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

Rainer Blümel[2] Ralf Laue[1] Sorin A. Huss[1]

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

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

CRyptographic Advances in Secure Hardware, 2005

Outline

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.

Outline

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 $(x_i + x_j) \cdot (y_i + y_j)$ 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

n	School-algorithm	accelerated multiplication	recursive 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

$X \cdot Y =$

$+x_2 \cdot y_2$

$+x_0 \cdot y_0$

$-x_0 \cdot y_0$
$-x_1 \cdot y_1$
$+(x_1 + x_0) \cdot (y_1 + y_0)$

$-x_0 \cdot y_0$
$+x_1 \cdot y_1$
$-x_2 \cdot y_2$
$+(x_2 + x_0) \cdot (y_2 + y_0)$

$-x_1 \cdot y_1$
$-x_2 \cdot y_2$
$+(x_2 + x_1) \cdot (y_2 + y_1)$

Idea for the Modular Multiplication

- The result of the modular Multiplication between X and Y with prime number P has to fulfill $0 \leq X \odot Y \ominus P \odot Z < P$.
- 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

$$
\begin{aligned}
X \odot Y \ominus P \odot Z = \\
\sum_{i=n-1}^{0} \Big\{ & \sum_{j=0}^{i-1} \big((x_i + x_j)\cdot(y_i + y_j) + p_i \cdot z_i - x_i \cdot y_i\big) \cdot 2^{(i+j)b} \\
& - (p_i \cdot z_i - x_i \cdot y_i) 2^{2ib} \\
& + \sum_{j=i+1}^{n-1} \big(p_i \cdot z_i - x_i \cdot y_i - (p_i + p_j)\cdot(z_i + z_j)\big) \cdot 2^{(i+j)b} \\
\Big\} &
\end{aligned}
$$

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

Outline

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.

Outline

Modular Multiplication Algorithm

```
A = 0
for i = n − 1 to i = 0 do
    Look Ahead: estimate z_i
    if z_i exceeds one digit then
        A = A ± P · 2^((i+1)b)
    end if
    Main Step: Update A
end for
if A > P or A < 0 then
    A = A ± P
end if
```

Look Ahead

- Equation for Look Ahead (considering two highest digits)

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

- 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:

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

  - $\div$ is implemented as bit shift operation.

Outline

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.

Outline

1. Introduction

2. Novel Algorithm
   - Basic Modular Multiplication Algorithm
   - Look Ahead - Procedure
   - Example

3. Implementation
   - Implementation Details
   - Experimental Results

Results

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

- 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?