ENGINEERING IN ADVANCED RESEARCH SCIENCE AND TECHNOLOGY

ISSN 2278-2566
Vol.02, Issue.03
August -2018
Pages: 637-641

AN EFFICIENT AND ENHANCED MODULAR MULTIPLIER FOR ARBITRARY MERSENNE PRIMES ON FPGA

1. E.NARAYAANA SWAMY, 2.Dr. N. A. V. PRASAD
1. M.Tech Student, Dept. of ECE, S.V Institute of Technology, Anantapuram, A.P
2. Professor & H.O.D, Dept. of ECE, S.V Institute of Technology, Anantapuram, A.P

ABSTRACT:
This paper presets s most known and recent methods for efficient modular multiplication. .This concept explained that the modular multiplication $A \times B$ mod M can be performed in two different ways: obtaining the product then reducing it; or obtaining the reduced product directly. There are various algorithms that implement modular multiplication. High-performance modular multipliers on FPGAs are commonly realized by several small-sized multipliers, an adder tree for summing up the digit-products, and a reduction circuit. As an enhancement modified square root carry select adder algorithm is implemented to further reduce latency and area constraints.

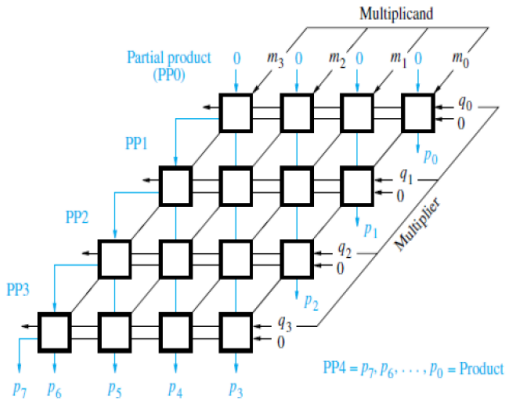
INTRODUCTION:
Electronic communication is growing exponentially so should be the care for information security issues [10]. Data exchanged over public computer networks must be authenticated, kept confidential and its integrity protected against alteration. In order to run successfully, electronic businesses require secure payment channels and digital valid signatures. Cryptography provides a solution to all these problems and many others [17]. One of the main objectives of cryptography consists of providing *confidentiality*, which is a service used to keep secret publicly available information from all but those authorized to access it. There exist many ways to providing secrecy. They range from physical protection to mathematical solutions, which render the data unintelligible. The latter uses *encryption/decryption* methods [10], [17], [30], [31]. The modular exponentiation is a common operation for scrambling and is used by several public-key cryptosystems, such Deffie and Hellman [8], [9] and the Rivest, Shamir and Adleman encryption schemes [34], as encryption/decryption method. RSA cryptosystem consiwsts of a set of three items: a *modulus M* of around 1024 bits and two integers *D* and *E* called *private* and *public* keys that satisfy the property $TDE \equiv T \mod M$. Plain text T obeying $0 \leq T < M$. Messages are encrypted using the public key as $C = TE \mod M$ and uniquely decrypted as $T = CD \mod M$. So the same operation is used to perform both processes: encryption and decryption. The modulus M is chosen to be the product of two large prime numbers, say P and Q. The public key E is generally small and contains only few bits set (i.e. bits = 1), so that the encryption step is relatively fast. The private key D has as many bits as the modulus M and is chosen so that $DE = 1 \mod (P-1)(Q-1)$. The system is secure as it is computationally hard toVdiscover P and Q. It has been proved that it is impossible to break an RSA cryptosystem with a modulus of 1024-bit or more. The modular exponentiation applies modular multiplication repeatedly. So the performance of publickey cryptosystems is primarily determined by the implementation efficiency of the modular multiplication and exponentiation. As the operands (the plaintext or the cipher text or possibly a partially ciphered text) are usually large (i.e. 1024 bits or more), and in order to improve time requirements of the encryption/decryption operations, it is essential to attempt to minimize the number of modular multiplications performed and to reduce the time required by a single modular multiplication. Modular multiplication $A \times B$ mod M can be performed in two different ways: multiplying, i.e. computing $P = A \times B$; then reducing, i.e. $R = P \mod M$ or interleave the multiplication and the reduction steps. There are various algorithms that implement modular multiplication. The most prominent are Karatsuba-Ofman's [12] and Booth's [3] methods for multiplying, Barrett's [2], [6], [7] method for reducing, and Montgomery's algorithms [18], and

