

Efficient Embedded Software Implementations of Public-Key Cryptography

Çetin Kaya Koç

koc@cryptocode.net

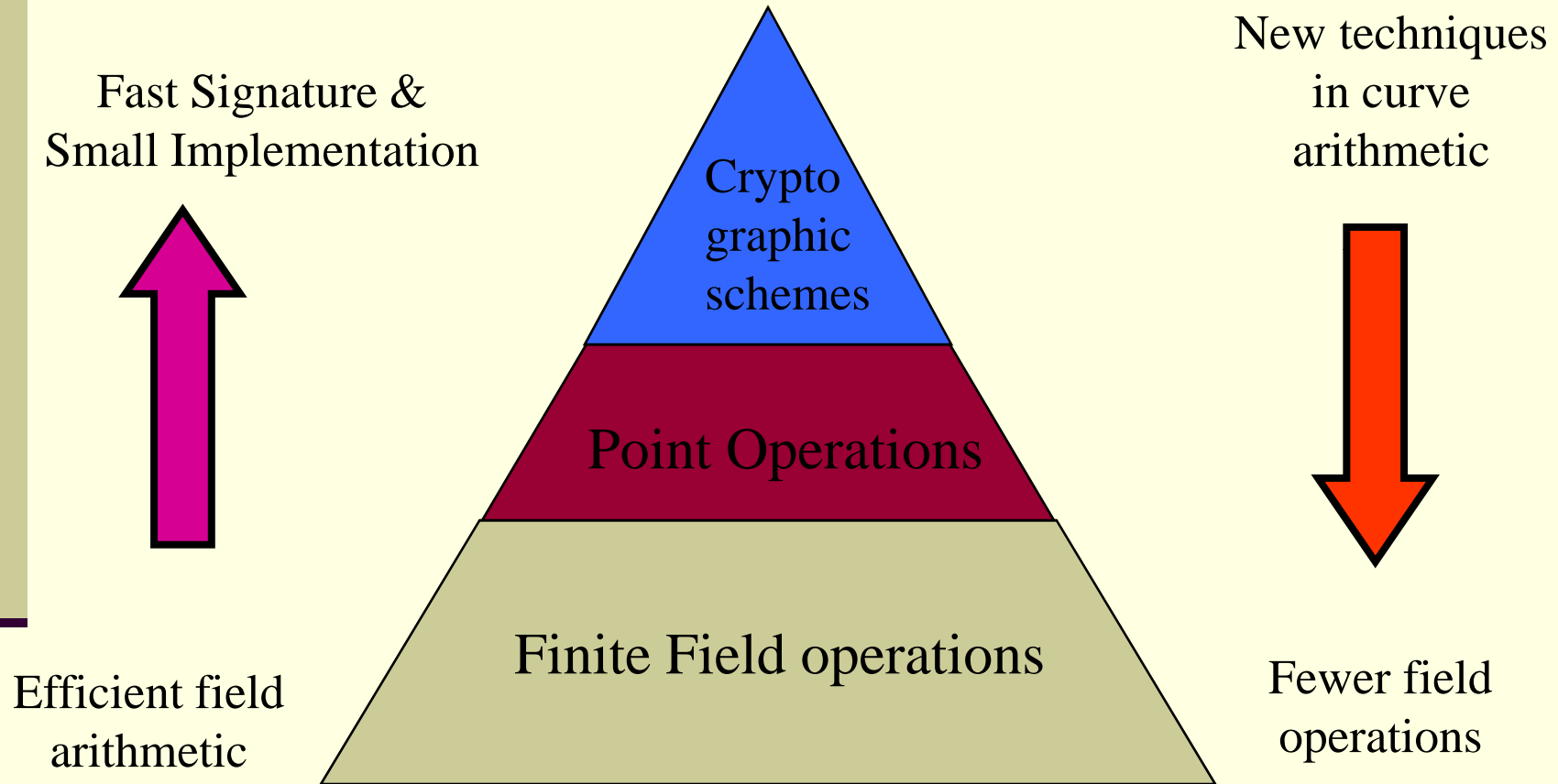
Characteristics of Embedded Systems

- Low Computing Power
- Low Dynamic Memory for Data
- Restricted Stack Memory Usage
- Diverse Memory Allocation Techniques
- Different Memory Management & Organization
- Limitations on Program Code Size
- Difficulty in handling exceptions on System Calls

