American Journal of Engineering and Applied Sciences

Original Research Paper

The Effect of RSA Exponential Key Growth on the Multi-Core Computational Resource

[1]Mohamad A. Mohamed, [2, 3]Ammar Y. Tuama,
[1]Mokhairi Makhtar, [1]Mohd K. Awang and [1]Mustafa Mamat

[1]Faculty of Informatics and Computing, Universiti Sultan Zainal Abidin, Besut, Malaysia
[2]Faculty of Science Computer and Information Technology, Universiti Putra Malaysia, Serdang, Malaysia
[3]Ministry of Higher Education and Scientific Research, Baghdad, Iraq

Article history
Received: 21-09-2016
Revised: 06-10-2016
Accepted: 24-11-2016

Corresponding Author:
Mohamad A. Mohamed
Faculty of Informatics and Computing, Universiti Sultan Zainal Abidin,Besut, 22200 Malaysia
Email:mafendee@unisza.edu.my

Abstract: Cryptography has been widely used as a mean to secure message communication. A cryptosystem is made up of a publicly available algorithm and a secretly kept key. The algorithm is responsible for transforming the original message into something unintelligible. The result of losing the key or cracked algorithm can be catastrophic, where all secret communications will be known to adversaries. One way to find the key is by brute-force attacks which try every possible combination of keys. The only way to prevent this is by having the key of sufficiently large enough such that finding the right key cannot be made in a reasonable time frame. However, large key size imposes extra computational works which result in larger energy consumption and thus more heat dissipation to the environment. Therefore, the selection of key size does not only depends on the required security level, but also factors such as the ability of the processor and the available memory resources. The advent of multi-core technology promises some improvements in the utilization of computational resources. Many reports support the idea that multi-core technology brought a significant improvement over the single core technology. In this study, we investigate this hypothesis on the RSA cryptosystem in relation to the key size. Earlier studies reported multi-core efficiency in normal applications, but the question arises if multi-core architecture remains superior to a single core architecture when dealing with applications involving large integers. From our experimentation, we observe that the higher the number of cores, the better the performance of the encryption and decryption processes. The quad-core technology can smoothly handle operations involving 8192 bits key.

Keywords: Cryptography, Multi-Core, Large Integer, CPU Resource, Memory Consumption

Introduction

In this challenging era, security is the main concerns and crucial for any communication system, especially for those residing within an insecure network environment. One way to secure data communication is via cryptography. Cryptographyis defined as an art and a science of hiding original data from unauthorized access. Cryptography provides security services such as confidentiality, integrity, authentication and non-repudiation, depending on the requirements. Cryptography has been widely used in various applications that require a certain level of security, such as web browsing and social networking and healthcare applications (Corchado *et al*., 2008). As an example, Skype used 2048 bits RSA public key and 256 bits AES symmetric encryption (Baset and Schulzrinne, 2004), although their security is still questionable.

A cryptographic system (cryptosystem) is the combination of encryption/decryption algorithm, key and key management functions used to perform cryptographic operations. Cryptographic algorithms can be categorized into the secret-key system and public-key system. The security of any cryptosystem depends solely on the secret key (private key) whereas the algorithm should be made public. A key is a piece of information that determines the functional output of a cryptographic algorithm or cipher. Keys are used for controlling the operation of a cipher so that only the correct key can transform ciphertext back to the plaintext. If we know the keys, then we can simply encrypt and decrypt messages. If the keys are found by the attacker, then the cryptosystem is said to be compromised. The strength of any cryptosystem is determined by the hardness of acquiring the key associated with the system which largely depends on the key size. Indeed, longer the key size requires more execution time. For this, an application that is less sensitive can have lower security but with better performance. On the other side, the art of recovering plaintext from ciphertext or the key used for decryption is known as cryptanalysis. Cryptanalysis is the study of taking encrypted data and trying to unencrypt it without the use of the key. Cryptanalysis is used by hackers with bad intentions, to break codes by finding weaknesses within them. It is also often used by the military. It is also appropriately used by designers of encryption systems to find and study the methods to improve the cryptosystem.

