*International Journal of Advanced Computer Research (ISSN (print): 2249-7277 ISSN (online): 2277-7970)*
*Volume-3 Number-4 Issue-13 December-2013*

# Hybrid Cryptographic Processor for Secure Communication Using FPGA

**Savitha Raj.S[1], Merlin Sharmila.A[2], Poorinima Beneta.P[3]**

## Abstract

***Cryptographic hash functions are mainly used for the purpose of authentication and for integrity of messages. In this paper, we investigate high-speed, efficient hardware algorithm which is a combination of both RSA and BLAKE for providing privacy and security in data networks including encryption/decryption. Hash function- BLAKE is a new standard candidate algorithm; it is one of the finalists in the SHA-3 competition by NIST. RSA is the asymmetric public key cryptography system. Since this is a hybrid algorithm it provides the advantages of both the asymmetric and symmetric key. The choice of the algorithm is based on the absence of backdoor in both these algorithms. The coding of the RSA and BLAKE is done in VHDL and the FPGA synthesis is done using Modelsim software. The results shows that the proposed algorithm is more secure than the simple RSA hardware implemented using the traditional multiplication algorithm.***

## Keywords

***Blake, Hash Function, RSA, Security***

## 1. Introduction

In the recent era the communication has revolutionized and shrunk our entire world. Communication carried over needs to be more secure to avoid fraudulent activities, such as impersonation. For example in order to cinch secure communication documents such as transcripts can be digitally signed; images created by a camera can be digitally watermarked. Cryptographic hash functions are primitives or building blocks utilized in the schemes that are used to provide information security.

**Savitha Raj. S,**Assistant Professor ,Department of Electronics and Communication Engineering, Mahendra College of Engineering, Salem, India.

**Merlin Sharmila.A,** Assistant Professor ,Department of Electronics and Communication Engineering, Mahendra College of Engineering, Salem, India.

**Poorinima Beneta. P,** Assistant Professor, Department of Electronics and Communication Engineering, Mahendra College of Engineering, Salem, India.

These schemes, such as digital signatures and digital watermarking, utilize a number of cryptographic primitives. The cryptographic hash function plays a critical role to provide information security. Security and speed of the hash function has a significant impact on overall security and computational efficiency.

A cryptographic hash function converts an arbitrary length of input data to a fixed-length output. Cryptographic hash functions are somewhat different from ordinary hash functions used in computer programs; however, for simplicity cryptographic hash functions will simply be referred to as hash functions throughout the rest of this thesis. The cryptographic hash function to ensure its security must have certain properties such as: pre-image resistance, second pre-image resistance, and collision resistance. The properties emanate from the ways in which hash functions have been attacked. Pre-image resistance property is related to the concept of one-way function. That is, it should be infeasible for an attacker to determine the original data (or message) from a given hash code or digest (the digest is another name for the hash code or hash value). Second pre-image resistance states that even the slightest change in a message will change the digest value[11]. That is, if an attacker is given a message, it should be infeasible for the attacker to manipulate the message and still obtain the same digest as the original message digest. Collision resistance gives the general analogy of fingerprint with respect to the message digests. That is, every message is expected to have a unique hash code and it should difficult for an attacker to find two messages with the same hash code.

Mathematically, a hash function (H) is defined as follows:

$H: \{0, 1\}^* \rightarrow \{0, 1\}^n$

In this notation, $\{0, 1\}^*$ refers to the set of binary elements of any length including the empty string while $\{0, 1\}^n$ refresh to the set of binary elements of length n. Thus, the hash function maps a input-arbitrary length of binary elements to the output-fixed length of binary elements. The research scene of hash functions has seen a surge of works since attacks [1]–[3] on the two most deployed hash functions, MD5 and SHA-1.

A notable milestone was the forgery of a MD5-signedcertificate using a cluster of PlayStation 3s [4].Such results have led to a lack of confidence in the current U.S.(and de facto worldwide) hash standard, SHA-2 [5], due to its similarity with MD5 and SHA-1. As a response to the potential risks of using SHA-2, the U.S. Institute of Standards and Technology (NIST) has started a public competition—the *NIST Hash Competition*—to develop the future hash standard SHA-3[6].Today, various cryptographic algorithms have been developed and it is broadly classified as symmetric key (DES, TDES, Blowfish, CAST, IDEA, RC4, RC6, AES) and asymmetric key (RSA, ECC) algorithms. In symmetric-key encryption technique, a single key is used for both encryption and decryption process. These algorithms are efficient, are secure, execute at high speeds, and consume less computer resources of memory and processor time. Key distribution problem, Key management problem and inability to digitally sign a message are the major disadvantages of symmetric key technique [12].