Challenges of Public Key Cryptography in Embedded Systems

- High Latency of Public Key Operations
- Excessive Demand on Dynamic Memory
- High Stack Memory Requirement
- High footprint
- Exception handling
- Secure Execution
- Removal of Sensitive Information from Memory

Elliptic Curve Cryptography



Finite Fields for ECC Usage

- Prime fields, $GF(p)$
- Binary extension fields, $GF(2^k)$.
- General Extension fields $GF(p^k)$ - Not as common
- **The Focus:** $GF(p)$ & $GF(2^k)$
- **The Goal:** Fast execution of finite field operations in embedded environment requiring low memory and footprint

Finite Field Operations in ECC

- Addition in $GF(p)$ and $GF(2^k)$
Inexpensive in terms of time and area
- Multiplicative inversion in $GF(p)$ and $GF(2^k)$
Prohibitively expensive in terms of time
Possible to avoid some of them
- Multiplication in $GF(p)$ and $GF(2^k)$
Expensive in terms of time and area
Most important operation
Focus of many design efforts

Incomplete Arithmetic in GF(p)

- An effective method to enhance the time performance of arithmetic operations in software particularly when the field degree is not on the word boundary of underlying processor
- For example, $2^{176} > p \geq 2^{175}$ when the word size of the computer is 32 bits
- Based on a technique eliminating the need for reduction when the intermediary result does not exceed the word boundary.
- Up to 13% overall speedup

Addition & Subtraction in $GF(p)$

- Takes up 15% plus of total computation time in an EC point multiplication operation with projective coordinates
- Implemented in Assembly which improves both operations at least 100%
- Incomplete arithmetic provides up to 40% and 25% performance improvement for addition and subtraction, respectively

Multiplication in $GF(p)$

- Takes up 70% plus of total computation time in an EC point multiplication operation with projective coordinates
- Assembly implementation improves 150%
- Incomplete arithmetic provides up to 5%
- CIOS Montgomery algorithm seems to be the best performer when the prime p is arbitrarily chosen

Multiplicative Inversion in $GF(p)$

- Very important operation when affine EC coordinates are used
- The Montgomery inversion algorithm is used
- A new correction phase, which provides 1.5 overall speedup, is utilized
- No assembly is used

Arithmetic Operations in $GF(2^k)$

- No need to use Assembly language since there is no carry handling
- The subtraction operation is the same as the addition
- Addition is a trivial operation which does not need any special treatment in software
- All operations can be optimized for irreducible trinomials or pentanomials of any type

Multiplication in $GF(2^k)$

- Takes up about 70% of total computation time in an EC point multiplication operation with projective coordinates
- Montgomery's algorithm is not used for trinomials or pentanomials
- Consists of two phases:
 - Polynomial Multiplication
 - Reduction with the irreducible polynomial

Multiplication Phase

- Non-recursive Karatsuba-Offman algorithm is quite effective since redundant operations are eliminated
- More improvement is possible with an increase in the code size and with acceleration tables
- Standard Karatsuba-Offman algorithm provides around 25% overall speedup

Reduction Phase

- Standard reduction method is the best when trinomials or pentanomials are used
- $GF(2^k)$ version of Montgomery's method performs better when randomly chosen irreducible polynomials are used
- Acceleration tables of different amounts can provide different levels of speedup

Elliptic Curve Point Operations

- Consists of a number of finite field operations
- For speed, mixed modified Jacobian coordinates offer the maximum performance
- For processors of low resources, affine coordinates provide a lean version of ECC
- Different acceleration techniques provide different levels of speedup at the expense of memory
- Simultaneous point multiplication with redundant representation improves signature verification

