes Buchmann
Fachbereich Informatik
TU Darmstadt
www.cdc.informatik.tu-darmstadt.de