Optimal Area and Delay Profile of FFT-Based Montegomery Multiplication Parallel Self Timed Adder

Sanskriti Yadav[1], Prof. Sujeet Mishra[2]

[1]*Mtech. Scholar,* [2]*Research Guide*

Department of Electronics and Communication Engg., Sanghvi Institute of Management and Science, Indore

Abstract - With the ongoing digital revolution and advances in high performance computing, powerful desktop computer systems are available to almost everybody at low cost. While there has always been a demand for hardware implementations of public key cryptography, the volume has risen dramatically in recent years, due to a paradigm. Because of its complexity, public–key cryptography is mainly used for digital signatures and the management of secret keys between two points. The encryption of bulk data is mainly established with secret–key cryptosystems, whereas the secret keys to be shared for a pair of users are distributed by public–key cryptosystems. Increasing demand for modular multiplication requires fast modular multiplication algorithms such as Montgomery multiplication the implementation, verification and testing of a Montgomery modular multiplication algorithm for the new efficient area and delay profile architecture of asynchronous parallel self timed adder based Montgomery multiplication reported in this work. This architecture proposes improvements on the well–known Montgomery multiplication algorithm and its previous implementations. The implementation and synthesis of proposed work has completed on Xilinx ISE design suite using hardware descriptive language HDL. The performance of proposed architecture has been evaluated based on area and delay profile.

Keywords- Asynchronous circuit, Parallel self timed adder, Montgomery Multiplication, delay profile, cryptosystems. FFT

I. INTRODUCTION

In the field of networking, role of network security is immense. In the age of information there is a need to keep information about every aspect of our live. These information needs to be hidden from unauthorized access (confidentiality), protected from unauthorized change (integrity), and available to an authorized entity when it is needed (availability). Hence the way of keeping the information securely is known as cryptography, which comes from a word with Greek origin, means “secret writing”. Many cryptographic algorithms are developed to achieve the above said goal. The algorithms should be such that an opponent cannot defeat its purpose. These algorithms generally consist of some arithmetic operations which are complicated and time consuming. It is because of the fact that these algorithms work with large amount of data either in blocks or simply in streams. Although a single traditional CPU is enough for performing these computations, but for a machine which works as a server in a huge network gets millions of client requests for performing cryptographic operations for them individually. This makes the workload huge. The computational resources may also be limited for example in smartcards, mobile phones, handheld computers, etc. Moreover if the associated network is of high speed, the speed of the necessary cryptographic computations also needs to be taken into account. For example in transmitting audio and video data for cable TV, video conferencing and sensitive financial and commercial data, the speed of the cryptographic module to be embedded, needs to be very high. So from the viewpoint of high speed and throughput, traditional software implementations of these complicated cryptographic algorithms are not efficient in real time applications like ATM, VPN, etc. This forces the system designers to go for hardware implementation of the cryptosystems. Fig. 1.1 basic encryption decryption algorithm.

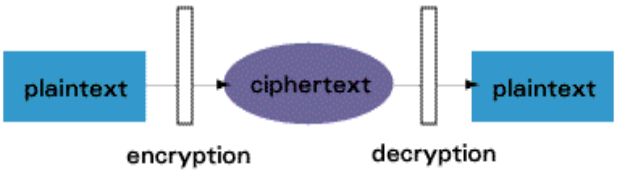

Fig. 1.1 Basic encryption decryption algorithm.

The rising growth of data communication technique and electronic transactions over the web has made system security to become the most important issue over the network. To provide modern security features, public-private key cryptosystems are used. One of such cryptosystem is RSA algorithm. Though computation in RSA takes more time by if the message to be encrypted is generated randomly then RSA will prove to be good cryptography algorithm for system security.

For the better working of RSA based cryptosystem the system has the public key for decryption and the user will have the device containing the private key assigned to the

user. And instead of entering the password the user will just have to insert the device to the system and the system will do the cross checking of the password for that particular user and allow access accordingly. As the user will not know the password as well as the password length so he can't give the password to any other person and the person will be solely responsible for any wrong doing in the system. The Montgomery algorithm computes P = (X*Y* (2n)-1) mod M. The idea of Montgomery is to keep the lengths of the intermediate results smaller than n+1 bit. This is achieved by interleaving the computations and additions of new partial products with divisions by 2; each of them reduces the bit- length of the intermediate result by one.

