Designing a high performance cryptosystem for video streaming application

- **Nguyen Van Toan**
- **Do Quoc Minh Dang**
- **Nguyen Duc Phuc**
- **Huynh Huu Thuan**
- **Nguyen Dinh Thuc**

University of Science , VNU-HCM

(Received on December 05 th 2014, accepted on September 23rd 2015)

ABSTRACT

This paper presents the hardware design of a high performance cryptosystem for video streaming application. Our proposed system is the combination of two cryptographic algorithms, symmetric key algorithm and asymmetric key algorithm (also called public key algorithm) to take their benefits. The symmetric key algorithm (ZUC) is used to encrypt/decrypt video, and the public key algorithm (RSA) performs the encryption/ decryption for the secret key. This architecture has high performance, including high security and high processing bit rate. High security is achieved due to the ease of key distribution of the asymmetric key cryptosystem and the secret key can be easily changed. The high processing bit rate of video encryption/decryption is the result of the high speed of encryption/decryption of the symmetric key algorithm. The H.264 video decoder is also integrated into this system to test the functionality of the proposed cryptosystem. This system is implemented in Verilog-HDL, simulated by using the ModelSim simulator and evaluated by using Altera Stratix IV-based Development Kit. The speed of video decryption achieves up to 4.0 Gbps at the operating frequency of 125 MHz, which satisfies applications with high bandwidth requirement such as video streaming.

Keywords: *cryptosystem, encryption, decryption, RSA, ZUC, FPGA.*

INTRODUCTION

Nowadays information security is a subject with a high interest. The development of computer networks, particularly the Internet, results more and more applications and services are carried out electronically, for example, PayTV, video streaming, internet-banking, and so on. Since the information on of these applications and services are possible transmitted in insecure channels, the demand of information security becomes essential. The increase of the demand of information security makes cryptography to become important.

Symmetric key cryptography uses the same key for both encryption and decryption. The advantage of symmetric key algorithms is that their execution is fast [1]. However, the critical issue of the symmetric key cryptosystem is the secret key distribution. On the other hand, the public key algorithm uses a pair of keys (public key and private key) to perform data encryption and decryption. The advantage of the public key cryptosystem is that providing public keys is easier than distributing secret keys securely [2]. However, the execution of public key algorithms is much slower than the execution of symmetric key algorithms. A hybrid cryptographic system in [2] was implemented by combining Advanced Encryption Standard (AES), Data Encryption Standard (DES) and public key algorithm (RSA), which offer benefits in key distribution and high security [2]. The data block is encrypted by using AES or DES while their secret keys are encrypted by using RSA algorithm. The encrypted secret key is then concatenated with the encrypted data to form the packets and sent to the destination. This implementation does not need key exchange separately [2]. However, every data block contains the encrypted key and each data block is encrypted by using a different session key, which does not save the transmission bandwidth. And the system must decrypt the secret key completely before data decryption and this is not appropriate with video streaming application. The system was proposed in [3] included 1024-bit RSA algorithm, 163-bit Elliptic Curve Cryptography (ECC) and 128-bit AES. In this system, AES was used to encrypt the transferred document to produce cipher-text, and RSA (or ECC) provided encryption/decryption for the secret key. This system also achieves high security. However, it does not allow us to change the secret key during data transfer. Both works [2, 3], AES cryptosystem (block cipher) was used to encrypt data. The drawback of the blocks cipher are: (1) data block needs to be padded if its size is less than block size, (2) be suffered error propagation, (3) the speed of encryption/ decryption is less than that of a stream cipher.

Our proposed cryptosystem combines the ZUC stream cipher [4] and the public key cipher RSA with 1024-bit key length. RSA is widely used public key algorithm [1]. The ZUC cipher is the new stream cipher that is commonly used in many countries [5]. It is simple, faster than block cipher [1]. The video content is encrypted/decrypted by using ZUC algorithm. And the secret key is encrypted/decrypted by using RSA algorithm. The encrypted symmetric key is then concatenated with the encrypted video to form the transmitted packets. In addition, our system allows to change the secret key. In case of no key changing, the encrypted key is not present in the transmitted packets, which saves the transmission bandwidth. Additionally, we build the system that enables to decrypt a new secret key and video in parallel. That means while RSA core is decrypting new secret key, ZUC core still uses the current secret key for data decryption. This feature was not implemented in the existing systems [2-3]. It is also difficult to implement this feature by software. Our proposed system achieves high security and speed which is very suitable for real time applications. In this paper, we focus on the implementation of the hardware architecture of cryptosystem for video streaming application.