Brickell's method [4], [37] for interleaving multiplication and reduction. curve cryptography (HECC) received much attention over the past years due to their small key sizes, which make them particularly suitable for resource constrained embedded devices. Well known schemes such as Diffie-Hellman key exchange or digital signature algorithm were extended to (H)ECC. In numerous works, these schemes were implemented on reconfigurable hardware, such as FPGAs, and optimized towards high-throughput and low-latency [1]– [6]. The performance of such schemes strongly depends on the implementation of the underlying field operations, e.g. modular addition, subtraction, multiplication, squaring, and inversion. In particular, the implementation of prime field multiplications should be optimized thoroughly because it requires a large amount of resources and time. Publickey cryptosystems using Mersenne prime field arithmetic recently gained in importance as Crandall's reduction procedure can be used [7].

MULTIPLICATION

Binary Multiplication:

Consider the multiplication of two integers in binary number system. This algorithm applies to unsigned numbers and to positive signed numbers. The product of two n-digit numbers can be accommodated in 2*n* digits, so the product of the two 4-bit numbers in this example fits into 8 bits. In the binary system, multiplication by the multiplier bit '1' means the multiplicand is entered in the appropriate position to be added to the partial product. If the multiplier bit is '0', then 0s are entered, as indicated in the third row of the shown example. Binary multiplication [2] of positive operands can be implemented in a combinational (speed up) two-dimensional logic array. Here, 'M' indicates multiplicand, 'Q' indicates multiplier & 'P' indicates partial product. The basic component in each cell is a full adder FA. The AND gate in each cell determines whether a multiplicand bit m_j, is added to the incoming partial-product bit, based on the value of the multiplier bit, q_i. For **i** in the range of 0 to 3, if $q_i = 1$, add the multiplicand (appropriately shifted) to the incoming partial product, PP_i, to generate the outgoing partial product, $PP_{(i+1)}$ & if $q_i = 0$, PP_i is passed vertically downward unchanged. The initial partial product PP_0 is all 0s. PP_4 is the desired product. The multiplicand is shifted left one position per row by the diagonal signal path. Since the multiplicand is shifted and added to the partial product depending on the multiplier bit, the method is referred as SHIFT & ADD method. The multiplier array & the components of each bit cell are indicated in the diagram, while the flow diagram shown explains the multiplication procedure.



MODULAR MULTIPLICATION:

In this method, the core operation of most algorithms for modular multiplication is addition. There are several different methods for addition in hardware: carry ripple addition, carry select addition and carry look-ahead addition and others[6]. The disadvantage of these methods is the carry propagation, which is directly proportional to the length of the operands. This is not a big problem for operands of size 32 or 64 bits but the typical operand size in cryptographic applications range from 160 to 2048 bits. The resulting delay has a significant influence on the time complexity of these adders. The carry save adder seems to be the most cost effective adder for our application.

Carry save addition is a method for an addition without carry propagation. It is simply a parallel ensemble of *n* full-adders without any horizontal connection. Its function is to add three *n*-bit integers *X*, *Y*, and *Z* to produce two integers *C* and *S* as results such that

$C + S = X + Y + Z$,

Where *C* represents the carry and *S* the sum. The i^{th} bit s_i of the sum *S* and the $(i + 1)^{st}$ bit c_{i+1} of carry *C* are calculated using the Boolean equations

$$s_i = x_i \oplus y_i \oplus z_i$$

$$c_{i+1} = x_i y_i \vee x_i z_i \vee y_i z_i$$

$$c_0 = 0,$$

When carry save adders are used in an algorithm one uses a notation of the form to indicate that two results are produced by the addition.

$(S, C) = X + Y + Z$

The results are now represented in two binary words, an n-bit word S and an $(n+1)$ Bit word C. Of course, this representation is redundant in the sense that we can represent one value in several different ways. This redundant representation has the advantage that the arithmetic operations are fast, because there is no carry propagation. On the other hand, it brings to the fore one basic disadvantage of the carry save adder.