Key size or key length is the size measured in bits of the key used in a cryptographic algorithm. An algorithm's key length is distinct from its cryptographic security, which is a logarithmic measure of the fastest known computational attack on the algorithm, also measured in bits. The security of an algorithm cannot exceed its key length (since any algorithm can be cracked by brute force). A key should, therefore, be large enough that a brute force attack (possible against any encryption algorithm) is infeasible (Barenghi *et al*., 2010). Each algorithm has a different level of cryptographic complexity. Therefore, it is usual to have different key sizes for the same level of security, depending upon the algorithm used. According to RSA Security, the security available with a 1024-bit key using asymmetric RSA is considered approximately equal in security to an 80-bit key in a symmetric algorithm (Bi *et al*., 2016; Miri, 2013). Any encryption algorithm depends on the length of the key and the computational effort required breaking the key (Al-Hamami and Al-Kubaysee, 2011). The new, faster and better methods for factorizing numbers are constantly being devised to attack RSA cryptosystem that relies on prime numbers.

In this study, we investigate the relationship between the key size of the RSA and the efficiency of the encryption algorithm on multiple cores computer technology over the single-core. RSA is special in that, its operations involve very large numbers. Therefore, knowing the behavior of such operations on multi-core technology can be very important and that can be extended to other cryptographic algorithms.

Furthermore, we study the behavior of multi-core technology when dealing with large integer operations such that found in the RSA cryptographic algorithm. For benchmarking, performance comparison against the single-core is made to measure the effectiveness of multi-core running this type of applications. It is important to know the behavior of such operations on multi-core technology so that we can extend to other cryptographic algorithms.

This paper is organized as the followings. Section 2 discusses the building block of RSA with a focus on the three major operations, key generation, message encryption and decryption. Section 3 discusses the concepts of multi-core technology and how it was designed to improve the performance of the old-fashioned single-core technology. Section 4 describes how the simulation works have taken place in the environment. Section 5 presents the simulation results accompanied by some discussions. Section 6 concludes the studies with possible future investigations.

RSA Algorithm

RSA is the best known and most widely used public-key cryptosystem that was invented by Rivest *et al*. (1978). Similar to ECC (Mohamed *et al*., 2011), El-Gamal and Cramer-Shoup, it can be used both for encryption and digital signatures. Some examples of real-world applications that employ RSA are PGP, OpenSSL and IPSec. Within these applications, RSA has been implemented as one of the optional algorithms for use. The security of RSA is assumed to be equivalent to integer factorization, although this has not been proved. It is based on the difference between the ease of finding large prime numbers and computing modular powers on the one hand and the difficulty of factorizing a product of large prime numbers as well as inverting the modular exponentiation. RSA algorithm is constructed on multiplicative group structure $\mathbb{Z}_N^*$ such that $\mathbb{Z}_N^* = \{x \in \mathbb{Z} \mid \gcd(x, N) = 1\}$ (Coutinho, 1999).

Encryption and decryption processes are based on the classics Euler's Theorem (a generalized version of Fermat's Little Theorem). By this theorem, we can

DOI: 10.3844/ajeassp.2016.1054.1061

guarantee the recoverability of the original messages. The whole process centers around the formula $M^{k\varphi(N)+1} \equiv M(\text{mod } N)$ for some integer k such that $N = p.q$ where $M<N$, for p and q primes and$M$ need not be relatively prime to N.

The idea of RSA secure message communication can be divided into two phases that are: Key exchange (similar to call establishment as in telephone communication) and data exchange (voice exchange). The whole operation of RSA key exchange is shown in Fig. 1. Initially, some random number generator is used to generate two primes p and q where $p \neq q$. The two integers can be tested using a probabilistic polynomial-time Monte-Carlo algorithm like Fermat's Primality, Solovay-Strassen or Miller-Rabin algorithm for its true primality (Rempe-Gillen and Rebecca, 2014). These algorithms are remarkably fast, an integer N can be tested in time that is polynomial in $\log_2 N$, the number of bits in the binary representation of N. If at least one has failed the test, then new integers must be generated and tested. For ensuring maximum security, p and q should be of equal length.