SYSTEM ARCHITECTURE

The overall block diagram of the proposed embedded system

The block diagram of the proposed embedded system is shown in Fig. 1.

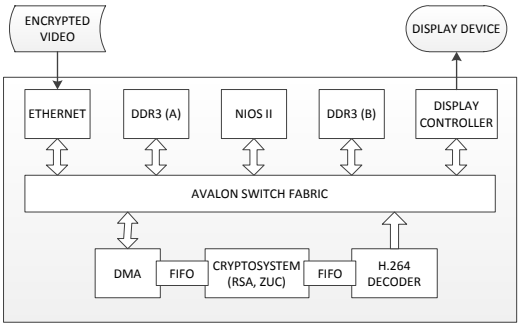

Fig. 1. The overall block diagram of the proposed embedded system

The encrypted data (the encrypted secret key and the encrypted video stored in Server) are streamed to the evaluation board via an Ethernet interface and are stored into DDR3 (A). DMA module reads the encrypted data from DDR3 (A) and pushes them into FIFO. The cryptosystem reads the encrypted data from the FIFO to decrypt the video content. Firstly, the RSA coprocessor decrypts the secret key. Secondly, the ZUC coprocessor uses that secret key to generate a keystream to decrypt the video content (video in compressed H.264 format). Thirdly, the video content is pushed into another FIFO. When the video content is available in the FIFO, the H.264 video decoder decodes the video content and writes it to DDR3 (B). Finally, the display controller reads video from DDR3 (B) and sends it to the display device. H.264 decoder module has a feature of being capable to decode H.264/AVC baseline profile video of VGA resolution (640x480) with 25 frames per second at the clock frequency of 25 MHz. Output frame format is in 4:2:0 YCbCr sampling format.

The block diagram of the proposed cryptosystem

Our proposed cryptosystem is the combination of ZUC algorithm and RSA algorithm. The RSA algorithm is used to encrypt/decrypt the secret key (key of ZUC algorithm). ZUC algorithm provides the encryption/decryption for the video content. Fig. 2 illustrates our proposed cryptosystem.

DECRYPT CONTROLLER controls to read the encrypted secret key from FIFO to its registers. And then RSA coprocessor performs to decrypt the secret key. When RSA coprocessor completes its decryption, it indicates to ZUC coprocessor by asserting zuc_key_valid signal. The ZUC coprocessor then loads the secret key into its LFSR and produces a keystream. The video content is recovered by XORing the encrypted video and the generated keystream. The decrypted video will be stored in the FIFO. Whenever the secret key needs to be changed (through the signaling in the header of the received packets), the RSA decrypts that new secret key while ZUC still uses the current key to produce the keystream for decrypting the video content. As soon as RSA coprocessor completes its operation, and the signaling in the received packet indicates to apply the new secret key, ZUC coprocessor then uses that new secret key to generate a keystream for the next decryption. Fig. 3 shows the frame format of each transmitted packet. It is made of the encrypted video, the encrypted secret key and the signaling. The signaling aims to: (1) when a new encrypted secret key is coming, (2) when a new secret key is applied.

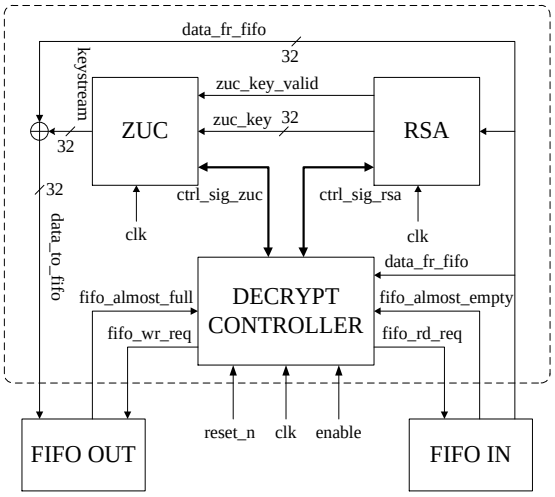

Fig. 2. The proposed cryptographic system

Fig. 3. Encrypted packet

The advantages of our system are as follows

High security is achieved because the secret key is encrypted with the RSA algorithm, and there is no key establishment separately before data transferring.

