FPGA Implementation of a Cryptographic Accelerator for IPSec authentications

Kwang-Youb Lee[1] and Jae-Chang Kwak[2]
[1] Department of Computer Engineering
[2] Department of Computer Science
Seokyeong University
16-1,Chongnung Dong,Songbuk Ku,Seoul,136-704,Korea
e-mail : kylee@skuniv.ac.kr, jckwak@skuniv.ac.kr

Abstract: IPSec authentication provides support for data integrity and authentication of IP packets. Authentication is based on the use of a message authentication code(MAC). Hash function algorithm is used to produce MAC , which is referred to HMAC. In this paper, we propose a cryptographic accelerator using FPGA implementations. The accelator consists of a hash function mechanism based on MD5 algorithm, and a public-key generator based on a Elliptiv Curve algorithm with small scale of circuits. The accelator provides a message authentification as well as a digital signature. Implementation results show the proposed cryptographic accelerator can be applied to IPSec authentications.

1. Introduction

IPSec provides the capability to secure communications across a LAN, across private and public wide area networks(WANs), and across the Internet. When IPSec is implemented in a firewall or router, it provides strong security that can be applied to all traffic crossing the perimeter. Traffic within a company or workgroup does not incur the overhead of security-related processing.

IPSec provides security services at the IP layer by enabling a system to select required security protocols. Two protocols are used to provide security : Authentication Header(AH), Encapsulating Security Payload(ESP). The Authentication Header provides support for data integrity and authentication of IP packets. Authentication is based on the use of a message authentication code(MAC).

A MAC, also known as a cryptographic checksum, is generated by a function C of the form MAC=$C_k(M)$. Hash function algorithm is used to produce MAC , which is referred to HMAC. SHA-1, HAS-160 and MD5 are hash algorithms, which have been specified for IPSec.

MD5 is a message digest algorithm developed by Ron Rivest at MIT[1]. It is basically a secure version of the previous algorithm, MD4 which is a little faster than MD5. Until last few years, when both brute-force and cryptanalytic concerns have arisen, MD5 was the most widely used secure hash algorithm. The algorithm takes a message of arbitrary length as an input and produces a 128-bit message digest as an output. The input is processed in 512-bit blocks.

In this paper, we construct a cryptographic accelerator using Field Programmable Gate Arrays (FPGA) implementations. The accelator consists of a hash function mechanism and a public-key generator. The hash function is based on MD5 algorithm. The public-key generator is based on a Elliptiv Curve algorithm with small scale of circuits. Each implementation result is provided. The cryptographic accelator provides a message authentification function as well as a digital signature function.

2. Message Authentication Architectures

An authentication technique involves the use of a secret key to generate a small fixed-size block of data, known as a cryptographic checksum or MAC, that is appended to the message. This technique assumes that two communicating parties A and B, and shares a common secret key k. When A has a message to send to B, it calculates the MAC as a function of the message and the key: MAC=$C_k(M)$. The message plus MAC are transmitted to the intended recipient. The recipient performs the same calculation on the received message, using the same secret key, to generate a new MAC. The received MAC is compared to the calculated MAC[2].

Figure 1 illustrates two ways in which a hash code can be used to provide a message authentication. In Figure 1(a), the message plus concatenated hash code is encrypted by a conventional encryption method. The hash code provides a structure required to achieve an authentication. Because encryption is applied to the entire message plus hash code, confidentiality is also provided. In Figure 1(b), only the hash code is encrypted, using public-key encryption and using the sender's private key. This provides authentication and a digital signature, because only the sender could have produced the encrypted hash code. In fact, this is the essence of the digital signature technique.

When confidentiality is not required, architecture (b) has an advantage over architecture (a) in the sense that less computation is required.

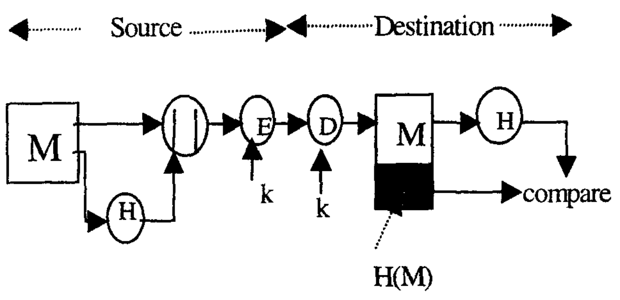
3. Design of a Public-Key Generator

Security issues will play an important role in the most of computer communications in the future. Elliptic curve cryptosystems allow for shorter operand lengths than other public-key schemes based on the discrete logarithm in finite fileds. The implementation of a crypto engine in this paper is based on elliptic curves.

