# Secure Binary Field Multiplication\*

Hwajeong Seo<sup>1</sup>, Chien-Ning Chen<sup>2</sup>, Zhe Liu<sup>2</sup>, Yasuyuki Nogami<sup>3</sup>, Taehwan Park<sup>1</sup>, Jongseok Choi<sup>1</sup>, and Howon Kim<sup>1\*\*</sup>

<sup>1</sup> Pusan National University, School of Computer Science and Engineering, San-30, Jangjeon-Dong, Geumjeong-Gu, Busan 609-735, Republic of Korea {hwajeong,pth5804,jschoi85,howonkim}@pusan.ac.kr <sup>2</sup> Nanyang Technological University, Physical Analysis & Cryptographic Engineering (PACE), {chienning}@ntu.edu.sg <sup>3</sup> University of Luxembourg, Laboratory of Algorithmics, Cryptology and Security (LACS), 6, rue R. Coudenhove-Kalergi, L-1359 Luxembourg-Kirchberg, Luxembourg {zhe.liu}@uni.lu <sup>4</sup> Okayama University, Graduate School of Natural Science and Technology, 3-1-1, Tsushima-naka, Kita, Okayama, 700-8530, Japan {yasuyuki.nogami}@okayama-u.ac.jp

Abstract. Binary field multiplication is the most fundamental building block of binary field Elliptic Curve Cryptography (ECC) and Galois/Counter Mode (GCM). Both bit-wise scanning and Look-Up Table (LUT) based methods are commonly used for binary field multiplication. In terms of Side Channel Attack (SCA), bit-wise scanning exploits insecure branch operations which leaks information in a form of timing and power consumption. On the other hands, LUT based method is regarded as a relatively secure approach because LUT access can be conducted in a regular and atomic form. This ensures a constant time solution as well. In this paper, we conduct the SCA on the LUT based binary field multiplication. The attack exploits the horizontal Correlation Power Analysis (CPA) on weights of LUT. We identify the operand with only a power trace of binary field multiplication. In order to prevent SCA, we also suggest a mask based binary field multiplication which ensures a regular and constant time solution without LUT and branch statements.

**Keywords:** Binary Field Multiplication, Embedded Processors, Side Channel Attack, Horizontal Correlation Power Analysis

<sup>\*</sup> This work was partly supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government (MSIP) (No.10043907, Development of high performance IoT device and Open Platform with Intelligent Software) and the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the ITRC(Information Technology Research Center) support program (IITP-2015-H8501-15-1017) supervised by the IITP(Institute for Information & communications Technology Promotion).

<sup>\*\*</sup> Corresponding Author

#### 1 Introduction

Elliptic Curve Cryptography (ECC) scalar multiplication (point addition and doubling) consists of field arithmetic including addition/subtraction, multiplication/squaring and inversion [9]. Of these operations, multiplication/squaring is the most performance-critical operations of point addition and doubling. Any efforts spent in optimizing these operations are deserved. The binary field multiplication is normally established with a combination of exclusive-or and bitshift operations. The advanced binary field implementations achieved remarkably good performance by reducing the number of partial products and replacing bit operations into consecutive look-up table accesses known as Lopez et al.'s method [12, 14, 17]. The alternative approach, Block-Comb based multiplication, is introduced in [18]. Several operands are grouped in block-wise and multiple operands are computed at once. Later, an unbalanced fashion of Block-Comb method is presented by Seo et al. in [15]. This method increases the size of block by exploiting the additional registers with instruction set level optimization techniques and then computes the multiplication in block-wise way as like former Block-Comb approaches. There is nice way to reduce the multiplication overheads into sub-quadratic complexity known as Karatsuba algorithm. This method efficiently replaces multiplication into several addition and subtraction operations. There exist several papers which have considered the Karatsuba's technique for speeding-up the performance of binary field multiplication over embedded processors. Lopez et al. in [12] conducts Karatsuba multiplication with look-up table accesses. This trial reduces the number of multiplication but it increases overheads for constructing the look-up table which is beyond benefits of Karatsuba approach. Oliveira et al. in [14] also mentioned that Lopez et al.'s normal method is faster than Karatsuba's multiplication by a factor of 44%. Recently, an alternative combination of Karatsuba algorithm and Block-Comb method is introduced [16]. The method exploits 3-term Karatsuba and reduces 3 63-bit partial products out of 9. The results show high performance gains in 163-bit Koblitz curve. There is relatively few number of papers considering Side Channel Attack (SCA) on these binary field multiplication. While the fastest implementations over embedded processors were a quite important factor in the past, because implementations of binary field ECC over embedded processor were too slow due to limited computing power and storages. However without secure implementations against side channel attack, practical applications are limited [3, 2, 11]. In this paper, we explore all binary field multiplication on embedded processors in terms of side channel attack and show the vulnerability of LUT based binary field multiplication as well. Lastly, we introduce a secure mask based binary field multiplication.

