A highly efficient modular Multiplication Algorithm for Finite Field Arithmetic in $\mathbb{GF}(P)$

Rainer Blümel² Ralf Laue¹ Sorin A. Huss¹

¹Integrated Circuits and Systems Laboratory, Department of Computer Science Technische Universität Darmstadt, Germany {laue|huss}@iss.tu-darmstadt.de

²cv cryptovision GmbH, Gelsenkirchen, Germany rainer.bluemel@cryptovision.com

CRyptographic Advances in Secure Hardware, 2005



- Introduction
- Novel Algorithm
 - Basic Modular Multiplication Algorithm
 - Look Ahead Procedure
 - Example
- Implementation
 - Implementation Details
 - Experimental Results



Motivation

- Most common public-key algorithm depend heavily on modular multiplication.
- As the multiplication is most critical digit-operation, it is taken as complexity metric here.
- Novel modular multiplication algorithm for $\mathbb{GF}(P)$ has a complexity of $n^2 + 7n$.
- This is superior to Montgomery $(2n^2 + 2n)$, although the algorithm flow is more complex.



Overview

- Basic idea of Karatsuba is used to get an improved multi-precision multiplication algorithm.
- This algorithm is used to speed up both multiplication and reduction phase of a modular multiplication.
- The quotient/multiplier of P is estimated with a technique similar to the Barrett reduction.
- Hardware-implementation was used to verify feasibility of the novel multiplication algorithm.



- 1 Introduction
- Novel Algorithm
 - Basic Modular Multiplication Algorithm
 - Look Ahead Procedure
 - Example
- Implementation
 - Implementation Details
 - Experimental Results





Application of basic Karatsuba idea

Karatsuba-Algorithm: Recursive application of

$$(x_1 \cdot 2^b + x_0) \cdot (y_1 \cdot 2^b + y_0) = x_1 \cdot y_1 \cdot 2^{2b} + x_0 \cdot y_0 + ((x_1 + x_0) \cdot (y_1 + y_0) - x_1 \cdot y_1 - x_0 \cdot y_0) \cdot 2^b$$

- Our *Improved Multiplication* algorithm uses this idea: Two digit multiplications are combined to one of the form $(x_i + x_i) \cdot (y_i + y_i)$ without using recursion.
 - Leads to higher computational effort than Karatsuba.
 - + Provides better flexibility and less overhead than Karatsuba.
 - + Retains part of its computational advantage compared to standard multiplication techniques.



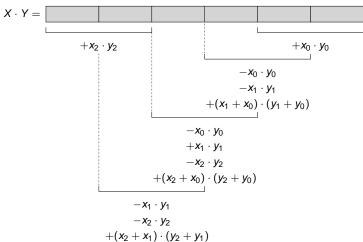
Complexity of Improved Multiplication

	School-	accelerated	recursive
n	algorithm	multiplication	Karatsuba
2	4	3	3
3	9	6	_
4	16	10	9
8	64	36	27
16	256	136	81

- Complexity is $\frac{n(n+1)}{2}$.
- Multiplier with operand width of (digit-length + 1) is needed.



Example for Improved Multiplication



Idea for the Modular Multiplication

- The result of the modular Multiplication between X and Y with prime number *P* has to fulfill $0 < X \odot Y \ominus P \odot Z < P$.
- Assuming Z is known, it is easy to implement $X \odot Y \ominus P \odot Z$ with the improved multiplication, thus accelerating both multiplication and reduction phase.
- The terms are sorted in the order of digits:
 - By interleaving both phases, the number of memory accesses is reduced (similar to Montgomery)
 - Opens a way to successively calculate Z (see next slide).



Basic Modular Multiplication Algorithm

$$egin{array}{lll} egin{array}{lll} egin{array} egin{array}{lll} egin{array}{lll} egin{array}{lll} egin{array}{ll$$

- In every step one new digit of Z is needed.
- It is estimated with the so called Look Ahead procedure.



- Introduction
- Novel Algorithm
 - Basic Modular Multiplication Algorithm
 - Look Ahead Procedure
 - Example
- Implementation
 - Implementation Details
 - Experimental Results





Pre-evaluation of Next Step

- Look Ahead pre-evaluates terms of highest digits of next step, omitting terms containing z_i .
- In most simple case:

$$A + (r_i - (p_i + p_{n-1}) \cdot (z_i + z_{n-1})) \cdot 2^{(i+n-1)b}$$
.

In other words:

$$\tilde{A} = A - (x_i \cdot y_i + p_{n-1} \cdot z_{n-1} + p_i \cdot z_{n-1}) \cdot 2^{(i+n-1)b}$$

- The highest digits of \tilde{A} now only contain the product of the highest digits of P and z_i .
- Because P is known, it is possible to estimate z_i .