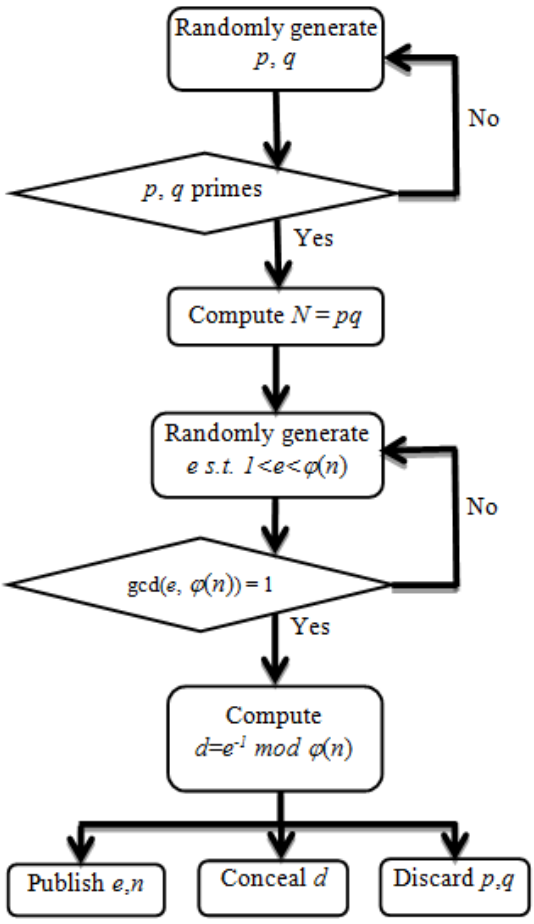

Fig. 1. RSA key generation

RSA computation takes place with integer modulo $N = p*q$, for two large secret primes p, q. The typical value for N is of 1024 bits or 300 decimal digits. For better security, N should be greater than 1024 bits. As an example, the use of 2048 bits N in applications like ZRTP and TLS would ensure the security for decades long. Next, we randomly generate the public key e such that $1<e<\varphi(N)$. Then we compute its inverse d such that $d = e^{-1} (mod\varphi(N))$. The resulting values can be categorized into three sects, one to be given to the public, another to be kept as a secret and the last one to be discarded secretly as shown in Fig. 1.

Let say the sender have a message (a file) to be encrypted and sent, assume he/she has received the public key of the receiver. First, the file needs to be encoded into numerical digits. Blocks of a few digitsare formed so as the binary value is less than that of N. Encryption is performed block by block such that the ciphertext $C = M^e(modN)$ where M is a message block. On the receiver side, each message blocked can be obtained by the formula $M = C^d(modN)$. Having the value of C, d and N, the computation of M is made easy. The original message can be obtained by decoding all the pieces together. The direct operation of multiplying M for e times (C for d times) before applying modulo-N can be time consuming. As a result, plenty of advances were discovered via the use of heuristics and meta-heuristics techniques (Saul *et al*., 2015; Juraj, 2004). From the perspective of mathematical attacks, there are three possible approaches, namely factoring N to find p and q, determining $\varphi(N)$ and reversing d of e, all of which is assumed to be equivalently hard problem. The security depends on the magnitude of N. The larger this integer is, the more difficult factorization will be and therefore increases its security. Until recently, the three most effective algorithms for this job are quadratic sieve, elliptic curve factoring algorithm and number field sieve (Pomerance, 1996).

Multi-Core Technology

Enterprises worldwide has been facing rapid business growth and to keep pace with the requirement for more complex applications, more powerful computer systems are needed. Advancement can be in either hardware or software. Earlier a server is used as a dedicated system to handle these needs and to extend the computational power, a mean for a faster processor speed is realized. However, this doesn't come without consequences; more speed means more operations per second, which could incur a corresponding increase in power consumption and thus emitting more heats. In such an environment, proper monitoring and control are needed to contend the amount of heat generated to avoid the possible

hazardous event. As such, the increment in processor speed is somewhat limited by the heat production.

This has opened up a room for other technology such as multi-processor and later multithreading to come along. Multi-processor technology boosts the performance without increasing the clock speed. In fact, to maintain the same performance as that of a single processor architecture, multiprocessor architecture can run at a lower clock speed. As such, by moving from a single high-speed CPU to multiple low-speed CPU could potentially produce a reduction in heat production. However, for a computer system to support multi-processor, it require more than one CPU sockets on the hardware. There is also an additional latency if the CPUs need to communicate with each other.