#### Summary of Research Contributions

The main contributions of our work are summarized as follows.

1. Side channel attacks on LUT based binary field multiplication. We present side channel attacks on consecutive memory access patterns. We used hor-

izontal Correlation Power Analysis (CPA) on weights of LUT. The method successfully extracts the correlation co-efficient from power traces of binary field multiplication and identifies the operands used in the operations.

2. Develop the secure binary field multiplication techniques. Unlike previous binary field multiplication methods, we designed a new mask based binary field multiplication. Since this regular and atomic method is branch-free and LUT-free, proposed method is much secure against simple power analysis than previous approaches. In order to boost the performance, we exploit several levels of Karatsuba multiplication.

The remainder of this paper is organized as follows. In Section 2, we overview the previous binary field multiplication methods. In Section 3, we point out the vulnerability of previous approaches in terms of side channel attack and show our side channel attack results on LUT based binary field multiplication. In Section 4, we present a mask based binary field multiplication. Finally, Section 5 concludes the paper.

# 2 Binary Field Multiplications

The look-up table based binary field multiplication, introduced in [12, 14], replaces the bit operations into look-up table access operations. In order to utilize the method, we should compute the look-up table by one operand and place them into memory as pre-computed results. Typically, the range is chosen to 4, indicating a 4-bit value  $(0x0\sim 0xf)$  for 16  $(2^4)$  cases. The look-up table occupies memory size at least  $16 \times m$ , where m is size of operands. After generating the look-up table with target operand, we access the look-up table by 4-bit wise and then update intermediate results with the pre-computed results from LUT, thereby reducing complex shift or bit-wise exclusive-or operations. The alternative approach is Block-Comb method. The method executes consecutive bit-wise exclusive-or on intermediate results under condition of bit setting by block-wise [18]. In every session, one bit of block is tested from the least significant to the most significant bits. If the bit is set to 1, the intermediate results are updated by block-wise operand. After then intermediate result is left-shifted by 1-bit to align the location of result. This process is iterated by size of word. Since, whole processes are conducted in block-wise fashion, the intermediate result and operand bit test are handled efficiently. The unbalanced Block-Comb method optimizes the utilization of general purpose registers [15] to retain more operands into registers. A key feature of the method is the computation of partial products using extra bits in operands. After bit test of the operand, the least significant bit of the operand is not used anymore. We can store the carry bit of intermediate result into the least significant bit of operand. The advantage of exploiting additional registers is that we can compute partial products in large block size, which reduces the number of block-wise partial products. Karatsuba method is a general technique to reduce the complexity of multiplication with small number of addition operations. In [19], Karatsuba method is applied to Lopez et