II. SELF-TIMED SYSTEM

A majority of the present-day digital systems are clock based or synchronous, which assume that signals are binary and time is discrete. In general, synchronous systems comprise a number of subsystems that change from one state to another depending on a global clock signal, with flip-flops (registers) being used to store the different states of the subsystems. A conventional synchronous system is portrayed by figure 2.1.

Fig. 2.1 A typical synchronous system stage.

Asynchronous circuits assume that signals are binary but the notion that time is not discrete. An asynchronous system is one in which there is no global synchronisation within the system; subsystems within the system are synchronised locally by the communication protocols between them. The results produced by the subsystems in an asynchronous system can be consumed by other subsystems as soon as they are generated without having to wait for a global clock tick.

Fig. 2.2 A typical asynchronous system stage.

Moreover in asynchronous systems, a sub-system can easily be replaced by another subsystem with the same functionality but with different performance, but this is not a straightforward task in case of a synchronous system as the clock period might have to be recomputed. An asynchronous system stage that involves request/acknowledge handshake (signal exchange) signalling protocol is shown in figure 2.2.

III. PROPOSED SYSTEM

The simplest way of adapting Montgomery's algorithm to large operand sizes would hence be, to just replace every arithmetic operation by its multi-precision equivalent. The criteria for selecting the most suitable algorithm are not limited to the number of multiplication operations alone. The specific architecture targeted for the implementation also plays an important role. Asynchronous parallel self timed adder based method is the most suitable one for implementation of modular multiplication. An area and delay efficient system has been proposed in this work implemented on Xilinx ISE design suite. RTL schematic of top module of proposed work has been shown in figure 3.1. To implement proposed system an efficient asynchronous parallel adder has been utilized to design a modular multiplication algorithm. As shown in figure there are two input pints in proposed design A (63:0) and B (63:0) are the 64 bit input vectors. M (63:0) is the 64 bit modular multiplier, clk is a clock input. Product (63:0) is the 64 bit product output. The proposed architecture is very useful in complex cryptographic application.

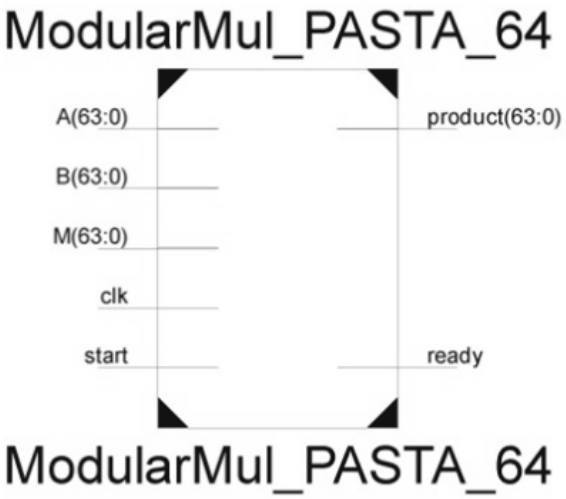

Fig. 3.1 RTL Schematic of Top Module of Proposed Architecture

Sub module of proposed module has been shown in Figure 3.2 RTL schematic of sub modules of proposed architecture. RTL schematic of expanded view of sub module has been shown in figure 3.3. Primitive arithmetic operations such as multiplication and addition are limited to a certain word size w. Operands of cryptographic algorithms, on the other hand, tend to be very large, so that multiple precision arithmetic comes into existence. The common trade-off when it comes to implementation of an

algorithm in hardware versus one in software is that flexibility is sacrificed for speed.