MONTGOMERY MULTIPLICATION:

The Montgomery modular product S of A and B can be obtained as $S = A \times B \times R^{-1} \pmod N$, where R^{-1} is the inverse of R modulo N. That is, $R \times R^{-1} = 1 \pmod N$. Since the convergence range of S in MM algorithm is $0 \leq S < 2N$, an additional operation $S = S - N$ is required to remove the oversize residue if $S \geq N$. To eliminate the final comparison and subtraction, Walter changed the number of iterations and the value of R to $k + 2$ and $2k+2 \bmod N$, respectively. Nevertheless, the long carry propagation for the very large operand addition still restricts the performance of MM algorithm.

SCS BASED MONTGOMERY MULTIPLICATION:

To avoid the long carry propagation, the intermediate result S of shifting modular addition can be kept in the carry-save representation (SS, SC). Note that the number of iterations has been changed from k to $k + 2$ to remove the final comparison and subtraction. However, the format conversion from the carry-save format of the final modular product into its binary format is needed. The architecture of SCS-based MM algorithm(denoted as SCS-MM-1 multiplier) composed of one two-level CSA architecture and one format converter, where the dashed line denotes a 1-bit signal. A 32-bit CPA with multiplexers and registers (denoted as CPA_FC), which adds two 32-bit inputs and generates a 32-bit output at every clock cycle, was adopted for the format conversion. Therefore, the 32-bit CPA_FC will take 32 clock cycles to complete the format conversion of a 1024-bit SCS-based Montgomery multiplication. The extra CPA_FC probably enlarges the area and the critical path of the SCS-MM-1 multiplier.

DIGIT-PRODUCT GENERATION:

To begin, the position of the digit-products within the adder tree must be determined. Before combining fast reduction with digit-product accumulation, we use asymmetric tiling to compile a set of 4-tuples ($i, j, \mu l, \mu h$). Here i and j identify the indices of the input digits Ai and Bj , and hence connect the 4-tuple to the respective embedded multiplier in the DSP slice. The elements μl and μh denote the lowerand higher bit position within the adder tree. Algorithm 1

Algorithm 1 Determine position of digit-products.
Input: $A = [A0, ..., Ax-1]$, s.t. $Ai \in [0, 2m)$; $i \in [0, x)$
$B = [B0, ..., By-1]$, s.t. $Bj \in [0, 2n)$; $j \in [0, y)$
Output: $T = \{(i, j, \mu l, \mu h)\}$
1: for j from 0 to $y - 1$ do
2: for i from 0 to $x - 1$ do
3: $\mu l \leftarrow im + jn$ _ lowest bit
4: $\mu h \leftarrow (i + 1)m + (j + 1)n - 1$ _ highest bit
5: $T \leftarrow T \cup \{($
$i, j, \mu l, \mu h)\}$ _ add tuple
6: end for
7: end for
8: return T

describes the digit-product generation. We assume that the input operands A and B are decomposed by m and n respectively. The output of Algorithm 1 is a set T storing instances of the 4-tuple.

DIGIT-PRODUCT SPLITTING:

For combined reduction, all digit-product bits exceeding the Mersenne prime M must be shifted to the right by p bits. Digit-product bits exceeding the position $2p$ are unused and set to 0. As a consequence, they do not contribute to the multiplier result and can be removed For example, digit-products that partly exceed the Mersenne prime M need to be split in two parts. The upper part of the digit-product is then shifted to the right by p bits whereas the lower part remains unaltered. Instead, we suggest to perform the shifting for fast reduction and the subsequent regrouping on bit-level. Therefore, it is required to disassemble all digit-products i.e.instances of 4-tuples contained in T bit-wise. Algorithm 2 performs this procedure and also shifts the corresponding bits for combined reduction and removes unused ones. Bits are described by another 4-tuple described by ($i, j, \mu a, \mu r$).The identifiers i and j are inherited from the respective digitproduct. The absolute bit position μa represents the position within the adder tree, whereas μr describes the bit position relative to the corresponding digit-product. The relative bit position μr is required for the hardware design to associate a single bit with the correct DSP multiplier position. The output of Algorithm 2 is a set Z storing instances of bitsrepresented by the respective 4-tuple. The disadvantage of these methods is the carry propagation, which is directly proportional to the length of the operands. This is not a big problem for operands of size 32 or 64 bits but the typical operand size in cryptographic applications range from 160 to 2048 bits.

