Research Article

Ubi-RKE: A Rhythm Key Based Encryption Scheme for Ubiquitous Devices

Jae Dong Lee, Hyung Jin Im, Won Min Kang, and Jong Hyuk Park

*Department of Computer Science and Engineering, Seoul National University of Science and Technology,
Seoul 139-743, Republic of Korea*

Correspondence should be addressed to Jong Hyuk Park; parkjonghyuk1@hotmail.com

Received 13 June 2014; Accepted 19 July 2014; Published 19 August 2014

Academic Editor: Young-Sik Jeong

Copyright © 2014 Jae Dong Lee et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

As intelligent ubiquitous devices become more popular, security threats targeting them are increasing; security is seen as one of the major challenges of the ubiquitous computing. Now a days, applying ubiquitous computing in number of fields for human safety and convenience was immensely increased in recent years. The popularity of the technology is rising day by day, and hence the security is becoming the main focused point with the advent and rising popularity of the applications. In particular, the number of wireless networks based on ubiquitous devices has increased rapidly; these devices support transmission for many types of data traffic. The convenient portability of ubiquitous devices makes them vulnerable to security threats, such as loss, theft, data modification, and wiretapping. Developers and users should seriously consider employing data encryption to protect data from such vulnerabilities. In this paper, we propose a Rhythm Key based Encryption scheme for ubiquitous devices (Ubi-RKE). The concept of Rhythm Key based Encryption has been applied to numerous real world applications in different domains. It provides key memorability and secure encryption through user touching rhythm on ubiquitous devices. Our proposed scheme is more efficient for users than existing schemes, by providing a strong cipher.

1. Introduction

Ubiquitous computing has recently influenced various fields like health, life, social infrastructure, education, crime prevention, and traffic. In ubiquitous computing environments, one important feature is wireless connectivity because virtually it is available on devices and anywhere. As a result, ubiquitous devices can easily transmit many types of data traffic. However, the convenient portability of ubiquitous devices can lead to security threats, such as loss, theft, data modification, and wiretapping [1, 2]. Developers should seriously consider data encryption to protect data from such vulnerabilities.

In general, researchers in the field of music have recognized rhythm; recently research on rhythms has advanced in many disparate areas, including physical science, biotechnology, and computer engineering. Rhythm has many features that provide numerous applications in various fields such as variety, complexity, and good memorability [3]. Humans

have their own unique rhythms. People exhibit such rhythms in voice or physical phenomena (e.g., rhythm of flashing eyes and heartbeat). In this paper, rhythm refers to musical rhythm that every person possesses. People can utilize this rhythm through a simple touch.

In this paper, we propose a Rhythm Key based Encryption scheme for ubiquitous devices (Ubi-RKE). Ubi-RKE uses touching rhythm on ubiquitous devices in real time. The rich key space, memorable key, and extensible key play a key role in ubiquitous devices security. These advantages provide more convenience than existing schemes, while providing a strong cipher.

We organized this paper as follows. Section 2 discusses the core technologies, existing research, security threats, and considerations of ubiquitous devices. Section 3 explains key generation, encryption and decryption, and Ubi-RKE design. Section 4 describes the comparison analysis of Ubi-RKE. In Section 5, we present performance evaluation. Finally, concluding remarks appear in Section 6.

TABLE 1: Security threats of ubiquitous device and their descriptions.

Security threat	Description
Malware	Malware can infect ubiquitous devices through e-mails, web sites, malicious files, and so forth. Users of infected devices can experience undesired effects, security breaches, and/or damage to the device.
Loss	Loss of a ubiquitous device is equivalent to loss of all information stored on the device. Ubiquitous device in wrong hands leads to unauthorized access and misuse and discloses the contents stored on the device.
Sniffing	Ubiquitous devices use a wireless communication method such as 3G, 4G, Bluetooth, and Wi-Fi. Users can connect to public and private wireless networks to communicate between devices and networks. This means that users have a potential risk of information leakage from an attacker seeking to access the data.
Unauthorized information access	Users install many applications on ubiquitous devices. We cannot check every piece of data inside and outside a network for every application. Recovery of leaked data is difficult, even if the data are stolen through the network.

TABLE 2: Security considerations and their descriptions.

