QR Tag based Verification Method for Smart IoT Applications

Abbas M. Al-Ghaili, Hairoladenan Kasim, Marini Othman and Zainuddin Hassan

Universiti Tenaga Nasional, Malaysia, {abbasghaili@yahoo.com}

ABSTRACT

A Quick Response (QR) tag based Verification Method (QRVM) used with Internet of Things (IoT) applications is proposed to verify and enable authorized requests by user to access a smart IoT-based application. QR-tag encrypted values are compared to original values. Three-layers have been proposed to attain security objectives. The first layer relates to the IoT-based application's integrity by performing a verification procedure to QR-tag's contents. The second layer concerns the availability whereas user's information are stored in offline database to disable any access caused by threats. The third one periodically generates an authenticated QR tag using 1-session private key to prevent both information leakage to attain it confidential. The QRVM aims to increase the IoT-based application privacy. The QRVM contribution is that it is useful to verify requests that need permissions to access such IoT automatic access systems and smart home applications. QRVM is evaluated in terms of security factors e.g., availability. Results confirmed it is faster than other competitive methods. In addition, results discussed QRVM's robustness against unauthorized access's attempts and brute force attack.

Keywords: IoT, QR tag, smart applications.

I INTRODUCTION

The number of recently published QR-tag related studies in the current decade increases rapidly compared to last three decades, as shown in Figure 1. QR tag features, such as storing a huge number of encrypted information in a small shaped-image, is one of the reasons attracting many researchers to propose secure systems (Nazemzadeh, Fontanelli, Macii, & Palopoli, 2017). Some examples include Internet of Things (IoT) based devices (Liu, Choo, & Grossschadl, 2018) intelligent systems (Rane, Dubey, & Parida, 2017), smart access cards (Huang, Liu, & Chen, 2013), and automation processes (Xiao-Long, Chun-Fu, Guo-Dong, & Qing-Xie, 2017) embedded systems (Ghaffari, Ghadiri, Manshaei, & Lahijani, 2017) providing securely smart services to users.

On the one hand, many smart systems have exploited QR tags due to the easily scanning process. Smart home applications usually have several purposes raised from the need of use. For example, monitoring application (Chen et al., 2015) and biometrics-based home access system (Kanaris, Kokkinis, Fortino, Liotta, & Stavrou, 2016). This variety indicates that IoT relies on some other essential tools e.g., encryption scheme in order for the system to be successfully designed.

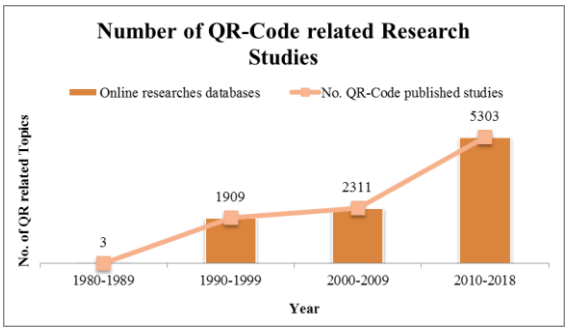

Figure 1. Number of QR-Code related Research Studies and Topics

Additionally, the proposed work in (Nazemzadeh et al., 2017) has used a QR tag in order to head the robot for measurements. The QR tag has been acted as a landmark image to ease the recognition process. One of the reviewed research works (S. S. Lin, Hu, Lee, & Lee, 2015) has proposed a simple method to generate a QR tag in such a way to easily scan it with no much details of black-white-boxes. Another graphical design of QR tag is proposed in (P. Y. Lin, 2016; Tkachenko et al., 2016). The QR tag design has been modified a bit to secure data and to verify documents being authenticated. Patterns of the original QR tag must be replaced by a specific array of patterns in order for the QR tag to be correctly scanned.

These have enhanced the user's life mode (Samuel, 2016) in many aspects. Another important issue to consider is the security and data integrity. Therefore, any change to the system's contents has to be in an authorized manner in order to keep the service implemented correctly and efficiently (Kirkham, Armstrong, Djemame, & Jiang, 2014). QR tag contents are asset and also are a very effective technology on which IoT applications rely.

On the other hand, these useful features a QR tag has, require a strong verification procedure. Once, data stored inside the QR tag has been extracted, the verification process is implemented. This is a very important step to make sure that extracted information is identical to the original one. Thus,

the QR tag related verification process is an essential step in smart home applications, because it affects the system accuracy and privacy. This has caused a very challenging task in designing RQ-tag related verification and encryption method. As a result, many QR tag related researches have used a simple layer of encryption. Such a layer is however suitable with smart applications but those which include sensitive data is vulnerable to threats and attacks. Therefore, to make the QR-tag based Verification Method (QRVM) achieve a high level of privacy, a strong verification policy upon access has been considered in design.