Asymmetric key technique solved the problems of symmetric key technique. In case of asymmetric key, instead of a single key, every person has a pair of keys. One key, which is known to everyone, is called the public key and the other which is known only to the owner is called as the private key [8]. There is a mathematical relationship between the public and private key. Thus, if any message 'm' is encrypted using any of the key, it can be decrypted by the other portion. Various asymmetric encryption algorithms (RSA, Elgamal) have been implemented. Details on the working of asymmetric encryption techniques can be had from Schneier [7], Stallings[8].Asymmetric encryption algorithms are broadly divided into three families:

- Algorithms based on the integer factorization problem ( RSA)
- Algorithms based on the discrete logarithm problem(e.g. DLP)
- Algorithms based on Elliptic Curves.

Data security plays a crucial and critical role in modern times for businesses transactions such as e-commerce and m-commerce applications. To address these security concerns, various security protocols that are of symmetric-key and asymmetric-key type have been developed. In this paper, we present a software implementation of a hybrid algorithm that combines both the symmetric key algorithm of BLAKE and the asymmetric- key algorithm of RSA the paper also presents future trends of research. In this paper, we present a hybrid algorithm for enhanced secure communication.

## 2. Algorithm Specification

### 2.1 Blake

BLAKE has two main versions:BLAKE-32 and BLAKE-64. A brief specification of these algorithms is given in this section.TheBLAKE-32 algorithm operates on 32-bitwords and returns a 256-bit hash value. It is based on the iteration of a *compression function*, described in the following.

1) *Compression Function*: Henceforth we shall use the following notations: if is a message (a bit string), $m^i$ denotes its 16-word block, and $m_j^i$ is the jth word of the th block of Indices start from zero, for example a block message is decomposed. We use similar notations for other bit strings. The compression function of BLAKE-32 takes as input the following four values:

- a chaining value h= $h_0, h_1 \ldots h_7$;
- a message block m= $m_0, m_1, .. m_{15}$;
- a salt s= $s_0, s_1 .. s_3$ ;
- a counter t= $t_0, t_1$ .

These inputs represent 30 words in total (i.e., 960 bits). The salt is an optional input for special applications, such as randomized hashing.The output of the compression function is a new chaining of words,$h' = h'_0, h'_1 .. h'_7$.

*2) Initialization*: A 16-word internal state is$v_0, .. v_{15}$ initialized such that different inputs produce different initial states. This state is represented as a 4 x 4 matrix

$$\begin{pmatrix} v_0 & v_1 & v_2 & v_3 \\ v_4 & v_5 & v_6 & v_7 \\ v_8 & v_9 & v_{10} & v_{11} \\ v_{12} & v_{13} & v_{14} & v_{15} \end{pmatrix}$$

Round Function: Once the state is initialized, the compression function iterates a series of ten rounds. A round is a transformation of the state that computes

$G_0(v_0, v_4, v_8, v_{12}) G_1(v_1, v_5, v_9, v_{13})$
$G_2(v_2, v_6, v_{10}, v_{14}) G_3(v_3, v_7, v_{11}, v_{15})$
And then
$G_4(v_0, v_5, v_{10}, v_{15}) G_5(v_1, v_6, v_{11}, v_{12})$
$G_6(v_2, v_7, v_8, v_{13}) G_7(v_3, v_4, v_9, v_{14})$

3) *Finalization:* After the sequence of rounds, the new chaining value is extracted from the state

$v_0, \ldots v_{15}$ with input of the initial chaining value h and the salt s.

$$h'_0 = h_0 \oplus v_0 \oplus s_0 \oplus v_8$$
$$h'_1 = h_1 \oplus v_1 \oplus s_1 \oplus v_9$$
$$h'_2 = h_2 \oplus v_2 \oplus s_2 \oplus v_{10}$$
$$h'_3 = h_3 \oplus v_3 \oplus s_3 \oplus v_{11}$$
$$h'_4 = h_4 \oplus v_4 \oplus s_4 \oplus v_{12}$$
$$h'_5 = h_5 \oplus v_5 \oplus s_5 \oplus v_{13}$$
$$h'_6 = h_6 \oplus v_6 \oplus s_6 \oplus v_{14}$$
$$h'_7 = h_7 \oplus v_7 \oplus s_7 \oplus v_{15}$$