Security consideration	Description
Data confidentiality	Ubiquitous devices use wireless communication to access network services. Attackers attempt data sniffing and data modification at that time. A security technique is required to protect the device from sniffing and modification and ensure data confidentiality.
Data integrity	The receiver of a data communication must verify whether the received message has been modified since it was sent. Data integrity has been protected if the received message is pure. Therefore, technologies such as hash functions are required to ensure the data integrity of received messages.
Device availability	Most portable devices use a battery for power. The user cannot perform any task on the device if the battery is exhausted. Therefore, minimizing battery drain is an important function on these devices. We should consider mobile device-optimized methods to ensure availability.

2. Related Work

In this section, we describe work related to Ubi-RKE, that is, core technologies, existing research on security threats, and considerations.

2.1. Core Technologies

2.1.1. Advanced Encryption Standard (AES) Algorithm. An AES is a symmetric-key block cipher. It encrypts and decrypts the data in blocks with 128 bits and uses 128-, 192-, and 256-bit key sizes [4]. We can use it to reinforce the strength of an operation, depending on the key length used in the AES. Each operation cycle is called round. The key length specifies the number of repetitions in transformation rounds required to convert plaintext into a ciphertext. The number of cycles in repetition for 128-, 192-, and 256-bit keys are 10, 12, and 14, respectively. The AES performs each round with the following operations: SubBytes, ShiftRows, MixColumns, and AddRoundKey [5].

2.1.2. Hash Function. This function converts an arbitrary-length message into a fixed length output. Accordingly, this function returns a hash value [6]. Although it is possible to obtain a result value through operation in a hash function, it is difficult to deduce an input value from the result value. We validate data integrity through these features of the hash function. We ensure that data integrity is a technique in which a message sent from a sender to a receiver arrives without external modification. Various functions such as the Merkle-Damgard- (MD-) based function, Secure Hash

Algorithm- (SHA-) based function, and Hash of Variable Length (HAVAL) are employed along with hash functions.

2.2. Existing Research. Research on cryptography has expanded beyond basic science into applied science. Abusukhon et al. proposed an algorithm that sends plain text encrypted as an RGB image [7]. Dutta et al. proposed an algorithm that encrypts plain text into a representation of continuous musical notes. The sender transmits a ciphertext represented by continuous musical notes to the receiver as a wave file that can generate sound via auxiliary equations [8]. Most encryption algorithms generate random numbers; generating a true random number is difficult. Therefore, encryption algorithms use pseudorandom number generators. Kelsey et al. described the vulnerability and attacks of pseudorandom number generators [9].

There have been many studies on rhythm as a part of biometric authentication. Wobbrock proposed an authentication technique that enters rhythm via a machine consisting of binary sensors called TapSongs [10]. Monroe et al. proposed an algorithm that can generate a lightweight password based on keystroke pattern and trained it to be applied when a password is entered via a keyboard [11]. Chang et al. proposed an authentication system that enters rhythm via mouse clicking in a device without a keyboard [12].

2.3. Security Threats and Considerations of Ubiquitous Devices. We describe the security threats and considerations of ubiquitous devices in this subsection. We summarize security threats and their descriptions in Table 1 and security considerations in ubiquitous devices in Table 2 [13–19].

	t_1	t_2	t_3		$\dots$		t_{n-1}	t_n		
A	1	0	0	0	1	1	1	0	0	1
B	0	0	1	1	1	0	0	0	0	1
C	0	0	0	0	0	0	0	0	0	0
D	0	0	0	0	0	0	0	0	0	0
E	1	0	1	0	0	0	0	0	1	0
F	0	0	1	0	0	1	0	0	0	0
Seed value										

FIGURE 1: Creation of seed a value.

3. Ubi-RKE

3.1. Key Generation. We propose a scheme that encrypts and decrypts the chosen data using rhythm in this subsection. The encryption technique creates a seed value in advance, which it uses to generate a key value. When a user enters rhythm using touch, it is stored in a corresponding track. We use a multitouch environment of ubiquitous devices that generates a number of tracks to divide a screen into subscreens. The scheme creates a seed value through collected index values from the generated tracks. Figure 1 shows four subscreens in a ubiquitous device for input. Users can modify the number of divided screens and input cycles per second, according to required data confidentiality.