We can change the secret key at anytime without key re-establishment as in the traditional cryptosystem.

Our system saves the transmission bandwidth by eleminating the encrypted secret key in the packets that is sent in case of no key changing.

Our proposed system enables to decrypt a new secret key and the encrypted video in parallel, which makes better the quality of service e.g., video decryption is performed continuously and smoothly.

Design of ZUC

ZUC is a word-oriented stream cipher [4]. It takes a 128-bit initial key and a 128-bit initial vector as input, and outputs a keystream of 32-bit words. The architecture of ZUC stream cipher is proposed as Fig. 4. The top layer is a linear feedback shift register (LFSR) that consists of 16 of 31-bit registers. The middle layer is bit reorganization (BR) that extracts 128 bits of registers of LFSR to form 4 of 32-bit words. The first three words are the inputs of nonlinear function F, and the last word is used in keystream generation. The bottom layer is the nonlinear function F that takes three words X0, X1, X2 as inputs and outputs 32 bit word W. The outputted keystream is shifted into a 32-bit register.

The LFSR has two operation modes: initialization mode and working mode. In initialization mode, the LFSR receives 31 bits of W (bit 31 to 1) as its input. In the working mode, the LFSR does not receive any input, and produces a 32-bit word per clock cycle. In hardware implementation, we use a multiplexer to select the input for these modes. We found that the critical path in the ZUC architecture is the circuit used to update LFSR in the initialization stage and the working stage. There is a chain of six modulo $(2^{31} - 1)$ additions to compute the value of S_{16}. Therefore, the timing optimization of this critical path improves the operating frequency of ZUC core. The expression of S_{16} is given in equation (4).

$$v=2^{15}S_{15}+2^{17}S_{13}+2^{21}S_{10}+2^{20}S_4+(1+2^8)S_0 \mod (2^{31}-1) \quad (3)$$

$$S_{16}=[v+(W>>1)] \mod (2^{31}-1) \quad (4)$$

We propose to use carry save adders (CSA) to calculate the intermediate values and ripple carry adder to calculate the final result. The hierarchical CSA tree is shown in the Fig. 5. In this architecture, one multiplexer selects the mode of LFSR: initialization mode or working mode. To perform modulo $(2^{31} - 1)$ addition, for each addition of CSA, carry is cyclic left-shifted by one bit. This implementation helps to improve the timing significantly because the delay of CSA is exactly equal to the delay of 1-bit full adder.