Another technique to increase the performance is via harnessing the multithreading technology. Multithreading is purely on the software level, it is an ability of a CPU or a single-core in a multi-core processor to execute multiple processes or threads of a process concurrently, appropriately supported by the operating system. Multithreading has to share the resources such as the computing units, the CPU caches and the translation look aside buffer. Multithreading can be implemented on multi-processor machines to further improve the performance. However, multithreading generally will increase the complexity of the computer programs, of which must be written such a way that there should be no conflict among different threads. This problem is very much related to the synchronization of the shared resources of which could result in potential deadlocks.

In recent years, the technology of multi-core processing (Blake *et al*., 2009; Roy *et al*., 2008; Geer, 2005) has been widely used in many computational fields including general-purpose, embedded, network, digital signal processing. In the hardware, the multi-core processor is a single computing component consisting of two or more independent processing units that each of it can perform different operations at the same time. This multi-core structure comes with cache modules that are shared between cores. This helps dramatically improve performance while keeping the physical CPU unit small, so it fits in a single socket. The idea of multi-core is to deliver an excellent performance while minimizing power consumption and producing lower heat emission than configurations that rely on a single high-clock speed core. This configuration reduces communication latency because the cores can communicate more quickly as they're all on the same chip.

It was reported that this new technology improves the performance and the energy consumption of many operations (Basmadjian and De Meer, 2012). The Wireless Sensor Networks (WSNs) nowadays begin using this technology to improve the sensor node processing power and minimize the power consumption of the processing unit (Munir *et al*., 2014; Shaik *et al*., 2015).

Nevertheless, there was no proof that all operations are suitable for multi-core technology. Video processing is one of those things where the entire task can be split into as many pieces as you'd like. In such a case, having more cores would certainly give an advantage. In other cases, it may be possible to render the operation to be slower or consume much more energy than on a single-core. For example, browser applications make use of only one or two cores, therefore having more cores will not be beneficial to the performance.

In this study, we investigate the effect of multi-core technology on secure communication via cryptography. Specifically, we will test the three RSA main operations, key exchange, encryption and decryption on a single core as well as a multi-core operation to inspect how the speed and storage parameters react over the exhaustive processes such as encryption and decryption.

Methods

We use simulation techniques to come about this (Rahman *et al*., 2012; Zhou and Tang, 2011). For this purpose, we employ standard RSA cryptosystem, tailored to the level of security needed by our application (with very much secure key size as compared to the existing) for better security. The large key size of RSA will be used to encrypt the data in communication between peers as senders and receivers. The increase in key size will help to strengthen the ability to resist the cryptanalysis types of attack. So, it will be more secure.

Nevertheless, not all algorithms can take advantage of the multi-core system and work in parallel and this is due to the processing dependencies which means that the processing steps depends on each other. However, our proposed algorithm performs data encryption process on each block of the input stream independently. This characteristic helps to take advantage of multi-core systems and raises the encryption and decryption speeds. For that, we test the proposed solution on a multi-core system to calculate the encryption and decryption speeds. Table 1 shows the experimental specification that is performed on Intel Core i7 2740 QM with 2.4 GHz processor.

Table 1. Experimentation parameters

System specification	
OS	MAC OS X 10.10
CPU	Intel core i7 2720 QM @ 2.4 GHz
RAM	8 GB-DDR II 1333 MHz
JAVA	Java Console 1.8 (Latest version)
Message size	512 Bite (64 Bytes)
No of cores	{1,2,4} Cores @ 2.4 GHz each
RSA Algorithm	Standard Java security Algorithm

DOI: 10.3844/ajeassp.2016.1054.1061

Results and Discussion

In this section, we analyze the performance of the RSA algorithm based on their timing performance to perform the three major tasks, key generation, message encryption and message decryption.

Key Generation

As key generation is a one-time task, which is done at the initial stage of message exchange, having instructions executed mostly in serial, we therefore simply measure the time taken using the single-core processor. As shown in Fig. 2, from the key size 512 bits until 2048 bits, the increment in the time required is well-behaved. However, beyond 3072 bits, the increment is directed towards exponentiation. Sometimes in the not so far future, we need to content this issues by researching into new techniques.

Message Encryption

For message encryption, the test result in Fig. 3 shows that the behavior was as anticipated. The quad-core is the most efficient in handling the encryption process followed by the dual-core and single-core accordingly. As much as the graph for quad-core technology is flat, it is evident that this technology is able to handle the increment in the key size, to the least up to 8192 bits.