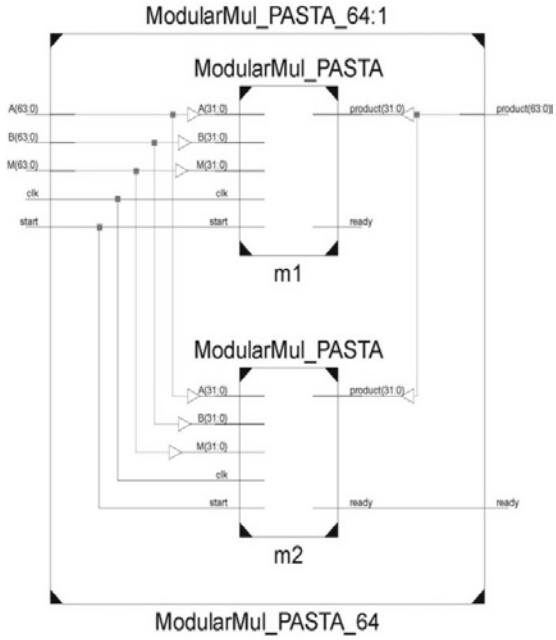

Fig. 3.2 RTL Schematic of Sub Modules of Proposed Architecture

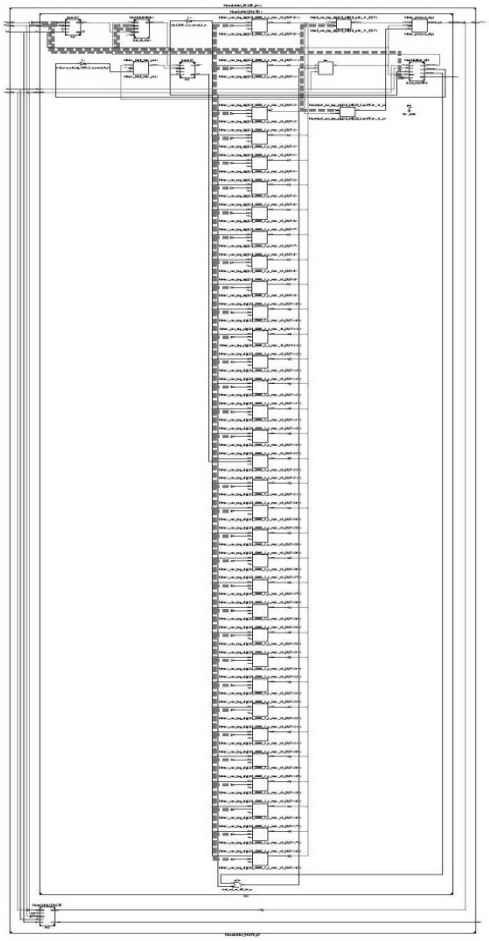

Figure 3.3 RTL Schematic of Expanded View of Sub Modules of Proposed Architecture

There are a number of different ways to improve on the performance of complex operations in hardware. While logic and arithmetic operations take at least one clock cycle each in software implementations, multiple logic operations can be combined into a single clock cycle in custom built hardware.

IV. SYNTHESIS OUTCOMES

Synthesis of proposed work has been done on Xilinx ISE simulated on ISIM HDL simulation environment. The components of the design were described the in structural VHDL code.

Table 1: Implementation Results Comparison with the Previous Architecture

Parameters	Previous Architecture	Proposed Architecture (PASTA)
Transform length / Number of digits (P)	64	64
Bit length of each digit (u)	32	32
LUTs	13975	789
Slices	3928	150
Period	5.34 ns	5.86 ns
Latency / Delay	5.67 us (5670 ns)	17.406 ns

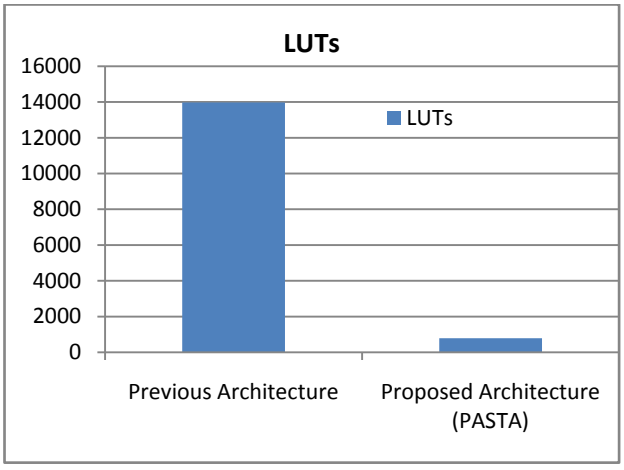

Figure 4.1 Graphical comparison of proposed work with existing work in terms of LUTs.

This approach makes the performance of the design less dependent on the synthesis features of the VHDL compiler suite. The correct function of the components has been verified using a VHDL testbench. A testbench essentially is a piece of behavioral VHDL code without any signals to the outside, that instantiates the component that is to be tested, also called Device Under Test (DUT), feeds specific data to its inputs (test vectors or patterns) and reads back the results, comparing them to the expected

results. Figure 4.2 synthesis screen of proposed work on Xilinx ISE 13.1.