Estimation of z_i

- The division by P is substituted by a multiplication with the reciprocal of the highest digits of P named q_{rec} (similar to Barrett).
- Basically two ways to handle inaccurate z_i:
 - z_i may exceed digit-length: Inaccuracies can be compensated during the estimation of z_{i-1} .
 - z_i may not exceed digit-length: Inaccuracies have to be addressed by a correction computation. But if the accuracy is high enough, corrections will be needed seldom enough, so they can be neglected for complexity considerations.



- Introduction
- Novel Algorithm
 - Basic Modular Multiplication Algorithm
 - Look Ahead Procedure
 - Example
- Implementation
 - Implementation Details
 - Experimental Results





Modular Multiplication Algorithm

```
1: A = 0
```

2: **for**
$$i = n - 1$$
 to $i = 0$ **do**

3: Look Ahead: estimate
$$z_i$$

4: **if**
$$z_i$$
 exceeds one digit **then**

5:
$$A = A \pm P \cdot 2^{(i+1)b}$$

6: end if

7: Main Step: Update A

8: end for

9: **if** A > P or A < 0 **then**

10: $A = A \pm P$

11: end if



Look Ahead

Equation for Look Ahead (considering two highest digits)

$$\begin{split} \tilde{A} &= A \quad - \quad (x_i \cdot y_i + p_{n-1} \cdot z_{n-1} + p_i \cdot z_{n-1}) \cdot 2^{(i+n-1)b} \\ &- \quad (x_i \cdot y_i + p_{n-2} \cdot z_{n-2} + p_i \cdot z_{n-2}) \cdot 2^{(i+n-2)b} \\ &- \quad (x_{i-1} \cdot y_{i-1} + p_{n-1} \cdot z_{n-1} + p_{i-1} \cdot z_{n-1}) \cdot 2^{(i+n-2)b}. \end{split}$$

- Involved are
 - The two highest terms of this step of the main loop.
 - The highest term of next step of the main loop.
- Only valid for general case (consult paper for details).



Estimation of z_i

• Taking two digits into account (c = 2b):

$$q_{rec} = \frac{2^{4b}}{p_{n-1} \cdot 2^b + p_{n-2}}$$

• The z_i can then calculated by:

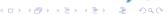
$$z_i = q_{rec} \cdot (\tilde{a}_{i+n+1} \cdot 2^{2b} + \tilde{a}_{i+n} \cdot 2^b + \tilde{a}_{i+n-1}) \div 2^{3b}$$

• ÷ is implemented as bit shift operation.



- Introduction
- Novel Algorithm
 - Basic Modular Multiplication Algorithm
 - Look Ahead Procedure
 - Example
- Implementation
 - Implementation Details
 - Experimental Results





System

- For test purpose an ECC implementation was done on a Virtex II Pro FPGA.
 - For portability reasons only CLBs were used, e.g., ignoring dedicated multipliers.
- ECAdd and ECDouble were taken from IEEE 1363.
- EC multiplication implemented by modified Lim/Lee.
- We used the variant without over-long z_i , because it allows for simpler digit operations.



Accuracy considerations

- Parameters should be chosen to minimize number of operations for Look Ahead and correction.
- This concerns mainly the number of used digits for the Look Ahead and the length of q_{rec}.
- In this case (as in example):
 - Look Ahead takes into account two highest digits.
 - Reciprocal has length of two digits.
 - Probability for correction is $\approx 2.5 \cdot 10^{-5}$ (i.e. one in 4000).
- Further increasing accuracy would have increased necessary effort unproportional.



- 1 Introduction
- Novel Algorithm
 - Basic Modular Multiplication Algorithm
 - Look Ahead Procedure
 - Example
- Implementation
 - Implementation Details
 - Experimental Results





Results

Parameter	Proposed algorithm		Montgomery	
set	ECInit	<i>ECMult</i>	ECInit	ECMult
secp112r1	15.911 <i>m</i> s	14.182 <i>ms</i>	17.376 <i>ms</i>	15.755 <i>m</i> s
secp128r1	21.702 <i>ms</i>	21.151 <i>m</i> s	24.777 <i>ms</i>	24.474 <i>m</i> s
secp160k1	37.829 <i>m</i> s	37.134 <i>m</i> s	46.263 <i>m</i> s	45.931 <i>m</i> s
secp192k1	59.392 <i>m</i> s	59.456 <i>m</i> s	76.498 <i>m</i> s	77.063 <i>m</i> s
secp224k1	88.920 <i>ms</i>	89.330 <i>ms</i>	119.211 <i>m</i> s	120.792 <i>m</i> s
secp256k1	126.696 <i>ms</i>	126.834 <i>ms</i>	175.476 <i>ms</i>	178.522 <i>m</i> s

- Mean values over 20 EC multiplications each.
- Modular multiplication with 100MHz, rest at 33MHz.



Summary

- Modular Multiplication Algorithm with superior complexity.
- Because of overhead: This is only valid, if multiplication is most expensive operation.
- Focussed on Hardware, because needed operation with operands of (digit length + 1) is expensive in Software.

- Future Work
 - Is the variant with over-long z_i feasible?
 - Adaption of Barrett: looks promising.



The End

Any Questions?