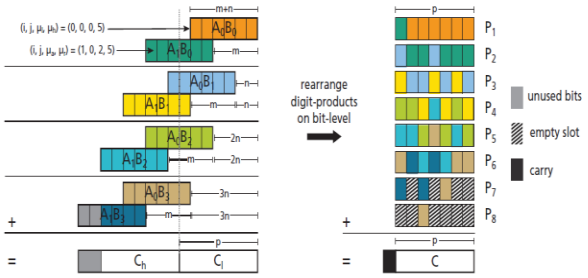
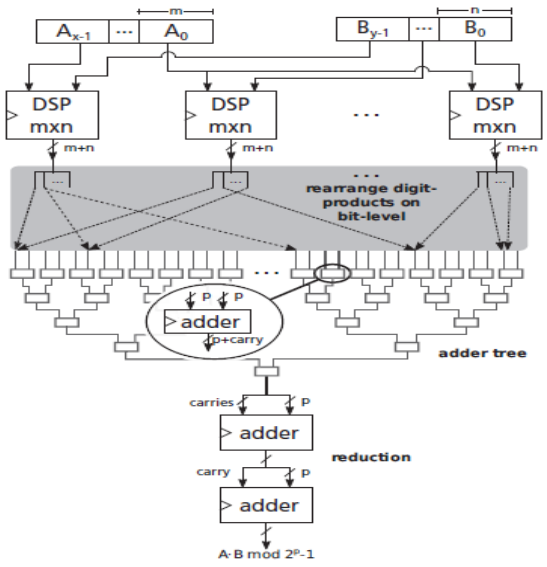


Figure . Adder tree optimized towards high-performance. Left: Digit-products generated for $m = 4$, $n = 2$ and $M = 27 - 1$. Right: Rearranged sliced digit-products to partial-products with combined fast reduction.

Algorithm 2 Slice digit-products in single bits.
Input: $T = \{(i, j, \mu l, \mu h)\}$, $M = 2p - 1$
Output: $Z = \{(i, j, \mu r, \mu a)\}$
1: for each t in T do
2: $(i, j, \mu l, \mu h) \leftarrow t$
3: for k in 0 to $(\mu h - \mu l)$ do

4: $v \leftarrow \mu l + k$
5: if $v < 2p$ then
6: $\mu r \leftarrow k$ _relative
7: $\mu a \leftarrow v \bmod p$ _absolute
8: $Z \leftarrow Z \cup \{(i, j, \mu r, \mu a)\}$ _ add tuple
9: end if
10: end for
11: end for
12: return Z

Reduced and equalized adder sizes allow a higher maximum clock frequency which translates to increased throughput and reduced latency



Above Figure depicts the hardware architecture of our modular multiplier for Mersenne primes. The hardware architecture is divided in four parts: the multiplication of digits using DSP multipliers, the subsequent rearranging of sliced digit-products to partial-products, the summation of partialproducts, and finally the two addition steps for full reduction. For high-performance purposes, DSP slices compute digit-products in full parallel. Furthermore, we make use of the registers that are embedded in each DSP slice. After all digit-products are obtained, the sliced DSP multiplier outputs are rearranged as discussed in previous sections. All single bits are grouped to partial-products which are then summed up with the subsequent adder tree. The rearrangement of digit-products to partial-products has no impact on area because it only translates to signal rewiring. We can pipeline the adder tree efficiently because all adders are similar sized translating to an equivalent circuit delay between register stages. With each adder tree level, the input size is increased by 1-bit corresponding to the carry of the previous addition. Once the accumulation of all partialproductsis completed, two extra additions are performed for full reduction. Finally, the result matches the modular multiplication i.e. $A \cdot B \bmod 2p - 1$.

BLOCK DIAGRAM OF CSLA WITH BEC:

The structure of the proposed 16-b SQRT CSLA using BEC for RCAwithCin=1 to optimize the area and power is shown in the figure.