The timing summary of proposed design has been shown in Figure 4.3 device utilization statistics and timing summary of proposed architecture achieved Clock period: 5.867ns (frequency: 170.457MHz). Table 1 shows the Implementation results comparison with the previous architecture. The graphical representation of LUTs comparison has shown in figure 4.1.

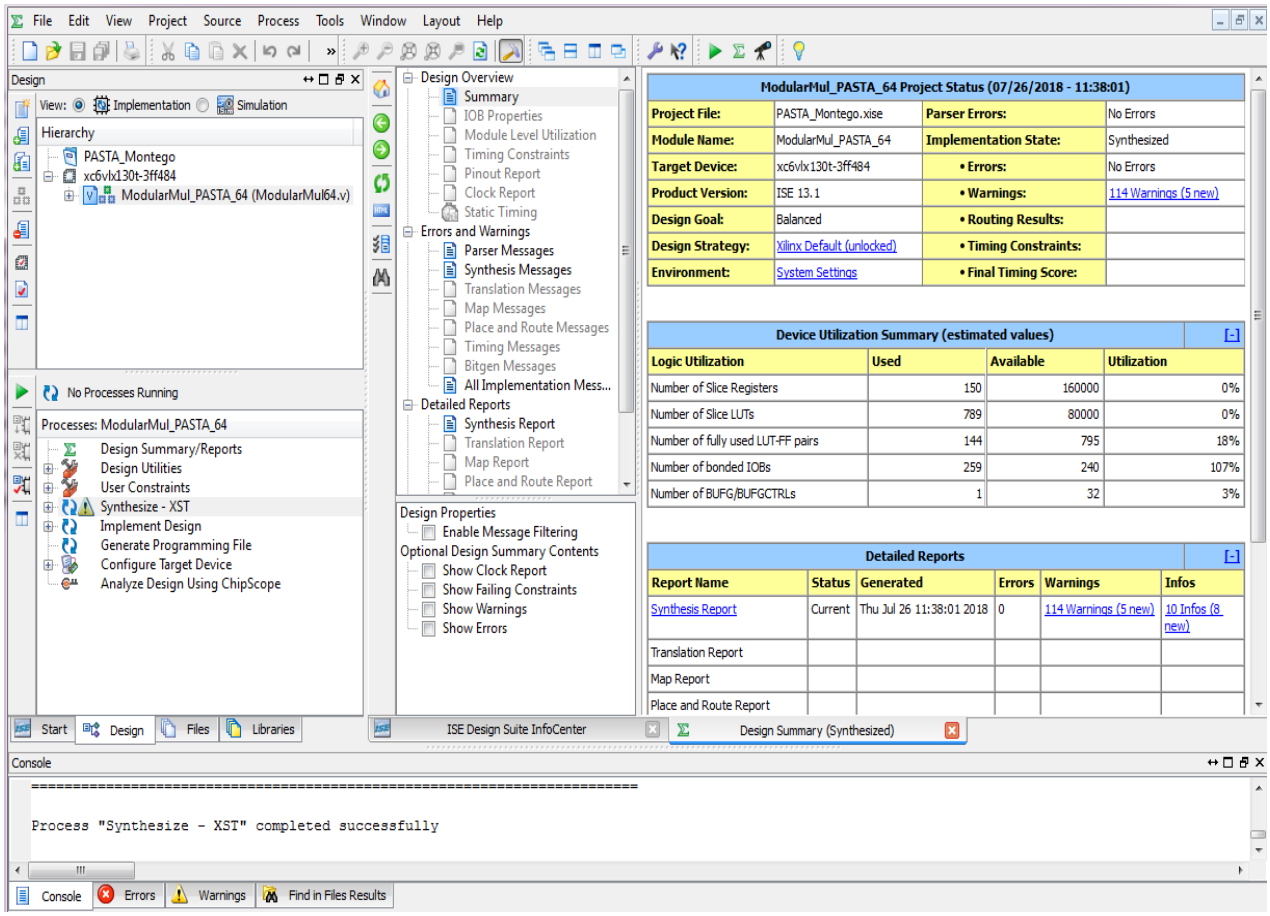

Figure 4.2 Device Utilization Summary of the Implementation on XILINX UI.

```
Device utilization summary:
---------------------------
Selected Device : 6vlx130tff484-3
Slice Logic Utilization:
 Number of Slice Registers:             150  out of  160000     0%
 Number of Slice LUTs:                  789  out of  80000      0%
    Number used as Logic:               789  out of  80000      0%

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:    795
   Number with an unused Flip Flop:     645  out of    795     81%
