

Hocheffiziente modulare Multiplikation für $\mathbb{GF}(P)$

Rainer Blümel*, Ralf Laue† und Sorin A. Huss†

* cv cryptovision GmbH † Integrierte Schaltungen und Systeme
Gelsenkirchen TU Darmstadt

Die Leistungsfähigkeit heutiger Public-Key-Systeme beruht hauptsächlich auf der Effizienz der modularen Körper-Multiplikation. Wir stellen einen Algorithmus für $\mathbb{GF}(P)$ mit nur $n^2 + 7n$ Wort-Multiplikationen vor.

Die grundlegende Idee ist von Karatsuba abgeleitet: Je zwei Wort-Multiplikationen werden zu einer Wort-Multiplikation zusammengefasst, allerdings ohne Einsatz von Rekursion. Die beschleunigte Multiplikationsformel ermöglicht es, eine Mehr-Wort-Multiplikation mit nur $\frac{n \cdot (n+1)}{2}$ Wort-Multiplikationen zu berechnen.

Die Reduktion lässt sich auch als Multiplikation darstellen: $0 \leq X \odot Y \ominus P \odot Z < P$. So kann die beschleunigte Multiplikation auch für die Reduktionsphase eingesetzt werden. Sortiert man die Terme in Reihenfolge absteigender Wertigkeit, ergibt sich $X \odot Y \ominus P \odot Z =$

$$\sum_{i=n-1}^0 \left\{ \sum_{j=0}^{i-1} ((x_i + x_j) \cdot (y_i + y_j) + p_i \cdot z_i - x_i \cdot y_i) \cdot 2^{(i+j)b} \right. \\ \left. - (p_i \cdot z_i - x_i \cdot y_i) 2^{2ib} \right. \\ \left. + \sum_{j=i+1}^{n-1} (p_i \cdot z_i - x_i \cdot y_i - (p_i + p_j) \cdot (z_i + z_j)) \cdot 2^{(i+j)b} \right\}.$$

In jedem Schritt wird nur je ein neues z_i benötigt, welches zu Beginn jedes Schrittes mit einem *Look Ahead*-Mechanismus abgeschätzt wird: Die höchstwertigen Terme des i -ten Schrittes – mit Ausnahme der z_i enthaltenden Teilterme – werden ausgewertet. Aus dem Ergebnis lässt sich z_i mit einer Division durch P errechnen. Als Ersatz für die Division wird eine Multiplikation mit dem Kehrwert der führenden Wörter von P verwendet (ähnlich zu Barrett).

Ungenauere z_i werden durch Korrekturberechnungen ausgeglichen. Geschickte Parameterwahl ermöglicht es, die Genauigkeit soweit zu erhöhen, dass der Aufwand zur Korrektur ignoriert werden kann (Fehlerwahrscheinlichkeit $\approx 10^{-5}$). Für jeden Schritt werden 2 Wort-Multiplikationen für $(p_i \cdot z_i - x_i \cdot y_i)$ und weitere $n - 1$ für die Summen benötigt. Unsere Implementierung benötigt weiterhin 3 Wort-Multiplikationen für die Auswertung der höchstwertigen Terme und 3 für die Multiplikation mit dem Kehrwert. Als Summe ergibt sich die oben erwähnte Komplexität von $n^2 + 7n$.