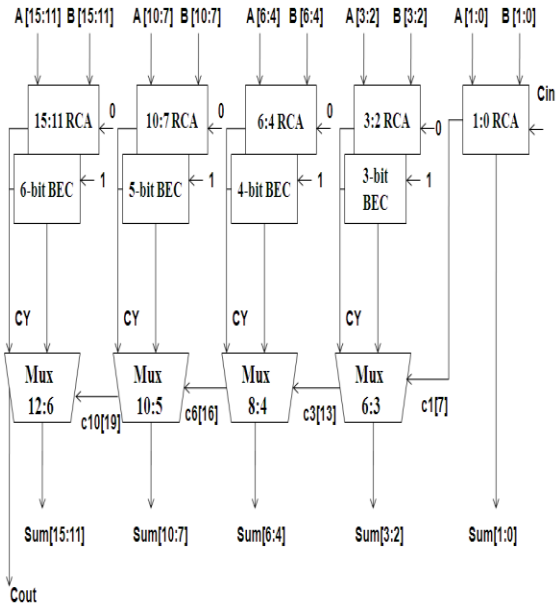
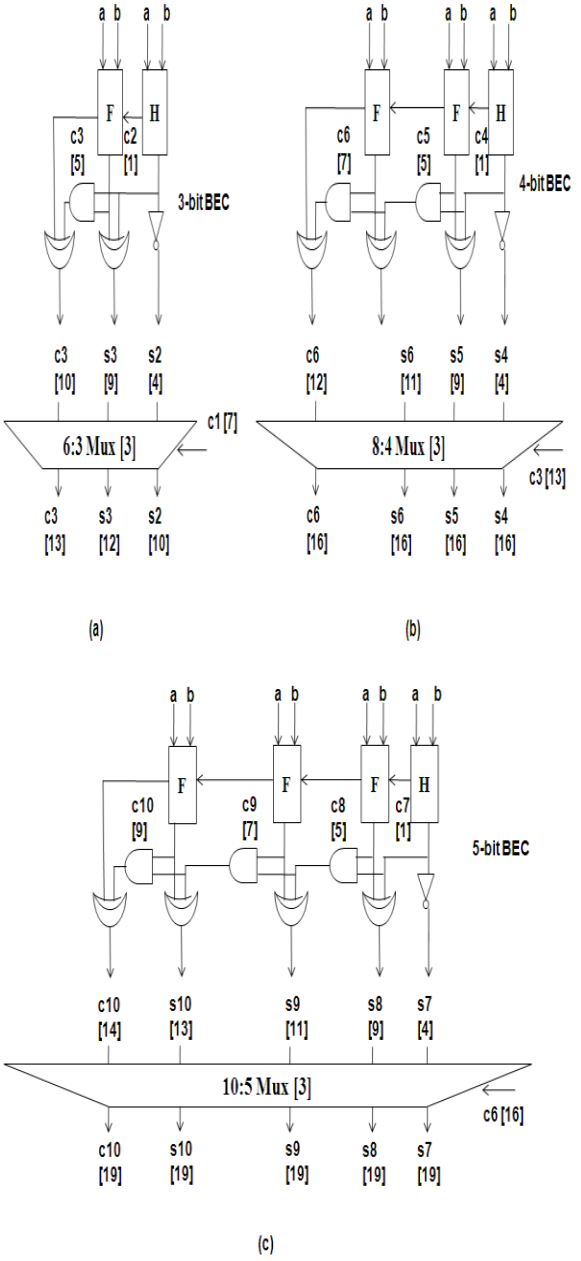


Fig: Modified system (Modified 16-b SQRT CSLA)

Comparing the block diagram of the regular square root CSLA with the modified square root CSLA, it can be seen that the RCA with Cin=1 is replaced by BEC (binary to excess-1 converter). This is done to reduce the area consumption. This can be seen after evaluating the group delay and the number of gates required for the design.

ARCHITECTURE OF MODIFIED MODEL:

The architecture of this modified model is shown in figure.