4) *Hashing a Message:* When hashing a message, the function starts from an initial value (IV ), and the iterated hash process computes intermediate hash values that are called chaining values. Before being processed, a message is first padded so that its length is a multiple of the block size (512 bits). It is used to avoid certain generic attacks on the iterated hash . The salt is chosen by the user, and set to zero by default.

### 2.2 RSA Algorithm

RSA algorithm (named after its founders, Ron Rivest, Adi Shamir, and Leonard Adleman) has become almost synonymous with public key cryptography. An interesting feature of RSA algorithm is that, it allows most of the components used in encryption process are re-used in the decryption process. So this can minimize the resulting hardware area.

RSA involves a public key and a private key. The public key can be known to everyone and is used for encrypting messages. Messages encrypted with the public key can only be decrypted using the private key. The keys for the RSA algorithm are generated the following way:

- Choose two distinct prime numbers *p* and *q*. For security purposes,p and q value should be choosed randomly. .
- Prime integers can be efficiently found using a primality test.
- Compute $n = pq$,*n* is used as the modulus for both the public and private keys
- Compute $\varphi(n) = (p-1)(q-1)$, where φ is Euler's totient function.
- Choose an integer *e* such that $1<e<\varphi(n)$ and $\gcd(e,\varphi(n)) = 1$, i.e. *e* and φ(*n*) are co prime.*e* is released as the public key exponent having a short bit-length and small Hamming weight results in more efficient encryption - most commonly 0x10001 = 65537. However, small values of *e* (such as 3) have been shown to be less secure in some settings.
- Determine $d = e^{-1} \bmod \varphi(n)$;i.e. *d* is the multiplicative inverse of *e* mod φ(*n*).

**Table-1: Example for RSA Method Public and Private Key Pair**

| p prime | q prime | n=p*q | m | e | calc.'d' | Private-ate (n,d) | Public (n,e) |
|---|---|---|---|---|---|---|---|
| 5 | 3 | 15 | 8 | 11 | 3 | 15,3 | 15,11 |
| 7 | 5 | 35 | 24 | 11 | 11 | 35,11 | 35,11 |
| 13 | 17 | 221 | 192 | 11 | 35 | 221,35 | 221,11 |

Extended Euclidean algorithm is often used for this type of computation where *d* is kept as the secret private key exponent (decryption). The public key consists of the modulus *n value* and the public (or encryption) exponent *e*. Due to the property of symmetry in modular arithmetic RSA encryption and decryption are mutual inverse and commutative as shown in equation (1) and (2).

## 3. Proposed Architecture

The overall architecture of the proposed system shown in Fig-1 consists of a random logic unit, an encryption unit, a decryption unit, a memory, a lookup table and multiplexers. The project concentrates on providing high data security to the data's stored in the memory which is done by means of two cryptographic algorithms i.e.RSA, BLAKE. Two operations i.e. either read/write to the memory are performed. When the data is to be written content goes here into the memory, the address of the memory, read/write enable and data to be written to the memory is given at the input. The input 64 bit data is divided into 4 blocks of 16 bit each.

The input at random logic selection is selecting the algorithm to be used for each block of data. Enable signal is generated and it is given to the data multiplexer, key multiplexer and the look up table respectively to perform the appropriate operation. Enable signal control the encryption to be used and the operation of the multiplexer. Based on the input the algorithm is chosen and the given data is encrypted according to it and is stored in the memory address specified via a multiplexer. At the same time in addition to it a private key is generated and is stored in the look up table via a multiplexer. The multiplexers used, select the proper encrypted data

and private key from the encryption unit and stores in the memory unit and look up table respectively also used to gets back the key for decryption of the data to the decryption unit. The encrypted data is stored in the given memory location by means of memory unit. While read operation is executed, the lookup table provides the private key which is used to recover the original data from the encrypted data from the stored address, also algorithm used to encrypt the data. From the memory & look up table, the decryption unit receives the encrypted data, appropriate private key and algorithm used for encryption. The multiplexer 3 gives the original data that obtained either of the two algorithms used in project. Thus the original data is secured by the proposed cryptographic processor and can't be hacked by any malware attack.