Figure 2 shows that the key generator module uses the seed value to generate key values used in the AES. The user enters the seed values. Then in order to prevent key generator from generating the vulnerable keys, the seed value is filtered in the key verification step. The size of the key value is 192 bits, which is represented by the matrix in Figure 2. For each element in the matrix, eight bits (1 byte) are assigned, storing 192 bits.

A hash function ensures data integrity by comparing the hash values at the encryption side and the hash values at the decryption side to check whether the plain text arrived without modification as intended. When a user encrypts a message, the proposed scheme ensures data integrity by hashing a ciphertext.

FIGURE 2: AES algorithm key generation.

3.2. Encryption and Decryption. Acronyms used in the encryption and decryption process are in Table 3.

We discuss the principle, structure, and process of the proposed scheme in this subsection. Figure 3(a) shows the proposed encryption scheme. The scheme creates a seed value according to the rhythm and the user enters it into the device; next the seed value is entered into a key generator. The AES algorithm uses this key along with the plain text to create a ciphertext. And then, ciphertext generates a hash value via a hash function.

Figure 3(b) shows the decryption scheme. The receiving device generates hash values for the ciphertext and compares them with the hash values from the sender.

The ciphertext is transferred to the AES decryption algorithm if the two sets of hash values are the same. If the hash values are not the same, the ciphertext is removed to reduce unnecessary computation. The decryption key should have the same value as the encryption key, because the same seed

TABLE 3: Description of acronym.

Acronym	Description
TR	Touch rhythm
P	Plain text
SV	Seed value
KG	Key generator
AES_E, AES_D	AES algorithm (encryption/decryption)
C	Ciphertext
HF	Hash function
S_HValue, R_HValue	Sender's hash value, receiver's hash value
CMP	Comparison function

value is entered for the decryption key that was entered for the encryption key. The AES algorithm uses this value along with the ciphertext to decrypt it into plain text.

3.3. Ubi-RKE Design. We describe the specification of Ubi-RKE in this subsection. Ubi-RKE consists of seven components: activity, handler, user interface, EventDriven-service (ED-service), TMM (time measurement manager), rhythm cryptography manager (RCM), and rhythm key encryption store (RKE Store). Figure 4 shows the architecture of Ubi-RKE.

There are five activities, that is, select file Activity (SFA), setting activity (SA), rhythm touch activity (RTA), encryption activity (EA), and decryption activity (DA). SFA reads the list of available files in the selected folder. SA processes the setting values such as the number of notes and length of time. RTA supports entering the rhythm key for encryption and decryption. EA and DA perform the encryption and decryption.

The *handler* consists of the initializer, updater, and exception handler. The initializer performs initialization in the updater. The updater transfers the message between the activity and the user interface. The exception handler processes any error messages that occur between the activity and the user interface.

The *user interface* consists of three modules. The environments setting module has two interfaces, that is, the number of notes and the length of time. The number of notes interface has three configuration types that it uses to set up the touching area. The length of time interface has three levels that the user can select, that is, short (S), medium (M), and long (L). Notes have four, six, or eight notes depending on the setting in values. Finally, the file select module provides the file list from which the user can select the needed file.

ED-service is composed of four modules, that is, event analyzer, file manager, touch device checker, and RCM-Connector. Event analyzer can deliver the analyzed values to all ED-service component modules to provide services. File manager controls whether the file is encrypted or not.

TMM can compose the rhythm key with the entered note in the activity component, which measures the time of input from the user.

FIGURE 3: Encryption and decryption process.

RCM is composed of five modules, that is, seed value generator (SVG), rhythm key verifier (RKV), rhythm key generator (RKG), hash function manager (HFM), encryption manager (E-manager), and decryption manager (D-manager). SVG delivers the user rhythm to the RKG module through the RKV. E-manager encrypts the file and then HFM generates the hash values from the encrypted file. Next, D-manager compares the generated hash values with the hash values in the RKE store. Then if the values match, the D-manager decrypts the encrypted file.

The *RKE store* module stores the hash values and the rhythm key. The RKE store provides the hash values and rhythm key value when requested by RCM.

3.4. Ubi-RKE Prototype Implementation. We discuss the implementation of Ubi-RKE in this subsection. Figure 5 shows the initial UBI-RKE screen and encryption process.