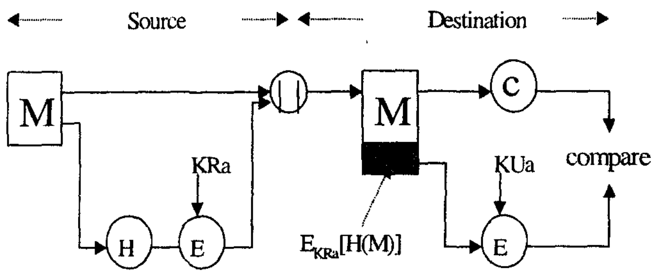
3.1 Elliptic Curve(EC) Scalar Multiplier

The basic operation in an EC cryptosystem is the scalar multiplication over the elliptic curve and the most efficient method for computing EC scalar multiplication is to use an double/addition method[3].

A scalar multiplication is done with a series of double/addition of elliptic curve points. In turn, each double/addition of EC points consists of a series of underlying field additions, squarings, multiplications and



(a). Message authentication and confidentiality



(b). Message authentication based on public-key encryption

Figure 1. Message Authentication Architectures

inversions. Fig. 2 shows the partitioning of the design into four levels. The elliptic curve used in this implementation is defined by Weierstrass equations as:
$y^2 + xy = x^3 + ax^2 + b$ where $a,b \in GF(2^m)$ and $b \neq 0$.

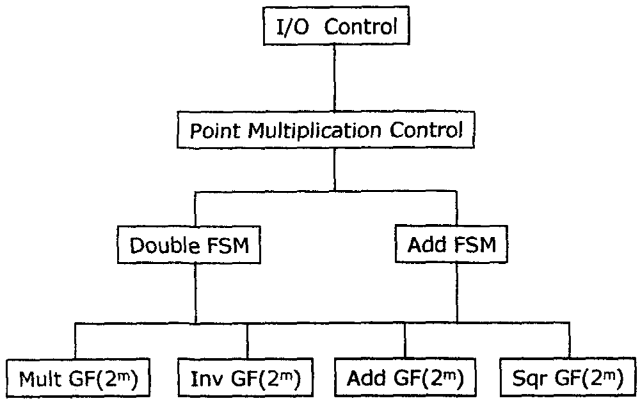


Figure 2. Design hierarchy

3.2 Proposed Architecture

We propose the structure of a combined operator architecture for Galois fields in standard basis. The proposed architecture combine multiplier, squarer, adder into a inverter, which is based on Almost Inverse Algorithm. This method offers a compact implementation of EC scalar multiplier. In Figure 3, multiplication and squaring is operating on register Z, B, and D , which are used for an inversion. Register Z contains a 193bit multiplicand. Also, register B contains a 193bit multiplier. The product of multiplication remains in the 193bit register D[4].

3.3 FPGA implementation and Results

The scalar multiplier is implemented with a parameterized VHDL description and is synthesized /mapped to a Xilinx FPGA(XCV800). By changing parameters for elliptic curves, a different instance can be acquired. Table 1 shows the timing comparison of a recent software Implementation[5] and the implemented FPGA chip. The speed-up ratios show that overall processing time is reduced by almose six times, which verifies the efficient hardware implementation.

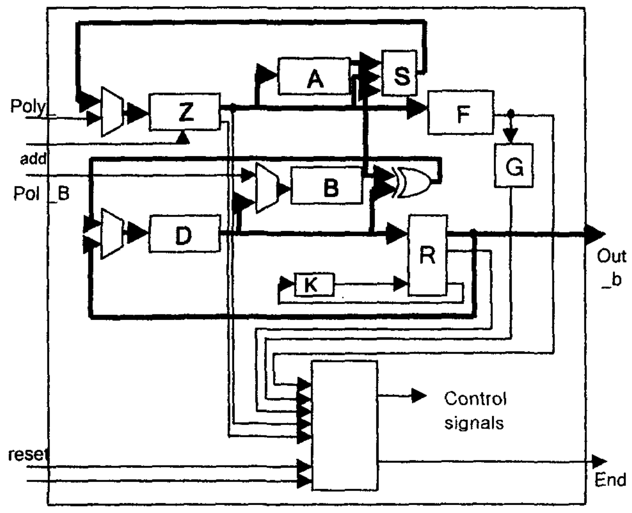


Figure 3. Structure of Public-Key generator based on EC

Table 1. Timing Comparison of a Recent Software Implementation and the EC scalar multiplier

Operation	Time in µ sec		Speed-up
	Software over $GF(2^{191})$	EC scalar multi. over $GF(2^{193})$	
Addition	0.6	0.1	6
Multiplication	39.0	4.5	8.7
Inversion	126.0	26.0	4.8
EC Addition	215.0	39.3	5.5
EC Doubling	220.0	39.0	5.6

4. Implementation of a MD5 Algorithm

The MD5 algorithm is designed to be quite fast on 32-bit machines. In addition, the MD5 algorithm does not require any large substitution tables; the algorithm can be coded quite compactly. The MD5 algorithm is an extension of the MD4 message-digest algorithm [6]. MD5 is slightly slower than MD4, but is more conservative in design.