Message Decryption

This performance analysis for message decryption is conducted and the result is shown Fig. 4. As expected, the quad-core technology has outperformed the dual-core and the single-core CPU. It can be observed that the execution time for quad-core is increasing at a very slow pace even at the key size of 8192 bits, at which the single-core is rising at an exponential rate.

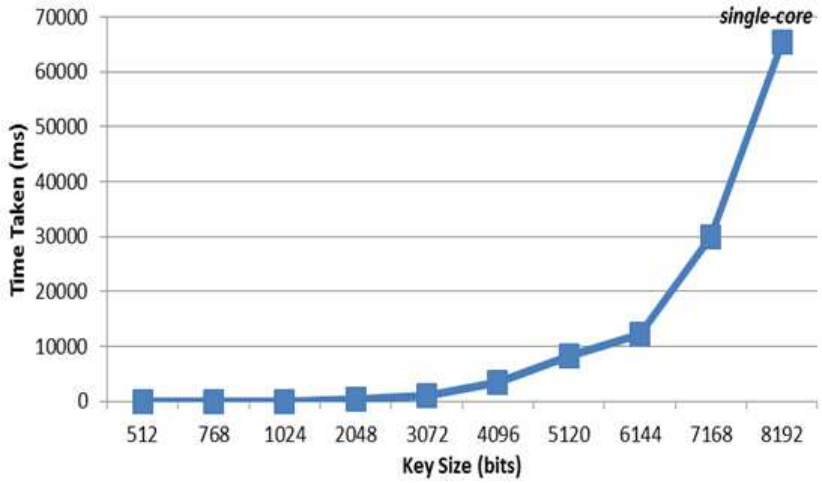

Fig. 2. Key generation time taken as a function of key size

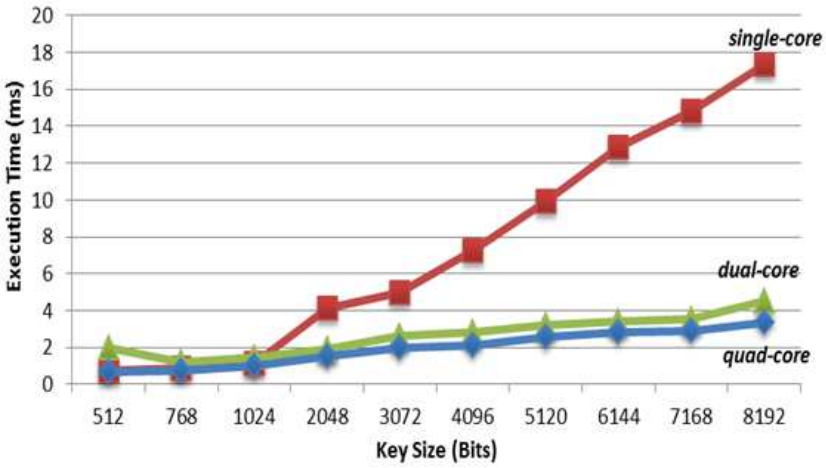

Fig. 3. Encryption time as a function of key size

DOI: 10.3844/ajeassp.2016.1054.1061

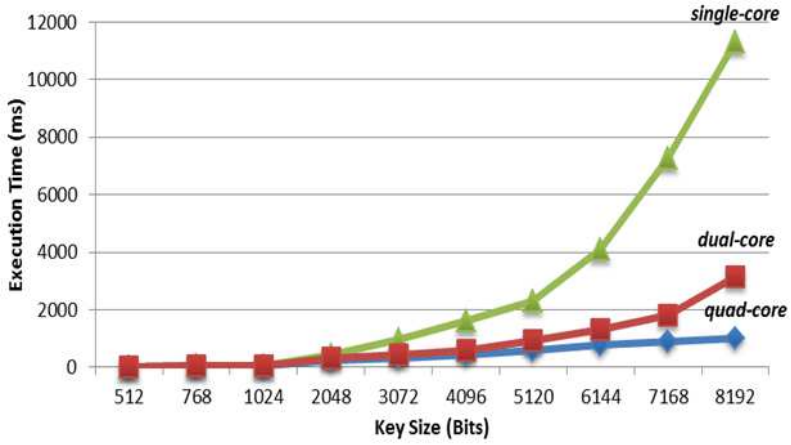

Fig. 4. Decryption time as a function of key size

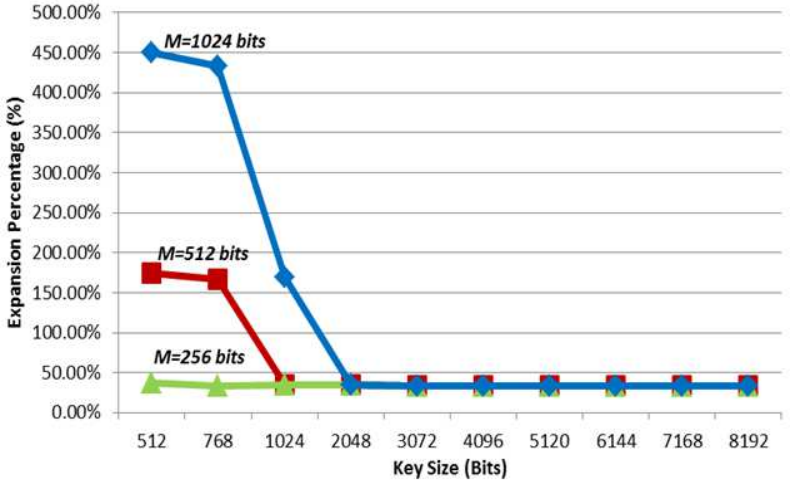

Fig. 5. Message expansion due to encryption process at different key size

Data Expansion Due to Encryption Process

The RSA takes the input of any variable size. In this study, we select three different input sizes, M = 256 bits, 512 bits and 1024 bits to investigate the effect of message size due to the encryption process at the different key size.

For the result shown in Fig. 5, as the input block sizes increase (from 256 bits to 1024), we observe an initial expansion shoots out at the message after encryption especially when the key size is 512 bits and this effect gradually decreases as the key size increases and for all input size, it seems to settle at one third increment of the original message.

This happens to all block sizes and seemingly, it gets stabilized to the point where the key size is at least double the message size. This expansion will also contribute to the longer decryption process as discusses in the first phenomena.

We have noticed the speed improves in encryption and decryption while using keys of lower strength. The larger the key size, the higher the complexity. Thus, the process becomes slower due to longer time is needed to do the calculation. The longer the encryption key, the longer the ciphertext.