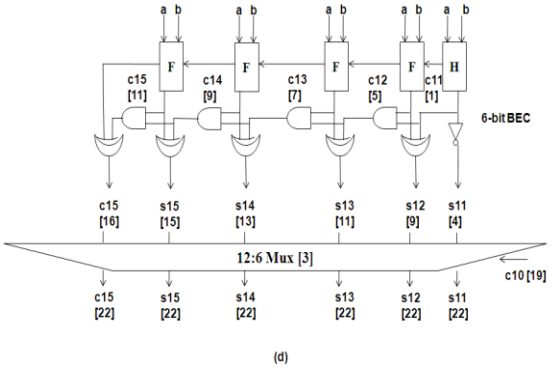
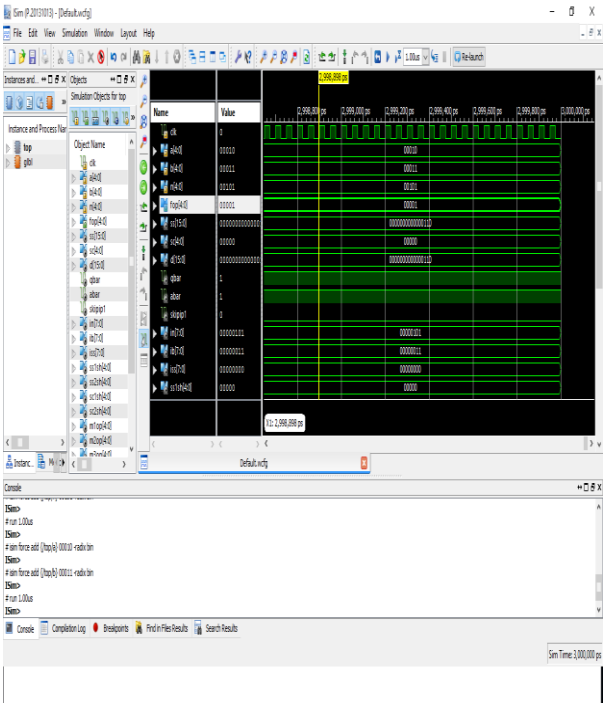


Fig: Delay and area evaluation of modified SQRT CSLA: (a) group2, (b) group3, (c) group4 (d) group5. H is a Half Adder.

RESULT:



CONCLUSION:

Finally, a high-performance modular multipliers for Mersenne primes on FPGAs based on a new optimization strategy of the adder tree and reduction circuits at the bit-level. The presented modular multiplier can operate at higher frequencies, leading to improved throughput and latency. The software to generate the hardware description of our modular multipliers in VHDL together with the respective test benches and test vectors are put in the public domain. Enhanced carry select adder improves area and latency parameters with at-most implicification.

REFERENCES:

[1] P. Sasdrich and T. G¨uneysu, "Implementing Curve25519 for Side- Channel–Protected Elliptic Curve Cryptography," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 9, no. 1, p. 3, 2015

[2] T. G¨uneysu and C. Paar, *Ultra High Performance ECC over NIST Primes on Commercial FPGAs*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 62–78. [Online]. Available: http: //dx.doi.org/10.1007/978-3-540-85053-3 5

[3] K. J¨arvinen, A. Miele, R. Azarderakhsh, and P. Longa, "FourQ on FPGA: New Hardware Speed Records for Elliptic Curve Cryptography over Large Prime Characteristic Fields," pp. 517–537, 2016. [Online]. Available: http://dx.doi.org/10.1007/978-3-662-53140-2 25

[4] P. Koppermann, F. D. Santis, J. Heyszl, and G. Sigl, "X25519 Hardware Implementation for Low-Latency Applications," in *2016 Euromicro Conference on Digital System Design (DSD)*, Aug 2016, pp. 99–106.

[5] W. N. Chelton and M. Benaissa, "Fast Elliptic Curve Cryptography on Fpga," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 16, no. 2, pp. 198–205, Feb 2008.

[6] P. Sasdrich and T. G¨uneysu, "Closing the Gap in RFC 7748: Implementing Curve448 in Hardware," Cryptology ePrint Archive, Report 2016/352, 2016, http://eprint.iacr.org/2016/352.

[7] R. Crandall, "Method and apparatus for public key exchange in a cryptographic system," 1992, uS Patent 5,159,632.

[8] C. Costello and P. Longa, "FourQ: four-dimensional decompositions on a Q-curve over the Mersenne prime," Cryptology ePrint Archive, Report 2015/565, 2015, http://eprint.iacr.org/2015/565.

[9] D. J. Bernstein, C. Chuengsatiansup, T. Lange, and P. Schwabe, "Kummer Strikes Back: New DH Speed Records," in *Advances in Cryptology – ASIACRYPT 2014: 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 317–337. [Online]. Available: http://dx.doi.org/10.1007/ 978-3-662-45611-8 17