al.'s method but it increases overheads for constructing the look-up table in online which is beyond benefits of Karatsuba approach. Oliveira et al. in [14] also mentioned that original Lopez et al.'s method is faster than Karatsuba's multiplication by a factor of 44%. Since Karatsuba method divides long integers into half and conducts multiplication operations on them, as many as we divide multiplication blocks, the pre-computation costs significantly increase. Block-Comb method is grouping several bytes of the operand into a block and then computes the multiplication in block-wise fashion. A Karatsuba Block-Comb (KBC) firstly groups the operand and then Karatsuba multiplication is conducted in blockwise fashion. However, the Block-Comb approach is vulnerable toward timing attacks. The computation time is highly relied on bit setting. In order to remove the timing information, constant time solution was introduced. In the method, each bit is evaluated and conduct operations in a regular timing. If the bit is set, it conduct bit-wise exclusive-or with operand. Otherwise it conducts bit-wise exclusive-or with zero register. Even though this method eliminates the timing information, it still leaks power consumption from branch operations.

# 3 Side Channel Attacks on Binary Field Multiplications

Real world multiplication can be conducted with dedicated 8-, 16-, or 32-bit multipliers of Arithmetic Logic Unit (ALU). In terms of binary field multiplication, high-end processors such as ARM and Intel chip-sets support polynomial multiplier. However, low-end 8-bit and 16-bit embedded processors do not support polynomial multiplications yet. In order to get acceptable speed performance over low-end devices, previous binary field multiplication operations exploit the basic logical operations or LUT computations. However, the works mainly focus on speed optimizations rather than secure implementations which expose information leakages. In this paper, we conduct the side channel attacks on binary field multiplication over the popular embedded board named ARDUINO UNO with the 8-bit 16MHz AVR microcontroller Atmega328p. An AVR processor, such as the Atmel ATmega series, features 32 general-purpose registers, of which six are used for pointers. In particular, the register pair (R26,R27) is aliased as X pointer, the register pair (R28,R29) is aliased as Y pointer, and the register pair (R30,R31) is aliased as Z pointer. The AVR microcontrollers have separate memories and buses for program and data. It has a total of 133 instructions and each instruction has a fixed latency. Ordinary arithmetic/logical instructions (e.g. add) are executed in a single clock cycle, while a mul instruction takes two clock cycles, and also load/store instructions take two cycles. Most of the software implementation on AVR processors is written in both mixed C and Assembly code. The function-call specifies that the first three 16-bit arguments (e.g., pointers) are passed in register pairs (R24, R25), (R22,R23), and (R21,R20). It furthermore specifies that registers R2-R17, R28, and R29 are "called- saved" registers, and the register R1 is assumed by the compiler to always contain zero thus has to be set to zero before returning from a function. In order to measure the power consumption of AVR processor, we manipulate the circuit to get the chip's

Constant	time with NOP	Constant time w	with ZERO Register
SBRS B0,0	NOPIN:	SBRS B0,0	ZEROIN:
RJMP NOPIN	NOP	RJMP ZEROIN	EOR CO, ZERO
EOR CO, AO	NOP	EOR CO, AO	EOR C1, ZERO
EOR C1, A1	NOP	EOR C1, A1	EOR C2, ZERO
EOR C2, A2	NOP	EOR C2, A2	EOR C3, ZERO
EOR C3, A3	NOP	EOR C3, A3	EOR C4, ZERO
EOR C4, A4	NOP	EOR C4, A4	EOR C5, ZERO
EOR C5, A5	NOP	EOR C5, A5	EOR C6, ZERO
EOR C6, A6	NOP	EOR C6, A6	NOP
RJMP NOPOUT	NOPOUT:	RJMP ZEROOUT	ZEROOUT:

Table 1. AVR program codes for constant time binary field multiplication

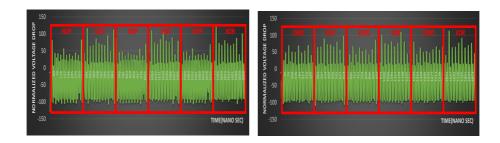


Fig. 1. Power traces for (left) NOP operations and (right) ZERO register based constant time binary field multiplication

power consumption with less noise. The power consumption of the AVR microcontroller is measured by LeCroy WaveRunner 610Zi oscilloscope with AP033 active differential probe at the sampling rate 10G/s.