The initial screen has two buttons, one for encryption and one for decryption. If the user clicks the encryption or decryption button, the UBI-RKE can encrypt or decrypt the files and folders.

EA encrypts the file or folder with the rhythm of the user in the encryption phase. E-1 shows which files and folders the user has selected for EA encrypt. The user can select one or more files and folders, and then EA will encrypt the file or folder using the user rhythm. E-2 is the configuration screen where the user enters the rhythm key as the number of notes and length of time. The number of notes is represented by the number of buttons, which can be set to four, six, or eight. Length of time indicates the time it takes to enter the rhythm. The S, M, and L lengths are short, middle, and long time, respectively. Based on the configured elements in E-2, the user can enter the rhythm keys.

Figure 6 shows the decryption process. The user checks the encrypted files and folders and selects them in D-1. Next D-2 display screen appears and requests the user to enter rhythm keys. If the rhythm keys are exactly equal to E-3's keys, the decryption process will be performed and then D-3a will list the encrypted files and folders. Because the selected file is decrypted, it is invisible in D-3a. If the rhythm keys are not

FIGURE 4: Ubi-RKE architecture.

FIGURE 5: Encryption execution.

equal to E-3a's keys, the decryption process is not carried out and an error message is shown like D-3b.

In Figure 7, the user enters rhythm keys gradually using six notes such as {E}, {B, E}, and {A, C}. The title bars of each process show a yellow time bar that increases from left to right with the course of time. Step (a) shows an initial screen of rhythm keystroke choices. We know that the user has not entered any keystrokes on this initial screen because the title bar has not changed. When the user enters the single note {E} in step (b), the yellow time bar starts to move across the title bar. Steps (c) and (d) show when the user entered the multinotes {B, E} and {A, C}. Finally, in step (e) the yellow time bar covers the entire title bar indicating that the user has completed entering the rhythm keystrokes.

4. Comparison Analysis

We compare and analyze the two encryption schemes presented in Section 2 [5, 6] with the proposed algorithm in this section. We derived the following factors for comparison

TABLE 4: Encryption analysis comparison.

Classification	Abusukhon et al. [7]	Dutta et al. [8]	Proposed scheme
The set of values usable as a key	O	O	O
Key memorability	X	X	O
Cipher strength	Δ	Δ	O
File independence	X	X	O
Ubiquitous device availability	Δ	Δ	O

(O: strong, Δ : medium, and X: weak).

analysis: the set of values usable as a key, key memorability, cipher strength, file independence, and ubiquitous device availability. Table 4 shows the analysis results.

The set of values usable as a key refers to a set of all keys that we can employ in the algorithm. The algorithms in [7, 8] and the proposed scheme have a sufficient number of usable

FIGURE 6: Decryption execution.

keys. Algorithm [7] performs encryption by substituting an RGB value for each character. Therefore, the number of keys that can be employed will be about $1.16798479 \exp+195$ by performing substitution of 26 characters. Algorithm [8] creates a 72×72 matrix with regard to 72 characters and thereby performs encryption by considering the relationship of the selected character with the preceding character. Therefore, the number of available keys that the algorithm can employ is $(72!)72$. The size of the set of available keys that the proposed algorithm can employ depends on the time and sensitivity of the rhythm the user entered. In the case of a 10 ms sensitivity and a 10-s duration under four subdivided screens, the available number of keys will be 1.0715×10301 . In addition, algorithms [7, 8], as well as our proposed scheme, can extend to match the size of a set of available keys.

Key memorability refers to how easily a person can memorize and remember the key values. Our scheme recognizes behavior and knowledge; therefore, it provides for relatively easier memorization of keys in algorithm [7] or [8]. Algorithms [7, 8] both randomly create key values whenever they perform encryption, which makes it difficult for the user to memorize a key value. In addition, algorithm [8] uses a matrix where the elements consist of musical scales representing key values. Therefore, it is difficult for users to memorize such a large matrix. In contrast, the proposed scheme makes it easier for a user to remember a key value because the algorithm bases the value on the user's chosen behavior with the touch screen.