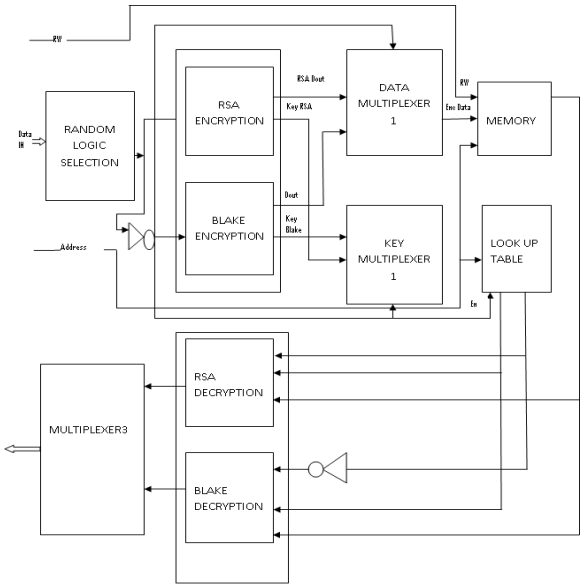


**Fig-1: Hybrid Processor using RSA and BLAKE**

### 3.1. Memory Architecture

The VLSI implementation needs memory to store 16 words of internal state and eight words of chaining value, plus additional registers to store the salt (four words), the counter (two words), and the message block (16 words), i.e., in total as 1472 bits of memory. The counter is used during four clock cycles and thus to be stored. In terms of area and energy consumption the memory unit is the main contribution compared to the minimum circuit needed to implement the compression function (initialization, rounds, and finalization). The primary interest is to design special-purpose register elements, in order to decrease the global resource requirements of the hash core. Here we introduce the compact architecture of BLAKE-32 and RSA semi-custom memories based on clock-gated latch arrays, which is able to store at most one new word per cycle. In general, depending on the word number of the target value to be stored, these memories replace the standard flip-flop cells by latch cells. The latches are organized in 32-bit banks, in which each bank stores a single word and is triggered by a dedicated gated clock. Our example architecture depicts a four-word latch array that is used to store the salt value. In the address decoder, the different one-hot enable signals are activated, depending on the write address. An input flip-flop bank is added in order to prevent timing loops inside the logic, caused by the transparent behavior of latches. The outputs of the flip-flops are connected to the inputs of all latch banks and this bank is in turn driven by a gated clock generated with the write enable signal. During write enable, the input word is first stored inside the flip-flop bank and subsequently passed to the activated latch bank. Memory access times for write and read operations have been further analyzed. Due to the flip-flop bank at the input, the write time is kept similar to a standard flip-flop-based memory. At the output, the read time is only affected by the size of the multiplexer, i.e., the number of words and the timing is equivalent in both architectures. The use of a different memory design to reduce area and power dissipation could potentially generate additional involuntary multiple signal transitions [12].

### 3.2 Measurements and Performance Comparision

All our implementations are designed in VHDL language and synthesized by using Modelsim Software. Hardware efficiency is defined as throughput per unit area (gates, slices). From our experiment, the delay of the critical path in the 32-bit is reduced 77.8% from the straightforward implementation of 512-bit data path. Compared with the compact BLAKE implementations in [10], [11], our design is one times and 40 time more hardware efficient. Note that in [9,10] the area does not count in the register m, s, and h. If our 32-bit design does not include the registers, a minimum area of 8,678 gates is achieved. When compared with other compact implementation of SHA-3 candidates, BLAKE is also three time and 36 times more efficient than Gostl and Skein-256 in [11]. For fair comparison, Gostl might still not achieve comparable hardware efficiency by using the more advanced $0.18 f\hat{E}m$ technology. Note that in [13], the much

higher throughput and hardware efficiency are achieved for BLAKE in that it is a full 512-bit datapath implementation. Compared with BLAKE, the standard hash function SHA-256 in [14] is three time more hardware efficient. One reason is that in SHA-256, one round only requires one clock cycle, whereas in BLAKE 16 clock cycles are required.The power consumption values are achieved by a simulation based evaluation. When comparing BLAKE with the compact implementation of the Whirlpool hash function, we find the performance parameters of the latency, frequency, and hardware efficiency are comparable. As for the ASIC implementation, the compact FPGA implementation of SHA-256 also shows greater performance advantages over BLAKE. The compact hardware implementations of BLAKE per-form better than LAKE both on ASIC and FPGA platforms. Furthermore, our compact design of LAKE shows better or comparable hardware efficiency than that in [13], which is a 512-bit full path implementation.