Firstly, Block-Comb variants including original Block-Comb, unbalanced Block-Comb and Karatsuba Block-Comb need to check the bit setting of the operands. If the bit is set, the operands are added to intermediate results. If the bit is not set, the whole operations are skipped. This irregular form of program leaks timing information because it has different timing pattern between branched or non-branched cases. In order to hide timing information, we can ensure constant time solutions by inserting the NOP or adding ZERO operations for branched cases. In Table 3, the AVR program codes for constant time binary field multiplication are described. It consumes same clock cycles with padding operations. However both approaches leak another important information namely power consumption. In Figure 1, the power traces of both methods are drawn. For NOP based operations, it vividly shows low power consumption than exclusive-or operations. For alternative approach based on ZERO register, it relatively consumes more power than NOP based approaches but it still shows different power consumption patterns compared with normal exclusive-or operation.

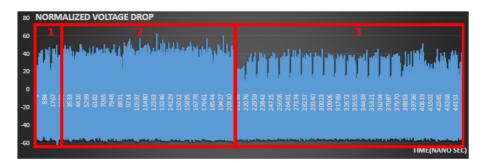
Algorithm 1 Lopez et al. multiplication in  $\mathbb{F}_{2^m}$  [12]

```
Require: A = A[0, ..., n-1], B = B[0, ..., n-1] where word size is 8-bit.
Ensure: C = C[0, ..., 2n - 1].
1: Compute T = U \cdot B for all polynomials U of degree lower than t = 4-bit.
2: C[0, ..., 2n - 1] \leftarrow 0
3: for k from 0 by 1 to n-1 do
4:
      u \leftarrow A[k] \gg t
      for j from 0 by 1 to n-1 do
5:
         C[j+k] \leftarrow C[j+k] \oplus T(u)[j]
6:
      end for
7:
8: end for
9: C \leftarrow C \cdot 2^t
10: for k from 0 by 1 to n-1 do
11:
       u \leftarrow A[k] \mod 2^t
12:
       for j from 0 by 1 to n-1 do
13:
         C[j+k] \leftarrow C[j+k] \oplus T(u)[j]
14:
       end for
15: end for
16: return C
```

On the other hands, Lopez et al.'s look-up table approach is relatively secure against simple power analysis than that of Block-Comb variants, because the method does not have branch statements and only consists of regular memory access operations. This nice property does not leak the irregular form of timing and power consumption. However, Messerges et al. showed that the power consumption of a smart card depends on the activity on both data and address bus [13]. Chen showed the method to identify the memory address from power consumption of consecutive memory accesses and distinguish the multiplication and squaring operations from the multiplication always based exponentiation algorithm [4]. The horizontal correlation analysis computes the correlation factor on segments corresponding to the atomic operations and extracts secret information [5]. This attack successfully finds the secret keys of RSA crypto systems from power traces.

In this paper, we attack the LUT based binary field multiplication, which has high relations between memory access pattern and weights of operands because LUT is written in online by referring the operand values and it is read by referring the another operand values. In Algorithm 1, Lopez's binary field multiplication consists of online LUT construction based on the operand (B)and consecutive LUT accesses by using another operand (A) as an offset. In Step 1, LUT is generated by multiplying the operands (B) and degree (t). The degree determines the size of LUT  $(2^t)^5$ . From Step 4, higher 4-bit of operands (A) are extracted by degree (t) and then the higher value (u) is used as index

<sup>&</sup>lt;sup>5</sup> Normally the degree (t) is set to 4 to get 16 cases because degree 8 needs too large LUT ( $2^8 = 256$ ). Since the other degrees including (5, 6, 7) are not the power of two, they are inefficient over 8, 16 or 32-bit machine.