We can infer the strength of a cipher is high, taking into account the total cost and computing rate of the device, unless the method to figure the algorithm out can be completed

within a given time. The authors of algorithms [7, 8] considered factors such as key space size, work-factor, decryption possibility, and other factors to determine the strength of a cipher. The researchers sufficiently studied the set of values usable as keys. However, they did not perform studies on other factors that may determine the strength of a cipher. We integrated extensively studied AES algorithms into our proposed scheme; this contributes to our confidence in the strength of the ciphers by using tried and tested methods to ensure data confidentiality and integrity.

File independence refers to whether an algorithm requires a file during the encryption and decryption processes. If an encryption scheme requires a file, it can incur overhead due to file input/output, resulting in performance degradation. The proposed scheme does not depend on an external file as do schemes [7, 8]. Algorithms [7, 8] both depend on external files in .png and .wav formats, respectively. They both incur overhead from the complicated internal operations required to reconvert and store these files, which results in resource wastage. The proposed scheme does not create a file for encryption or decryption and prevents this type of resource wastage.

We use ubiquitous device availability referring to the encryption capabilities that use the features of a ubiquitous device. The proposed scheme ensures better availability in ubiquitous devices either than [7] or [8]. Algorithms [7, 8] both generate random key values. However, these algorithms are neither suitable nor available for ubiquitous devices. The proposed scheme dynamically creates a key value based on user input that follows user intentions. In addition, the user can enter rhythm keys using a touch panel irrespective of

FIGURE 7: Rhythm key entry process.

TABLE 5: Specifications of testing devices.

Type	Device 1	Device 2	Device 3
CPU	Dual Core 900 MHz (ARM Cortex-A9)	Quad Core 1.4 GHz (Exynos 4412)	Quad Core 2.3 GHz (Snapdragon 800)
RAM	1 GB (DDR2 SDRAM)	2 GB (DDR2 SDRAM)	3 GB (LP DDR3)
OS	Android 4.0 (Icecream Sandwich)	Android 4.3 (Jelly bean)	Android 4.4 (Kikat)

the presence of a keyboard or a mouse. Therefore, the proposed scheme has better availability in ubiquitous devices than the previously proposed schemes.

5. Performance Evaluation

In this section, we conducted an experiment using a prototype of our proposed scheme. We measure performance by carrying out experiment on three kinds of smart devices. The Table 5 shows the used devices.

We accomplish an experiment to evaluate time effectiveness. For performance comparison, javax.crypto package and the default key generator in standard Java API were used. And we used a fixed-size file (16,650,240 bytes) as a plaintext on experiment to compare encryption time between RKE and JAVA API in different smart devices. Also we experimented 500 times on each key and encryption. The measurement ranges from input of the key to acquisition of ciphertext.

The changes depending on key length, device type, and encryption scheme are shown in Table 6. The difference of consequence between RKE and Java API method is slight (maximum: 33.4 ms, minimum: -48.5 ms, average: 1.4 ms). In particular, the experiment result showed that RKE is faster than traditional method on 192 bits key.

6. Conclusion

Various fields such as health, education, and crime prevention have applied ubiquitous computing. In particular ubiquitous devices using wireless communication can be used anywhere, but this advantage can increase the occurrence of security threats such as loss, theft, and wiretapping.

In this paper, we discussed considerations for security issues of ubiquitous devices and addressed the need for encryption. We also proposed a Rhythm Key based Encryption scheme called Ubi-RKE. It provides both key memorability and strong encryption through user entered

TABLE 6: Performance evaluation.

Key length	Device 1		Type Device 2		Device 3	
	RKE	Java API	RKE	Java API	RKE	Java API
128 bits	5876.5 ms	5843.1 ms	1475.1 ms	1481.9 ms	1156.8 ms	1156.8 ms
192 bits	6416.3 ms	6464.8 ms	1529.0 ms	1532.2 ms	1187.3 ms	1188.9 ms
256 bits	6881.9 ms	6854.9 ms	1587.0 ms	1588.2 ms	1235.4 ms	1221.1 ms

rhythm. We also described the advantages of Ubi-RKE in ubiquitous devices over other methods through comparison and analysis.

Future research will require additional study with applications that use large sets of rhythm created keys. We also suggest that further research provides secure and efficient operations with guaranteed availability that can address attacks that exploit battery power consumption, which is considered as a weakness of ubiquitous devices.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This research was supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the ITRC (Information Technology Research Center) support program (NIPA-2014-H0301-14-1021) supervised by the NIPA (National IT Industry Promotion Agency).