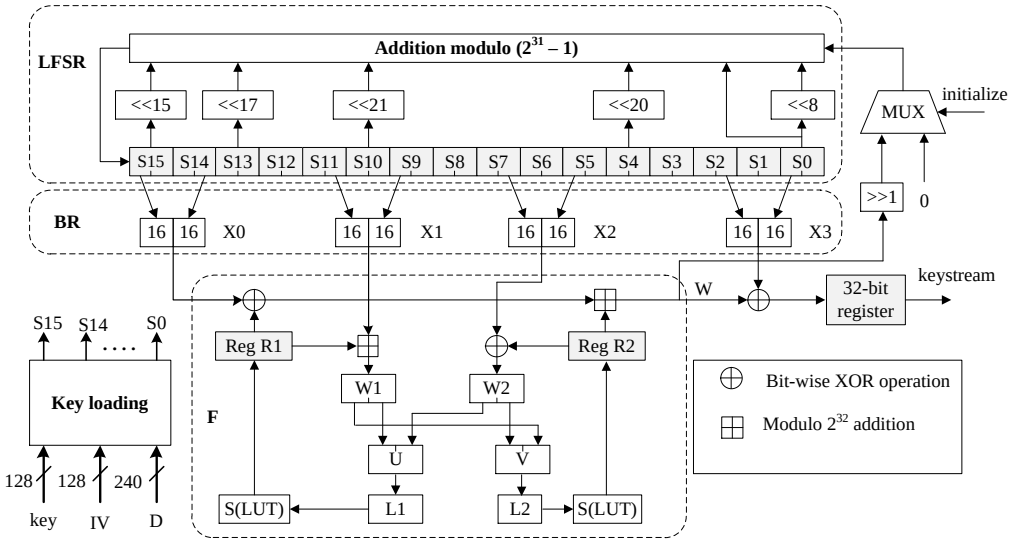

Fig. 4. Architecture of ZUC

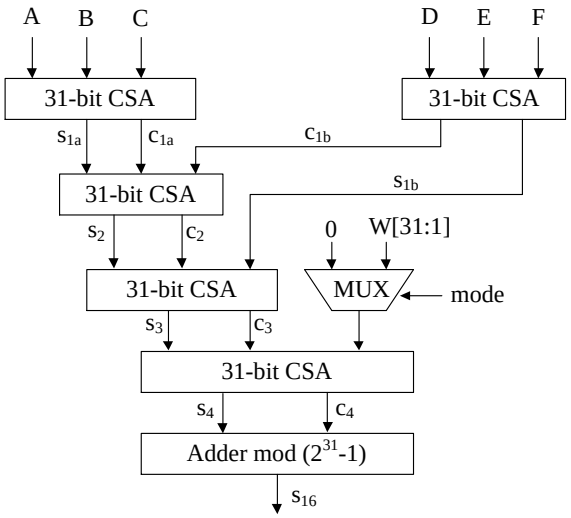

Fig. 5. Hierarchical Carry Save Adder tree

Design of RSA

The most popular public key algorithm is RSA invented by Rivest, Shamir, and Adleman [1]. For high security reason, the key length of the RSA algorithm is 1024 bits or greater [7]. The main operation of the RSA algorithm is the modular exponentiation. The modular exponentiation is performed by a series of the modular multiplications. The Montgomery multiplication (MP) on the large integer number is the efficient method to perform the modular multiplication. There are two methods to compute the modular exponentiation: right-to-left (R-L) method and left-to-right (L-R) method. The R-L method is faster than L-R method because the multiplication and squaring can be performed in parallel. However, the price paid for hardware resource is higher. In this paper, we compute the modular exponentiation by using L-R method and the Montgomery multiplication.

Algorithm 1 implements the Montgomery multiplication. The addition of long operands in the loop is performed by 3-to-2 carry save adder (CSA). To get the final result, we need to add the carry output and the sum output of CSA. In this paper, we use 32-bit RCA and a shift register to implement this final addition because of its simplicity and area saving. It takes (k+3+k/32) clock cycles to complete the Montgomery multiplication, where k is the size of the operands; k/32 is the number of clock cycle to complete the final addition. Fig. 6 shows the CSA-based Montgomery multiplier.

Algorithm 2 implements the modular exponentiation by using the Montgomery multiplier. In this algorithm, C is the operand that has the length of 1024 bits; d_i is the exponent with the length of 1024 bits.The block diagram of the modular exponentiation is shown in Fig. 7. This architecture uses only one Montgomery multiplier. Two multiplexers are used to select the inputs for the Montgomery multiplier. Based on the input value d_i, the control block determines the values of sel_1 and sel_2.

Algorithm 1 – Montgomery multiplication by using CSA

```
//Inputs: x, y, n
ps = 0, pc = 0, ss = 0, sc = 0;
for (i = 0; i <= k+1; i= i + 1){
      (sc, ss) = (ps + pc + x(i) * y);
      (pc, ps) = (ss + sc + ss(0)*n)/2;
}
return (ps + pc)
//Output: p = xyr^-1 mod n with r = 2^(k+2)
```

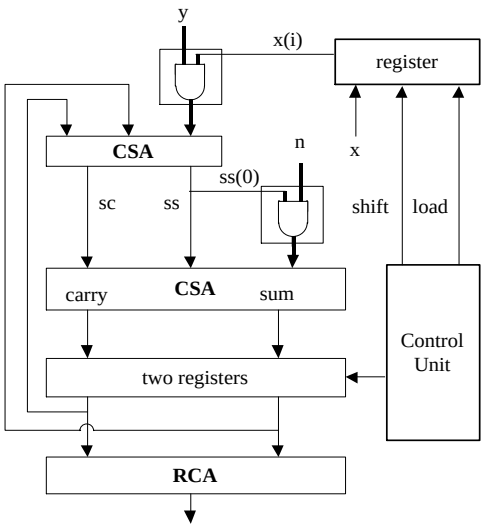

Fig. 6. Montgomery multiplier

Algorithm 2 – Modular exponentiation, L-R method

```
a = C.r mod n;
b = 1.r mod n;
for (x = b, i = k – 1; i >= 0; i = i + 1) {
    x = MP (x, x);
    if (d_i == 1) x = MP (x, a);
}
x = MP (x, 1);
return x;
//Output: x = C^d
```