**Fig. 2.** Power traces of Lopez et al. method (①, ②) LUT construction, (③) 4-bit wise binary field multiplication with LUT

of LUT (T). In Step 6, the LUT variables are added to intermediate results (C). Same procedures are conducted again for lower 4-bit of (A). In Figure 4, the power traces of Lopez et al. with degree 4 are described. The Section (1) is constructing the LUT with variable (0, 1, 2, 4, 8) and Section (2) is constructing the LUT with remaining variables (3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15). The Section (3) is conducting the 4-bit wise binary field multiplication with LUT. Since, the LUT is based on operands (A), the power consumption is highly relied on weights of operands. In Table 2, the source codes for look-up table construction is described. The LUT construction conducts consecutive memory store. The ST instruction, transferring the data from register to memory storages, takes 2 clock cycles. This instruction consumes certain pattern of power consumption depending on weights of data, because bit one or zero generates different power consumption. If we extract current data patterns from power consumption, we can identify the operands used for building LUT. With the leakage information, we can identify the scalar addition and doubling from scalar multiplication where the information is the secret key of ECDH operations.

In order to show the practical attack results, we target the smallest field namely sect113r1. If we succeed in attacks on the smallest field, we can readily extend to larger binary fields such as 128, 163, 193, 233, 239, 283, 409 and 571-bit. The 113-bit binary field multiplication over 8-bit processor needs  $14 \left( \left\lceil \frac{113}{8} \right\rceil \right)$  bytes to store one element of LUT. For side channel attack, we collected power traces of look-up table construction cases including 0x1, 0x3, 0x5, 0x7, 0x09, 0xb, 0xd, 0xf from (1) and (2) in Figure 4. Each case consists of 14 memory store operations. In one round of multiplication power analysis with their weights. We compute the correlation factor on segments corresponding to the atomic LUT access operations and identify the specific operands. AVR processor uses data bus when it stores byte-wise data from registers to memory. This causes certain power consumption patterns depending on the weights of passing data. The idea is to calculate the Pearson Correlation Coefficient  $\rho$  between power consumption and the Hamming Weight of the LUT namely HW<sub>LUT</sub>(A<sub>i</sub>). If it accesses the LUT

8

Look up table construction for $0x01$ and $0x02$ cases						
ST Z+, AO	ST Z+, A9	ROL A3	ROL A12	ST Z+, A6		
ST Z+, A1	ST Z+, A10	ROL A4	ROL A13	ST Z+, A7		
ST Z+, A2	ST Z+, A11	ROL A5	ROL A14	ST Z+, A8		
ST Z+, A3	ST Z+, A12	ROL A6	ST Z+, AO	ST Z+, A9		
ST Z+, A4	ST Z+, A13	ROL A7	ST Z+, A1	ST Z+, A10		
ST Z+, A5	ST Z+, A14	ROL A8	ST Z+, A2	ST Z+, A11		
ST Z+, A6	LSL AO	ROL A9	ST Z+, A3	ST Z+, A12		
ST Z+, A7	ROL A1	ROL A10	ST Z+, A4	ST Z+, A13		
ST Z+, A8	ROL A2	ROL A11	ST Z+, A5	ST Z+, A14		

Table 2. AVR program codes for LUT computations of Lopez et al.

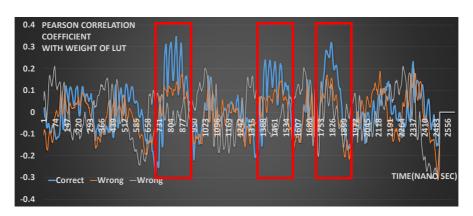


Fig. 3. Correlation value of weights and power consumptions

by using *i*-th operand (A) at the time t, there will be a higher coefficient  $\rho$ . In the correlation graph of Figure 3, we found distinguished three peaks colored in blue. The other lines show wrong estimations with wrong LUT and weights.