Out of regularities from the four different experimentations, we observe the following two noticeable phenomena. The first one is the timing taken to decrypt the message is somewhat 1000 times bigger than the time needed to encrypt the message. This is because the decryption key (d) ranges from just twice as big to almost 17 times as large as the encryption key (e). It is in the know that the value of modulo n is normally much bigger than p and q because $n = p*q$ and the value of d as compared to e. It is more complex to decipher a text than to cipher it because ciphertext is longer than the original text. As a result, decryption is more complex and will end up slower than encryption. So, a balance is to be made by considering both. We found out that 1024 bit RSA is

very efficient in terms of speed and meets minimum security requirement.

Conclusion

In this research, we study how does the multi-core technology reacts to RSA applications by varying the key size. Throughout the study, we discover that multi-core technology does improve the execution time for both encryption and decryption processes. This effect becomes more significant as the key size increases since the one-core processor becomes highly exhaustive. In conclusion, multi-core technology is inevitably a solution of choice for next generation RSA application with growing key size.

Acknowledgement

The authors are grateful to the anonymous referees who have contributed to improving the quality of this paper.

Funding Information

This work is partially supported by Fundamental Research Grant Scheme (FRGS, Grant No: RR074) under the Ministry of Education (MOE) and Universiti Sultan Zainal Abidin (UniSZA), Malaysia.

Author's Contributions

The authors contributed equally to the writing of this paper and they read and approved the final manuscript.

Ethics

This article is original and contains unpublished material. The corresponding author confirms that all of the other authors have read and approved the manuscript and no ethical issues are involved.

References

Al-Hamami, A.H. and B.S.O. Al-Kubaysee, 2011. A fast approach for breaking RSA cryptosystem. World Comput. Sci. Inform. Technol. J., 1: 260-263.