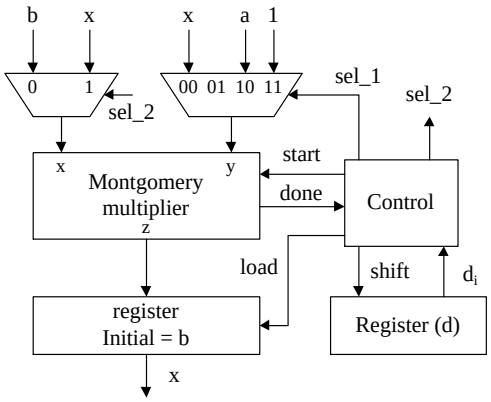

Fig. 7. Modular exponentiation using MP

RESULTS AND DICUSSION

Experimental results of ZUC and RSA

The ZUC implementation is passed all test sets that were provided by ZUC Implementor's Test Data [7]. All the stages of the ZUC core have been implemented in hardware. To make the fair comparison, the implementation is synthesized with Quartus II (Altera) and ISE (Xilinx) as well. In [5], they implemented a pipeline architecture that achieves the maximum operating frequency of 222 MHz. However, it costs higher hardware resources, higher latency (4 extra clock cycles), and initialization stage was implemented in software to reduce hardware resources. In [6], their proposal used ripple carry adders in series, which limits the operating frequency of the circuit. Our proposal uses hierarchical CSA tree, and RCAs, which achieves throughput up to 4.45 Gbps in Virtex 5, and 4.0 Gbps in FPGA Stratix IV EP4SGX230KF40C2.

Table 1. The comparison of the two architectures

Architecture	Technology	Slices/ALUTs	Frequency (MHz)	Bit rate (Gbps)
Our proposal	EP4SGX230KF40C2	1166 ALUTs	125	4.0
Our proposal	XC5VLX50-3FF324	384 slices	139	4.45
ZUC [5]	XC5VLX110T	575 slices	222	7.1
ZUC [6]	XC5VLX50-3FF324	385 slices	65	2.08

In the RSA implementation, we use 3-to-2 CSA and 32-bit RCA to implement the Montgomery multiplier, which is technology independent. It takes $2(k+3+k/32)*k_d$ clock cycles to complete the modular exponentiation, where k is the bit length of the modulus, k/32 is the number of clock cycles cost to complete the final addition (sum and carry) in the Montgomery multiplication and k_d is the bit length of the key. Compared with systolic architecture [3], our implementation has a higher operating frequency. The architecture in [9] used 4-to-2 CSA to implement the Montgomery multiplication. However, this costs some extra registers to store intermediate results of CSA.

Table 2. The comparison of the two implementations

Architecture	Technology	LEs	Fmax (MHz)	Number of clock cycles
Our proposal	EP4SGX230KF40C2	16964	214.10	$(k+3+k/32)(2k_d+1)$
Our proposal	EP1S40F780C5	16969	145.07	$(k+3+k/32)(2k_d+1)$
[3]	EP1S40F780C5	12881, 5120 RAM bits	100.25	-
[9]	XC2V6000	22075 Slices	93.34	$2(k+2)(k_d+3)$

Experimental results of the proposed cryptosystem

The design is synthesized with Quartus II tool based on Stratix IV FPGA EP4SGX230KF40C2. The results show that our proposed system allows the secret key to be changed. At the operating frequency of 125 MHz, the total processing bit rate is 4.0 Gbps that satisfies the required bandwidth in the video streaming application. Fig. 8 and Fig. 9 show the decryption process for video content. The original video content is recovered by XORing the generated keystream and the encrypted video. Fig. 9 shows the new secret key applied when the signaling value of 0x2.