## 4. Results and Discussion

The proposed design has been implemented by VHDL, simulated with ModelSim 6.2b and the Hybrid implementation of the algorithm in FPGA is done .The first step in the RSA implementation is the prime number generation and the data is then encrypted to a different form. And then for this message a key value is generated. And the original message is then decrypted and obtained.

The BLAKE algorithm creates a key value and the cipher text value for the original message to be encrypted and the initialization and the box values are generated for it.

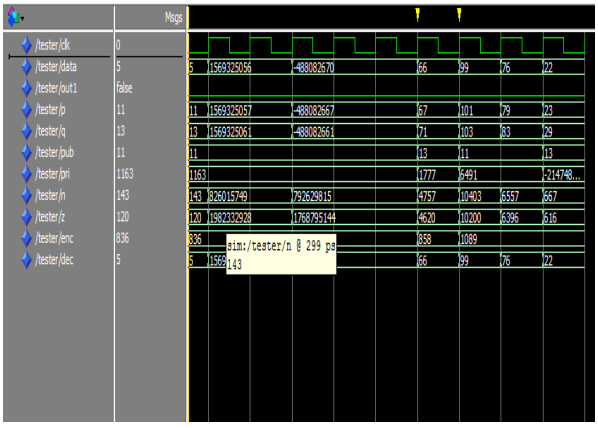


**Fig-2: Encryption/Decryption for RSA**

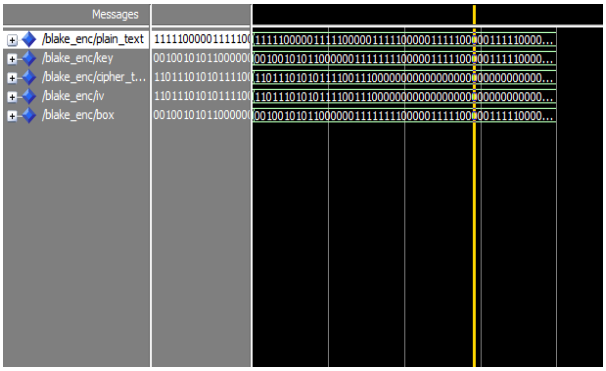


**Fig-3: Encryption/Decryption for BLAKE**

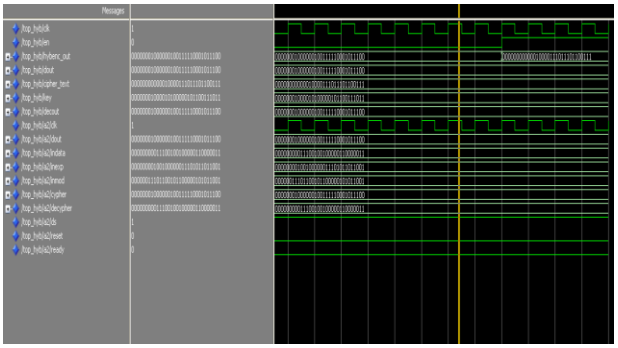

**Fig-4: Hybrid Encryption/Decryption using RSA and BLAKE**

The simulation result shown in Fig- 4 is the process of hybrid encryption and decryption.First if the enable signal is 1 then only RSA encryption takes place.Incase of if the enable signal is 0 then dual encryption takes place,that is first RSA encryption and next the encrypted value is again encrypted using BLAKE algorithm.Reverse process is carried out in case of decryption.By means of this process dual encrytion process the security is highly enhanced.

## 5. Conclusion

In this paper, we evaluate compact software implementation of BLAKE and RSA hash functions. From the performance analysis of the proposed design, it is evident that the speed of the Blake hash function can be improved by the proposed design. The VHDL code for RSA and BLAKE Encryption/Decryption algorithm is developed block wise. Optimized and Synthesizable VHDL code for each block synthesized using Modelsim and the VHDL implementation has shown that the language provides a useful tool of practicing the algorithms without drawings of large amounts of logic gates.