   Number with an unused LUT:             6  out of    795      0%
   Number of fully used LUT-FF pairs:   144  out of    795     18%
   Number of unique control sets:         4

IO Utilization:
 Number of IOs:                         259
 Number of bonded IOBs:                 259  out of    240   107%
=========================================================================

Timing Details:
---------------
All values displayed in nanoseconds (ns)
=========================================================================
Timing constraint: Default period analysis for Clock 'clk'
  Clock period: 5.867ns (frequency: 170.457MHz)
  Total number of paths / destination ports: 14375 / 214
-------------------------------------------------------------------------
```

Fig. 4.3 Device Utilization Statistics and Timing Summary of Proposed Architecture

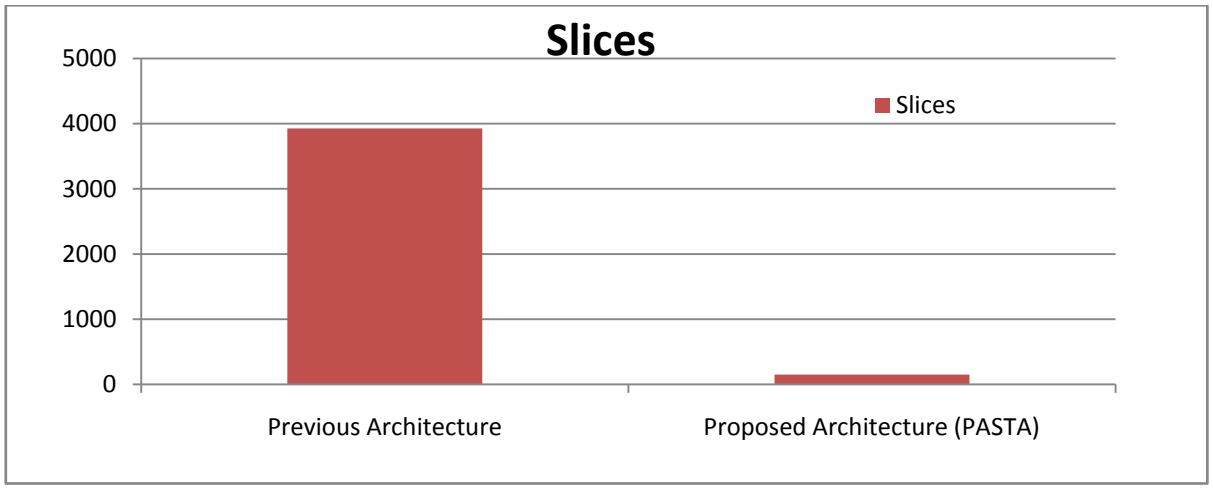

Fig. 4.4 Graphical comparison of proposed work with existing work in terms of Slices.