Special Curve Solutions

- All operations (field & point arithmetic) can be highly optimized if fixed special curves are chosen
- For $GF(p)$, the speedup is up to 100% over the standard implementation
- For $GF(2^k)$, the speedup is about 50%
- For $GF(2^k)$, the speedup can increase as high as 400% if Koblitz curves are used

Desired Characteristics of the Embedded Software (1)

- Reentrant

A computer program or routine is described as **reentrant** if it is designed such that a single copy of the program's instructions in memory can be shared by multiple users or separate processes. The key to the design of a reentrant program is to ensure that no portion of the program code is modified by the different users/processes, and that process-unique information (such as local variables) is kept in a separate area of memory that is distinct for each user or process

- Thread-safe

A piece of code is **thread-safe** if it is reentrant or protected from multiple simultaneous execution by some form of mutual exclusion

Desired Characteristics of the Embedded Software (2)

- Modularity and OS independence
- No OS or C calls within the library
- Low stack usage (under 1K is typical)
- Streamlined dynamic memory usage
- All memory allocation should (can) be made at one point by user
- Allocation can be made automatic with either user-supplied or standard memory manager

Desired Characteristics of the Embedded Software (3)

- Memory requirement for any curve can be reported by a function before any cryptographic operation is performed
- Supports any curve of infinite precision with highly optimized manner
- Configurable RNG
- Safe execution
- Removes any sensitive information from the memory after the cryptographic computations

Desired Characteristics of the Embedded Software (4)

- A set of acceleration techniques provide different levels of speedup
- Dynamic memory is divided into two parts: curve-specific or temporary. This feature allows several possibilities:
 - Curve-specific memory can be shared between different threads using the same curve
 - Curve-specific memory can be allocated and initialized offline

Performance on 80-MHz ARM7TDMI

Bitsize	Signature (ms)			Memory (KB)		
	PC0	PC1	PC2	PC0	PC1	PC2
192 – GF(p)	70	28	13	2	12	69
256 – GF(p)	145	53	24	3	20	85
163 – GF(2^k)	105	40	23	2	10	59
257 – GF(2^k)	274	109	45	3	22	96

- Random curves are used. Further improvement is possible with fixed curves
- PC0: No precomputation
- PC1: Precomputation Level 1, etc.

Performance on 20-MHz PALM

Bitsize	Signature (s)			Memory (KB)		
	PC0	PC1	PC2	PC0	PC1	PC2
192 – GF(p)	5.1	2.1	1.4	2	12	26
256 – GF(p)	11.1	4.3	3.1	3	20	44
163 – GF(2^k)	4.9	2.0	1.5	2	10	22
257 – GF(2^k)	12.0	4.6	3.5	3	22	50

- Random curves are used. Further improvement is possible with fixed curves
- PC0: No precomputation
- PC1: Precomputation Level 1, etc.
- Precomputation is restricted due to limited memory

References

- C. K. Koc, T. Acar, and B. S. Kaliski Jr. Analyzing and comparing Montgomery multiplication algorithms. *IEEE Micro*, 16(3):26-33, June 1996.
- C. K. Koc and T. Acar. Montgomery multiplication in $GF(2^k)$. *Designs, Codes and Cryptography*, 14(1):57-69, April 1998.
- E. Savas and C. K. Koc. The Montgomery modular inverse - revisited. *IEEE Transactions on Computers*, 49(7):763-766, July 2000.
- M. Aydos, T. Yanik, and C. K. Koc. High-speed implementation of an ECC-based wireless authentication protocol on an ARM microprocessor. *IEE Proceedings - Communications*, 148(5):273-279, October 2001.
- T. Yanik, E. Savas, and C. K. Koc. Incomplete reduction in modular arithmetic. *IEE Proceedings - Computers and Digital Techniques*, 149(2):46-52, March 2002.
- T. Wollinger, J. Pelzl, V. Wittelsberger, C. Paar, G. Saldamli, and C. K. Koc. Elliptic and hyperelliptic curves on embedded μP . *ACM Transactions on Embedded Computing Systems*, 3(3):509-533, August 2004.