## References

[1] X. Wang and H. Yu, "How to break MD5 and other hash functions," in Advances in Cryptology—EUROCRYPT 2005, ser. Lecture Notes in Computer Science. Berlin, Germany: Springer, 2005, vol. 3494, pp.19–35.

[2] C. D. Cannière and C. Rechberger, "Finding SHA-1 characteristics: General results and applications," in Advances in Cryptology—ASIACRYPT 2006, ser. Lecture Notes in Computer Science. Berlin, Germany: Springer, 2006, vol. 4284, pp. 1–20.

[3] M. Stevens, A. Lenstra, and B. de Weger, "Chosen-prefix collisions for MD5 and colliding X.509 certificates for different identities," in Advances in Cryptology—EUROCRYPT 2007, ser. Lecture Notes in Computer Science. Berlin, Germany: Springer, 2007, vol. 4515, pp.1–22.

[4] A. Sotirov, M. Stevens, J. Appelbaum, A. Lenstra, D. Molnar, D. A.OsvikB, and B. de Weger, "MD5 considered harmful today. Creating a rogue CA certificate," presented at the 25th Chaos Commun. Congr.,Berlin, Germany, 2008.

[5] NIST, Gaithersburg, MD, "Announcing the secure hash standard," FIPS 180-2, 2002.

[6] NIST, Gaithersburg, MD, "Call for a new cryptographic hash algorithm (SHA-3) family, federal register," 2007. [Online]. Available:http://www.nist.gov/hash-competition.

[7] SchneierB. Applied Cryptography. John Wiley & Sons Inc., New York, New York, USA, 2nd edition, 1996.

[8] Stallings W. Cryptography and Network Security. Prentice Hall, Upper Saddle River, New Jersey, USA, second edition, 1999.

[9] J.-P. Aumasson, L. Henzen, W. Meier, and R. C.-W Phan, "SHA-3 Proposal BLAKE, submission to NIST," 2008. [Online]. Available:http://131002.net/blake/.

[10] NIST, Gaithersburg, MD, "SP 800-106, randomized hashing digital signatures," 2007.

[11] J. Kelsey and B. Schneier, "Second preimages on n-bit hash functions for much less than $2^n$ work," in EUROCRYPT, ser. Lecture Notes in Computer Science, R. Cramer, Ed. New York: Springer, 2005, vol.3494, pp. 474–490.

[12] Feldhofer.M, Wolkerstorfer.J and Rijmen. V, (2005)' AES implementation on a grain of sand',in Proc. IEEE Inf .Security.

[13] Feldhofer.M and Wolkerstorfer.J,(2007) 'Strong crypto for RFID tags-A comparision of low power hardware implementations', in Proc.IEEEInt.Symp.Circuits Syst.(ISCAS).

[14] Henzen.L, Carbognani.F, Felber.N and Fichtner.W,(2008)'VLSI hardware evaluation of the stream ciphers Salsa20 and ChaCha, and the compression function Rumba',in Proc. IEEE Int.Conf.Signals,Circuits Syst.(SCS).

**S. Savitha Raj** currently working as Assistant Professor in Mahendra College of Engineering-Salem. She completed her M.E. in VLSI Design from Sree Sastha Institute of Engineering & Technology, Chennai. She has received her Bachelor's Degree in Electronics & Communication Engineering from Idhaya Engineering College for Women in 2010 from Anna University, Chennai. Her areas of interests include VLSI Design, Mobile communication, Cryptography and Network security, Image Processing.

**A. Merlin Sharmila** currently working as Assistant Professor in Mahendra College of Engineering-Salem. She completed her M.E. in Communication systems at Hindustan University, Chennai. She has received her Bachelor's Degree in Electronics& Communication Engineering from Idhaya Engineering College for Women in 2011 from Anna University, Chennai. Her areas of interests include Communication Networks, video Signal Processing.

**P. Poorinima Beneta** is currently working as Assistant Professor in Mahendra College of Engineering-Salem. She completed her M.E. in Communication systems at Hindustan University, Chennai. She has received her Bachelor's Degree in Electronics & Communication Engineering from Idhaya Engineering College for Women in 2011 from Anna University, Chennai. Her areas of interests include Mobile Communication Wireless Sensor Networks, Signal Processing.