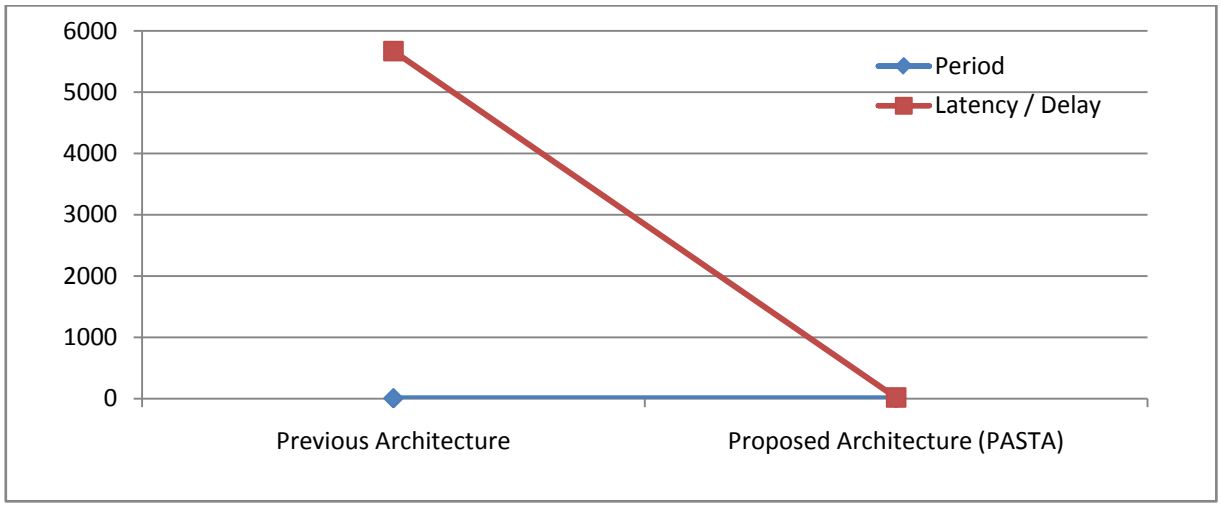

Fig. 4.5 Delay comparisons.

V. CONCLUSION AND FUTURE SCOPES

A new architecture for efficient area and delay profile architecture of asynchronous parallel self timed adder based Montgomery multiplication algorithm has been proposed, which combines positive features from previously proposed architectures with recent advances in digit multiplier design. The outcome of proposed work has highly scalable design with the ability to perform integer and binary arithmetic multiplication operation at high speeds. Analysis and comparison of previous work results with proposed work results proved the efficiency and advances of proposed work. The most important outcome of this analysis was to identify delay and area utilized to implement and synthesize design. In this work the architecture has been implemented in VHDL, synthesized and tested successfully.

The architecture presented in this work can be considered for future work to improve performance in terms of high seeped and less area. In the current/future era where issues such as reliability and variability tend to assume greater significance than quality-of-results. The mathematical proof of the control logic for these architectures can be an interesting area of study.

REFERENCES

[1] W. Dai, D. D. Chen, R. C. C. Cheung and Ç. K. Koç, "Area-Time Efficient Architecture of FFT-Based Montgomery Multiplication," in IEEE Transactions on Computers, vol. 66, no. 3, pp. 375-388, March 1 2017.

[2] D. D. Chen, G. X. Yao, R. C. C. Cheung, D. Pao and Ç. K. Koç, "Parameter Space for the Architecture of FFT-Based Montgomery Modular Multiplication," in IEEE Transactions on Computers, vol. 65, no. 1, pp. 147-160, Jan. 1 2016.

[3] R. Verma, M. Dutta and R. Vig, "FPGA implementation of RSA based on carry save Montgomery modular multiplication," 2016 International Conference on Computational Techniques in Information and Communication Technologies (ICCTICT), New Delhi, 2016, pp. 107-112

[4] W. Hentabli and F. Merazka, "An extension of RSA_512 to RSA_1024 core under hardware platform based on montgomery powering," 2015 10th International Conference for Internet Technology and Secured Transactions (ICITST), London, 2015, pp. 448-453.

[5] B. Hanindhito, N. Ahmadi, H. Hogantara, A. I. Arrahmah and T. Adiono, "FPGA implementation of modified serial montgomery modular multiplication for 2048-bit RSA cryptosystems," 2015 International Seminar on Intelligent Technology and Its Applications (ISITIA), Surabaya, 2015, pp. 113-118.

[6] B. G. Aswathy and R. Resmi, "Modified RSA public key algorithm," 2014 First International Conference on Computational Systems and Communications (ICCSC), Trivandrum, 2014, pp. 252-255.

[7] A. Nadjia and A. Mohamed, "High throughput parallel montgomery modular exponentiation on FPGA," 2014 9th International Design and Test Symposium (IDT), Algiers, 2014, pp. 225-230.

[8] R. L. Rivest, A. Shamir, and L. Adleman, “A method for obtaining digital signatures and public-key cryptosystems,” Commun. ACM, vol. 21, no. 2, pp. 120–126, 1978.

[9] R. L. Rivest, “A description of a single-chip implementation of the RSA cipher,” Lambda, vol. 1, no. Oct.–Dec., pp. 14–18, 1980.

[10] “Recommendation for key management,” NIST, Tech. Rep. Spe- cial Publication 800-57, Part-1, Rev.-3, 2012.

[11] P. L. Montgomery, “Modular multiplication without trial divi- sion,” Mathematics Comput., vol. 44, no. 170, pp. 519–521, 1985.

[12] A. Karatsuba and Y. Ofman, “Multiplication of multidigit num- bers on automata,” Soviet Physics Doklady, vol. 7, 1963, Art. no. 595.