## 4 Secure Binary Field Multiplication

In Section 3, we explored the vulnerability of existing binary field multiplication methods. The main vulnerabilities of existing approaches are branch statements and predicable memory access patterns based on the weights of memory. In order to reduce the information leakages, we introduce a Masked Block-Comb method. The method avoids the branch operation and even LUT accesses but it uses an operand masking to conduct the regular and atomic form of binary field multiplication.

In Algorithm 2, 32-bit wise MBC multiplication is introduced. In Step 3, *i*-th bit of operand A[j] is stored in *BIT* variable. In Step 4, zero value is subtracted from *BIT* and borrow bit is stored into *T*1 variable. In Step 5, zero value is subtracted from *T*1. This outputs 0 if *T*1 is zero and if not it outputs

Algorithm 2 Masked Block Comb on 32-bit **Require:** Two 32-bit operands A and B**Ensure:**  $C(64\text{-bit}) = A \cdot B$ 1: for *i* from 7 by 1 to 0 do for j from 3 by 1 to 0 do 2: 3:  $BIT = A[j]\&(1 \ll i)$ 4:  $\{T1, T0\} = 0 - BIT$ 5:MASK = 0 - T16: for k from 3 by 1 to 0 do 7:  $C[k+j] = C[k+j] \oplus (B[k]\&MASK)$ 8: end for 9: end for  $C = C \ll 1$ 10:11: end for 12: return C

Table 3. AVR program codes for masked block comb multiplication

Constant time with masked operand					
LDI BIT,	0880	SUB ZERO, BIT	AND MO, BIT	EOR CO, MO	
AND BIT,	AO	SBC BIT, BIT	AND M1, BIT	EOR C1, M1	
CLR ZERO		MOVW MO, BO	AND M2, BIT	EOR C2, M2	
		MOVW M2, B2	AND M3, BIT	EOR C3, M3	

**Oxff.** After then Step 7, operand B[k] is masked with variable MASK and then conduct Block-Comb style multiplication. This operation is conducted by the number of block size (4, 32-bit). After then intermediate results C is left shifted by 1-bit and this process is iterated by 7 times more. The detailed source code is available in Table 3. We set the bit with LDI operation and then conduct AND with operand A0. If the bit is set, the (SBC BIT, BIT) operation outputs Oxff. After then this masking bit are used for AND operation with operand B. If the BIT is Oxff, masked operand M is operand B. If not masked operand M is set to zero variable. The computed masked operand M is finally bit-wise exclusive-ored with intermediate results C. Since MBC method exploits many masking process, it shows low performance results than previous approaches. In order to boost the performance, we adopted the asymptotically faster multiplication namely Karatsuba multiplication.

In Algorithm 3, 64-bit wise Karatsuba Masked Block-Comb method is introduced. We selected 64-bit for practical usages because size of binary field ECC is multiple of 64-bit and modern processors including INTEL and ARM provide 64-bit polynomial multiplication. This means our method can readily adopt the other 64-bit optimization techniques to the embedded processors as well [7, 8, 6]. In Step 1, 32-bit wise MSK multiplication is conducted on  $A[3 \sim 0]$  and  $B[3 \sim 0]$  and outputs lower part of intermediate results ( $L[7 \sim 0]$ ). In Step 2, higher part ( $H[7 \sim 0]$ ) are computed by multiplying  $A[7 \sim 4]$  and  $B[7 \sim 4]$ . In Step 3, high and low part of operands A and B are bit-wise exclusive-ored

 Algorithm 3 64-bit Karatsuba Block Comb

 Require: An eight 8-bit operand A(64-bit) and B(64-bit) 

 Ensure:  $C(128-bit) = A \cdot B$  

 1:  $L[7 \sim 0] = (A[3 \sim 0] \times_{32-bit} B[3 \sim 0])$  

 2:  $H[7 \sim 0] = (A[7 \sim 4] \times_{32-bit} B[7 \sim 4])$  

 3:  $M[7 \sim 0] = ((A[7 \sim 4] \oplus A[3 \sim 0]) \times_{32-bit} (B[7 \sim 4] \oplus B[3 \sim 0]))$  