References

- [1] D. J. Levitin and P. R. Cook, "Memory for musical tempo: additional evidence that auditory memory is absolute," *Perception & Psychophysics*, vol. 58, no. 6, pp. 927–935, 1996.
- [2] J.-S. Oh, C.-U. Park, and S.-B. Lee, "NFC-based mobile payment service adoption and diffusion," *Journal of Convergence*, vol. 5, no. 2, pp. 8–14, 2014.
- [3] A. Goode, "Managing mobile security: how are we doing?" *Network Security*, vol. 2010, no. 2, pp. 12–15, 2010.
- [4] S. P. Singh and R. Maini, "Comparison of data encryption algorithms," *International Journal of Computer Science and Communication*, vol. 2, no. 1, pp. 125–127, 2011.
- [5] T.-Y. Chang, Y.-J. Yang, and C.-C. Peng, "A personalized rhythm click-based authentication system," *Information Management & Computer Security*, vol. 18, no. 2, pp. 72–85, 2010.
- [6] M. Abutaha, M. Farajallah, R. Tahboub, and M. Odeh, "Survey paper: cryptography is the science of information security," *International Journal of Computer Science and Security*, vol. 5, no. 3, pp. 298–308, 2011.
- [7] A. Abusukhon, M. Talib, and I. Ottoum, "Secure network communication based on text-to-image encryption," *International Journal of Cyber-Security and Digital Forensics (IJCSDF)*, vol. 1, no. 4, pp. 263–271, 2012.
- [8] S. Dutta, C. Kumar, and S. Chakraborty, "A symmetric key algorithm for cryptography using music," *International Journal of Engineering and Technology*, vol. 5, no. 3, pp. 3109–3115, 2013.
- [9] J. Kelsey, B. Schneier, D. Wagner, and C. Hall, "Cryptanalytic attacks on pseudorandom number generators," *Lecture Notes in Computer Science*, vol. 1372, pp. 168–188, 1998.
- [10] J. O. Wobbrock, "TapSongs: Tapping rhythm-based passwords on a single binary sensor," in *Proceedings of the 22nd Annual ACM Symposium on User Interface Software and Technology (UIST '09)*, pp. 93–96, October 2009.
- [11] F. Monrose, M. K. Reiter, and S. Wetzel, "Password hardening based on keystroke dynamics," *International Journal of Information Security*, vol. 1, no. 2, pp. 69–83, 2002.
- [12] T. Chang, Y. Yang, and C. Peng, "A personalized rhythm click-based authentication system," *Information Management and Computer Security*, vol. 18, no. 2, pp. 72–85, 2010.
- [13] T. Kwon and S. Na, "TinyLock: affordable defense against smudge attacks on smartphone pattern lock systems," *Computers & Security*, vol. 42, pp. 137–150, 2014.
- [14] S. Chiasson, E. Stobert, A. Forget, R. Biddle, and P. C. van Oorschot, "Persuasive cued click-points: design, implementation, and evaluation of a knowledge-based authentication mechanism," *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 2, pp. 222–235, 2012.
- [15] P. Andriotis, T. Tryfonas, G. Oikonomou, and C. Yildiz, "A pilot study on the security of pattern screen-lock methods and soft side channel attacks," in *Proceedings of the 6th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '13)*, pp. 1–6, April 2013.
- [16] C.-I. Fan and Y.-H. Lin, "full privacy minutiae-based fingerprint verification for low-computation devices," *Journal of Convergence*, vol. 3, pp. 21–24, 2012.
- [17] M. Cho and I.-H. Lee, "Optical image encryption and decryption considering wireless communication channels," *Journal of Information Processing Systems*, vol. 10, no. 2, pp. 215–222, 2014.
- [18] H. T. Panduranga, S. K. Naveen Kumar, and H. S. Sharath Kumar, "Hardware software co-simulation of the multiple image encryption technique using the Xilinx system generator," *Journal of Information Processing Systems*, vol. 9, no. 3, pp. 499–510, 2013.
- [19] M. I. Malkawi, "The art of software systems development: reliability, availability, maintainability, performance (RAMP)," *Human-Centric Computing and Information Sciences*, vol. 3, no. 1, article 22, 17 pages, 2013.