4.1 MD5 Algorithm description

The MD5 algorithm involves repeated uses of a compression function, f, that takes two inputs(an 128-bit input from the previous step, called the chaining varable, and a 512-bit block) and produces and 128-bit output. The message is padded to ensure that its length in bits plus 64 is divisible by 512. That is, its length is congruent to 448 modulo 512. Padding is always performed even if the length of the message is already congruent to 448 modulo 512. Padding consists of a single 1bit followed by the necessary number of 0bits.A 64bit binary representation of the original length of the message is concatenated to the result of above one. The expanded message at this level will exactly be a multiple of 512bits.

4.2 Design of MD5 logics

The MD5 message digest algorithm takes a message of arbitrary length as an input and produces a 128-bit message digest as an output. The input is processed in 512-bit blocks. The MD5 algorithm has the property that every bit of the hash code is a functions of every bit in the input. The complex repetition of the basic functions(F,G,H,I) produces the results that are well mixed. This design has a Full Loop Unrolling Architecture. This architecture has a 64-step combinational logic core. The barrel shifter has been removed by direct wiring. The use of double buffering eliminates the loading time from the critical timing path. In addition to the core, the other main component is the padding circuits. Figure 4 shows the block diagram of the MD5 design.

For counting the length of the message, the "LASTBYTE" signal makes the start of counting the number of bytes. Then, the number of input bytes was shifted by 3bit-left

For example, if the message is less than 55byte (440bit), it is padded by "100...00" to a length of 512bit except the length of message. After all padding has been processed, the output from the 64 steps is put into the register A,B,C,D.

If the message is longer than 56byte(448bit), it is padded by "100....00" to a length of 512 bit. After padding and processing from the 64 steps, the "000.....000 56byte (448bit) is concatenated with the length of message(64bit). Then, the output from final 64 steps is made.

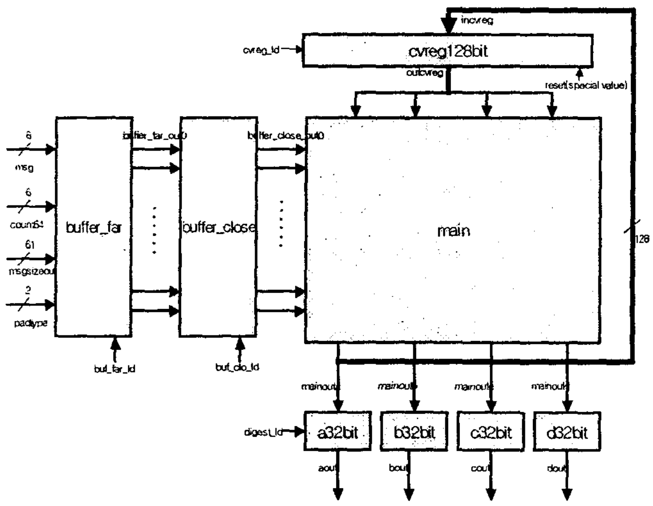


Figure 4. Block diagram of a MD5

4.3 FPGA implementation and Results

The top-level design was described in VHDL and the FPGA was utilized. This VHDL code was synthesized and implemented on the Virtex XCV800 target device with clock rate up to 100MHz. The utilization of slices was 5,067 out of 9,408 (53 %).

For the full-loop design, the performance is about 500Mbps @ 100MHz. According to the performance measurements on software implementations given in RFC 1810, the throughput has been less than 100Mbps. DEC Alpha(190MHz) has given a throughput of 87-100Mbps[7].

5. Conclusions

In this paper, we propose a cryptographic accelerator using Field Programmable Gate Arrays (FPGA) implementations. The accelator consists of a hash function mechanism and a public-key generator. The hash function is based on MD5 algorithm. The public-key generator is based on a Elliptiv Curve algorithm with small scale of circuits. The cryptographic accelator provides a message authentification function as well as a digital signature function. Implementation result show that the public key generator can generate a key within 6 msec. And the perfomrance of hash function mechanism is about 500Mbps @ 100MHz. With this implementation results, the proposed cryptographic accelerator can be applied to IPSec authentications.

● This work was supported by System2010 and IDEC

References

[1]. R. Rivest, *The MD5 Message-Digest Algorithm*, RFC 1321,MIT LCS & RSA Data Security, Inc.,April 1992.
[2]. W. Stallings, *Cryptography and Network Security*, Prentice Hall, 1999.
[3]. J.A.Solinas,"An Improved algorithm for architecture on a family of elliptic curves", *Journal of Cryptography*, 1997.
[4]. J.S.Ha,Y.H.Kim,K.Y.Lee,"Compact implementation of Elliptic Curve Cryptography System using a FPGA",The 9th Korean conference on Semiconductors,pp813-814,Feb.,21-22,2002.
[5]. M.A.Hasan, A.G. Wassal,"VLSI Algorithms, Architecture, and Implementation of a Versatile GF(2^m) Processor", *IEEE Trans. Computers*, vol 49, no. 10, pp1064-1073, Oct.,2000.
[6]. R. Rivest, *The MD4 Message-Digest Algorithm*, RFC 1320,MIT LCS & RSA Data Security, Inc.,April 1992.
[7]. J. Touch, *Report on MD5 Performance*, RFC 1810, June 1995.