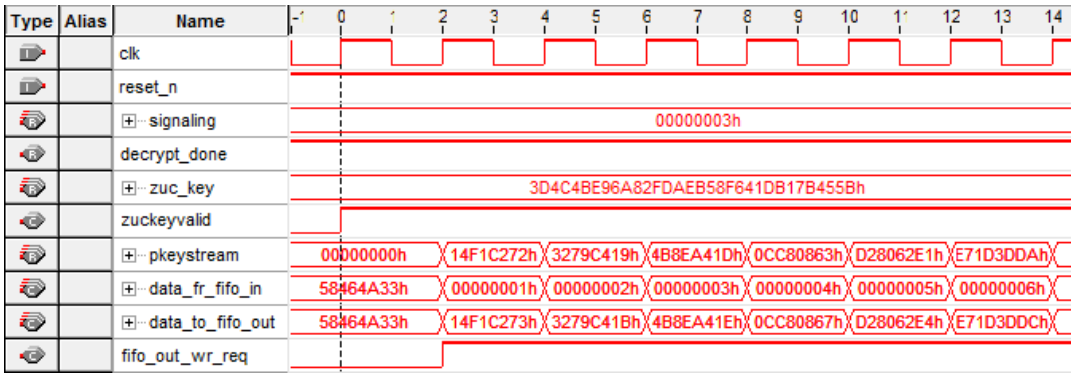

Fig. 8 The result captured by SignalTap Logic Analyzer (using the first key)

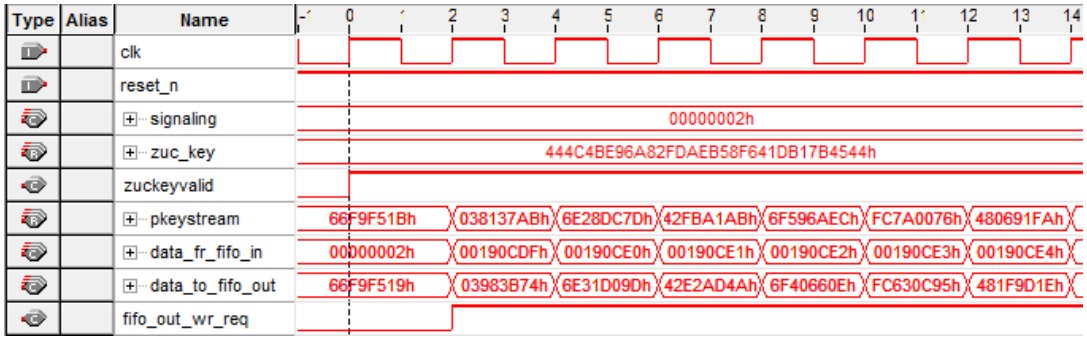

Fig. 9 The result captured by SignalTap Logic Analyzer (using the second key)

To test the operation of our cryptosystem, we integrated H.264 decoder into our system (Fig. 1) to decode the video content. Fig. 10 shows the video content in memory captured by In-system Memory Content Editor tool that is integrated into the Quartus II tool. Fig. 11 shows one video frame that is displayed on the display device.

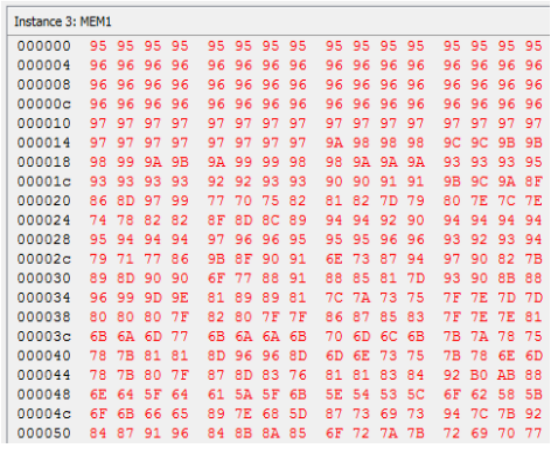

Fig. 10 Video captured by In-system Memory Editor

Fig. 11 Video content displayed on the display device

CONCLUSION

The high performance cryptosystem is presented in this paper that has been implemented and prototyped on FPGA Stratix IV EP4SGX230KF40C2. The experimental results show that the key exchange does not need to be performed on a dedicated channel as in the traditional cryptosystem. In addition, key changing can be performed during one session, which maximizes the security of this cryptosystem. The decryption bit rate of this architecture is up to 4.0 Gbps at the operating frequency of 125 MHz, which is high enough for the real-time application such as video streaming. In this implementation, we focus not only on improving the operating frequency, but also optimizing the hardware resources.