 4:  $M[7 \sim 0] = M[7 \sim 0] \oplus L[7 \sim 0] \oplus H[7 \sim 0]$  

 5:  $C = H[7 \sim 0] \cdot 2^{64} \oplus M[7 \sim 0] \cdot 2^{32} \oplus L[7 \sim 0]$  

 6: return C 

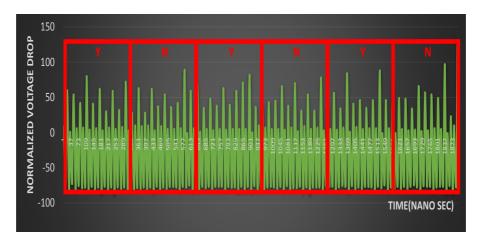


Fig. 4. Power traces of proposed method

and then the operands are multiplied to output middle part ( $M[7 \sim 0]$ ). In Step 4, higher and lower parts are bit-wise exclusive-ored with middle part. It Step 5, all intermediate results are bit-wise exclusive-ored. Total 64-bit Karatsuba Masked Block-Comb needs 1926 clock cycles. In order to construct binary field multiplication for sect163k1 and sect233k1 with 64-bit API, we need 192 and 256-bit multiplications. For 163-bit multiplication we adopt three terms Karatsuba multiplication. The method reduces the three multiplication out of nine multiplication. Firstly we construct 128-bit binary field multiplication with 1 level Karatsuba with 64-bit 1 level Karatsuba and then this 128-bit multiplication is used for 1 level of 256-bit multiplication. Totally three levels of Karatsuba multiplication is used.

As like previous Block-Comb variants, we also collected the power traces of proposed methods. The detailed descriptions are available in Figure 4. The red blocks represent one round of Table 3. The operation consists of register based operations and there is no consecutive memory accesses. For this reason, the power traces do not leak information through timing and power consumption that is common vulnerabilities of Block-Comb methods. Since MBC method

Method	Technique	Vulnerability	Clock Cycle	
163-bit binary field m				
Kargl et al. [10]	4-bit wise Lookup Table	LUT access	5,057	
Aranha et al. [1]	4-bit wise Lookup Table	LUT access	4,508	
Seo et al. [15]	Unbalanced Block Comb	Branch op.	4,346	
Seo et al. [16]	Karatsuba Block Comb	Branch op.	$3,\!274$	
Seo et al. [16]	Constant Karatsuba Block Comb	Branch op.	5,005	
Proposed Method	-	$14,\!445$		
233-bit binary field multiplication				
Aranha et al. [1]	4-bit wise Lookup Table	LUT access	8,314	
Proposed Method	-	20,537		

Table 4. Clock cycles for 163- and 233-bit multiplication on binary fields.

does not access memory in consecutive way, our SCA on LUT does not work properly. In Figure 4, the symbols including Y and N represent 0xff mask and 0x00 mask, respectively. In Table 4, performance of binary field multiplication is described. LUT based approach exploits 4-bit wise lookup table accesses but it leaks memory access pattern as we pointed out in this paper. In terms of Block-Comb method, it shows the highest performance. However it is not constant time solution and branch operation leaks power consumption information. The constant Karatusba Block-Comb method is even an atomic solution but it is still based on branch operation which leaks power consumption information. Our proposed method namely masked Karatsuba Block-Comb method shows the lowest speed performance but it does not include vulnerabilities found in existing approaches. Furthermore, our method is easily scalable with Karatsuba approaches. The overheads from 163 to 233 is only 1.4 but previous works are 1.8. For this reason, our method would be better choice for large operands.

## 5 Conclusion

In this paper, we conduct side channel attacks on LUT based binary field multiplication by collecting the power traces from LUT accesses. We also presented the novel binary field multiplication namely masked Karatsuba Block-Comb. This method exploits masking method to get regular form of results which is also easily scalable for large operands. Our future works are attacking the other algorithms based on LUT such as window methods for exponentiation and scalar multiplication. It is also worth to note that binary field squaring is also using constant look-up table and our attack would be available in this case as well. Furthermore, for higher performance we will try to optimize the current implementation techniques for practical applications. Lastly but not least we will evaluate performance of GCM and other ECC primitives with proposed secure binary field multiplication.