Barenghi, A., B. Guido, B. Luca, P. Mauro and P. Gerardo, 2010. Low voltage fault attacks to AES and RSA on general purpose processors. IACR Cryptology ePrint Archive.

Baset, S.A. and H. Schulzrinne, 2004. An analysis of the skype peer-to-peer internet telephony protocol.

Basmadjian, R. and H. De Meer, 2012. Evaluating and modeling power consumption of multi-core processors.Proceedings of the 3rd International Conference on Future Energy Systems: Where Energy, Computing and Communication Meet, May 09-11, ACM, Madrid, Spain, pp: 1-10. DOI: 10.1145/2208828.2208840

Bi, Y., S. Kaveh, Y. Jiann-Shiun, S. Francois-Xavier and J. Yier, 2016. Leverage emerging technologies for DPA-resilient block cipher design. Proceedings of the Design, Automation and Test in Europe Conference and Exhibition, Mar. 14-18, IEEE Xplore Press, pp: 1538-1543.

Blake, G., R.G. Dreslinski and T. Mudge, 2009. A survey of multi-core processors. IEEE Signal Process. Magazine, 26: 26-37. DOI: 10.1109/MSP.2009.934110

Corchado, J.M., B. Javier, P. de Yanira and I.T. Dante, 2008. Intelligent environment for monitoring Alzheimer patients, agent technology for health care. Decision Support Syst., 44: 283-396. DOI: 10.1016/j.dss.2007.04.008

Coutinho, S.C., 1999. The Mathematics of Ciphers: Number Theory and RSA Cryptography. 1st Edn.,Taylor and Francis, Natick, ISBN-10: 1568810822, pp: 198.

Geer, D., 2005. Chip makers turn to multi-core processors. Computer, 38: 11-13. DOI: 10.1109/MC.2005.160

Juraj, H., 2004. Algorithmics for Hard Problems: Introduction to Combinatorial Optimization, Randomization, Approximation and Heuristics. 1st Edn., Springer.

Miri, A., 2013. Advanced Security and Privacy for Rfid Technologies. 1st Edn., IGI Publishing, Hershey, ISBN-10: 1466636866, pp: 231.

Mohamed, M.A., M.R. Md Said, K.A. Mohd Atan and Z. Ahmad Zulkarnain, 2011. Shorter addition chain for smooth integers using decomposition method. Int. J. Comp. Math. 88: 2222-2232. DOI: 10.1080/00207160.2010.543456

Munir, A., A. Gordon-Ross and S. Ranka, 2014. Multi-core embedded wireless sensor networks: Architecture and applications. IEEE Trans. Parallel Distributed Syst., 25: 1553-1562. DOI: 10.1109/TPDS.2013.219

Pomerance, C., 1996. A tale of two sieves. Not. Amer. Math. Soc., 43: 1473-1485.

Rahman, M., T.K. Saha and A.A. Bhuiyan, 2012. Implementation of RSA algorithm for speech data encryption and decryption. Int. J. Comput. Sci. Netw. Security, 12: 74-82.

Rempe-Gillen, L. and W. Rebecca, 2014. Primality Testing for Beginners. 1st Edn., American Mathematical Society, Providence, ISBN-10: 0821898833, pp: 240.

Rivest, R.L., A. Shamir and L. Adleman, 1978. A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM, 21: 120-126. DOI: 10.1145/359340.359342

Roy, A., J. Xu and M.H. Chowdhury, 2008. Multi-core processors: A new way forward and challenges. Proceedings of the International Conference on Microelectronics, Dec. 14-17, IEEE Xplore Press, pp: 454-457. DOI: 10.1109/ICM.2008.5393510

Saul, D., M. Efren and O. Luis-Guillermo, 2015. Evolutionary programming for the length minimization of addition chains. Eng. Applic. Artificial Intell., 37: 125-134. DOI: 10.1016/j.engappai.2014.09.003

Shaik, A.K., C.A. Kumar and M. Saheb, 2015. Implementation of wireless sensor networks for industrial applications using the multi-core architecture. Int. J. Eng. Res. Applic.

Zhou, X. and X. Tang, 2011. Research and Implementation of RSA algorithm for encryption and decryption. Proceedings of the 6th International Forum on Strategic Technology, Aug. 22-24, IEEE Xplore Press, pp: 1118-1121. DOI: 10.1109/IFOST.2011.6021216