Acknowledgement: *The authors would like to thank to CESLab for technical support and for providing us with FPGA evaluation board. The Department of Science and Technology of Ho Chi Minh City has funded this research.*

Thiết kế phần cứng hệ thống mật mã có hiệu năng cao cho ứng dụng truyền video

- **Nguyễn Văn Toàn**
- **Đỗ Quốc Minh Đăng**
- **Nguyễn Đức Phúc**
- **Nguyễn Đình Thúc**
- **Huỳnh Hữu Thuận**

Trường Đại học Khoa học Tự Nhiên, ĐHQG-HCM

TÓM TẮT

Bài báo này trình bày về thiết kế phần cứng hệ thống mật mã có hiệu năng cao dành cho ứng dụng truyền video. Hệ thống chúng tôi đề nghị là hệ thống kết hợp hai thuật toán mã hóa đối xứng và mã hóa công khai nhằm tận dụng các ưu điểm của chúng. Thuật toán mã hóa đối xứng ZUC được sử dụng để mã hóa/giải mã video, trong khi đó thuật toán mã hóa công khai RSA thực hiện mã hóa/giải mã khóa bí mật. Kiến trúc này đạt được hiệu năng cao như: độ bảo mật cao và tốc độ xử lí (mã hóa/giải mã) cao. Hệ thống đạt được độ bảo mật cao nhờ sự trao đổi khóa bí mật dễ dàng của hệ mật mã công khai. Nhờ tốc độ mã hóa/giải mã cao của thuật toán mã hóa khóa đối xứng mà tốc độ mã hóa/giải mã của hệ thống đạt được là rất cao. Bộ giải mã video H.264 cũng được tích hợp vào hệ thống để kiểm thử chức năng của hệ thống mật mã. Hệ thống này được thực hiện phần cứng bằng ngôn ngữ đặc tả phần cứng Verilog-HDL, sau đó được mô phỏng bằng bộ mô phỏng ModelSim, và được kiểm tra, đánh giá trên bộ Kit của Altera dùng FPGA Stratix IV. Tốc độ giải mã mà hệ thống đạt được lên đến 4.0 Gbps tại tần số hoạt động là 125 MHz, thỏa mãn các ứng dụng truyền video.

Keywords: *hệ thống mật mã, mã hóa, giải mã, RSA, ZUC, FPGA.*

REFERENCES

[1]. A. Menezes, P. Oorschot, S. Vanstone, Handbook of applied cryptography, CRC Press (1997).

[2]. A.A. Gutub, F.A. Khan, Hybrid crypto hardware utilizing symmetric-key & public-key cryptosystems, International Conference on Advanced Computer Science Applications and Technologies, IEEE (2012).

[3]. M.K. Hani, H.Y. Wen, A. Paniandi, Design and Implementation of a private and public key crypto processor for next-generation its security applications, *Malaysian Journal of Computer Science*, 19, 1, 29-45 (2006).

[4]. ETSI/SAGE Specification. Specification of the 3GPP confidentiality and integrity algorithms 128-EEA3 & 128-EIA3. Document 2: ZUC Specification; Version: 1.6; Date: 28th June 2011.

[5]. L. Wang, et al, Evaluating optimized implementations of stream cipher ZUC algorithm on FPGA, *Springer*, 202-215 (2011).

[6]. P. Kitsos, N. Sklavos, A.N. Skodras, An FPGA implementation of the ZUC stream cipher, 14th Euromicro Conference on Digital System Design, IEEE (2011).

[7]. C. McIvor, M. McLoone, J.V. McCanny, Fast Montgomery modular multiplication and RSA cryptographic processor architectures, *Conference Record of the thirty-seventh Asilomar Conference*, 379-384 (2003).

[8]. ETSI/SAGE Specification. Specification of the 3GPP confidentiality and integrity algorithms 128-EEA3 & 128-EIA3. Document 3: Implementor's Test Data; Version: 1.1; Date: 4th Jan 2011.

[9]. N. Wen, Z.B. Dai, Y.F. Zhang, FPGA Implementation of alterable parameters RSA public-key cryptographic Co-processor, *IEEE* (2005).