#### References

- D. F. Aranha, R. Dahab, J. López, and L. B. Oliveira. Efficient implementation of elliptic curve cryptography in wireless sensors. *Adv. in Math. of Comm.*, 4(2):169– 187, 2010.
- E. Brier, C. Clavier, and F. Olivier. Correlation power analysis with a leakage model. In *Cryptographic Hardware and Embedded Systems-CHES 2004*, pages 16– 29. Springer, 2004.
- S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi. Towards sound approaches to counteract power-analysis attacks. In *Advances in CryptologyCRYPTO99*, pages 398–412. Springer, 1999.
- C.-N. Chen. Memory address side-channel analysis on exponentiation. In *ICISC* 2014. 2014.
- C. Clavier, B. Feix, G. Gagnerot, M. Roussellet, and V. Verneuil. Horizontal correlation analysis on exponentiation. In *Information and Communications Security*, pages 46–61. Springer, 2010.
- C. P. Gouvêa and J. López. Implementing gcm on armv8. In *Topics in Cryptology* CT-RSA 2015, pages 167–180. Springer, 2015.
- 7. S. Gueron. Aes-gcm software performance on the current high end cpus as a performance baseline for caesar competition.
- 8. S. Gueron and M. E. Kounavis. Intel® carry-less multiplication instruction and its usage for computing the gcm mode. *Intel white paper (September 2012)*, 2010.
- D. Hankerson, S. Vanstone, and A. J. Menezes. Guide to elliptic curve cryptography. Springer, 2004.
- 10. A. Kargl, S. Pyka, and H. Seuschek. Fast arithmetic on atmega128 for elliptic curve cryptography. *IACR Cryptology ePrint Archive*, 2008:442, 2008.
- P. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In Advances in CryptologyCRYPTO99, pages 388–397. Springer, 1999.
- J. López and R. Dahab. High-speed software multiplication in f2m. In Progress in CryptologyINDOCRYPT 2000, pages 203–212. Springer, 2000.
- T. S. Messerges, E. A. Dabbish, and R. H. Sloan. Investigations of power analysis attacks on smartcards. In USENIX workshop on Smartcard Technology, volume 17, page 17, 1999.
- L. B. Oliveira, D. F. Aranha, C. P. Gouvêa, M. Scott, D. F. Câmara, J. López, and R. Dahab. Tinypbc: Pairings for authenticated identity-based non-interactive key distribution in sensor networks. *Computer Communications*, 34(3):485–493, 2011.
- H. Seo, Y. Lee, H. Kim, T. Park, and H. Kim. Binary and prime field multiplication for public key cryptography on embedded microprocessors. *Security and Communication Networks*, 7(4):774–787, 2014.
- H. Seo, Z. Liu, J. Choi, and H. Kim. Karatsuba–block-comb technique for elliptic curve cryptography over binary fields. *Security and Communication Networks*, 2015.
- S. C. Seo, D.-G. HAN, H. C. Kim, and S. HONG. Tinyecck: Efficient elliptic curve cryptography implementation over gf(2m) on 8-bit micaz mote. *IEICE transactions* on information and systems, 91(5):1338–1347, 2008.
- M. Shirase, Y. Miyazaki, T. Takagi, and D.-G. HAN. Efficient implementation of pairing-based cryptography on a sensor node. *IEICE transactions on information* and systems, 92(5):909–917, 2009.
- P. Szczechowiak, L. B. Oliveira, M. Scott, M. Collier, and R. Dahab. Nanoecc: Testing the limits of elliptic curve cryptography in sensor networks. In *Wireless* sensor networks, pages 305–320. Springer, 2008.