In this paper, the proposed QRVM considers a number of security issues including integrity, authentication, and privacy. Additionally, the QRVM proposes a verification procedure with three layers to increase the security of IoT-based applications. The QRVM secret key design has considered key's length to strengthen the IoT application privacy.

This paper is organized as follows: Section II gives a simple overview. In Section III, the proposed QRVM for IoT applications is explained. Section IV discusses Performance analysis and evaluation. Conclusion is provided in Section V.

II OVERVIEW

The block-diagram, architecture, and security mechanism of QRVM are briefly overviewed.

A. Graphical Overview

The QRVM graphical overview is depicted in Figure 2.

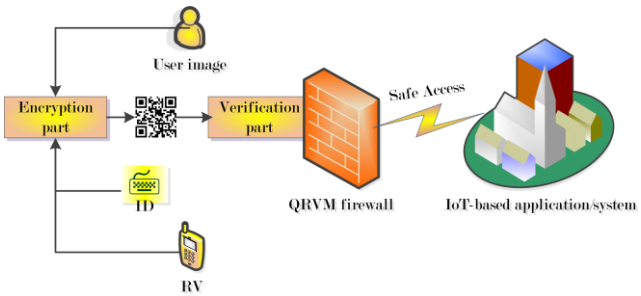

Figure 2. A Graphical Overview of the Proposed QRVM

B. The Proposed QRVM Block-diagram

The proposed QRVM block-diagram is shown in Figure 3. The user needs to provide related information to IoT-based application database. Then, information will be encrypted. The QR-tag will be generated based on the encrypted values. Later, the user needs to enter certain values to allow QR scan procedure. Values are verified using a hash based comparison using original values previously stored.

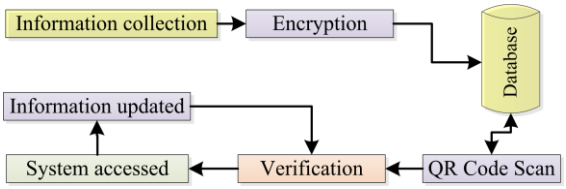

Figure 3. The QRVM Block-diagram

In this figure, once related information has been extracted and encrypted, it is stored. When the user wants to access, QR tag is scanned for a verification procedure via database (e.g., offline database). Finally, the IoT-based application accepts and updates information in the database storage or rejects any an unauthorized access.

C. The Proposed QRVM Architecture

The proposed QRVM architecture is illustrated in Figure 4. Once the QR tag is scanned, its information is verified and then the Reference Value (RV) is provided to the user to increase the access security. After that, the user id will be checked. Finally, the database is updated based on some security criteria.

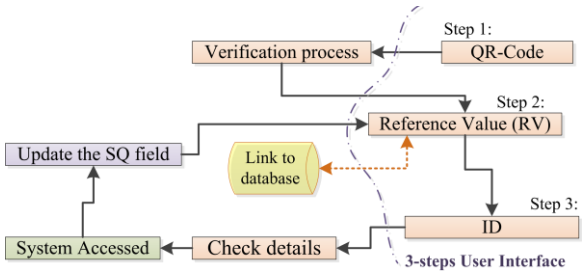

Figure 4. The Architecture for QRVM

D. Security Mechanisms of QRVM FOR IoT Applications

In order to verify the three objectives of computer security for the information system that uses the proposed QRVM. A brief discussion on how these security factors are achieved is discussed in this sub-section. A series of cascade encryption operations is used to encrypt user information using a 1-session private key policy to prevent information leakage.

An updated QR tag is periodically (every 24 hours) generated again using a different RV to guarantee more security in terms of authentication.

Additionally, offline database storage is considered and connected to the encryption procedure. Thus, stored values are re-called by the IoT-based application anytime to keep it always available.

This procedure is followed by the verification process to detect whether the QR tag information is identical to the currently captured ones. This procedure performs the following steps: first, the IoT-based application captures a recent face image.

Then, the image is analyzed and its attributes are compared to original ones stored in the database. Also, the fingerprint will be detected and compared to the reference. Sometimes, the IoT-based application will ask the user to insert RV and ID, see Figure 4.

III THE PROPOSED QRVM FOR IOT APPLICATIONS

The QRVM technical procedure is discussed in detail.

A. QRVM vs. Encryption

After the encryption process has been performed, a hash value is converted to a QR tag logo (as an image) in order to be sent to the user device e.g., mobile app. Then, user is asked to enter a user id value after the IoT-based application has accepted the QR-tag. Another example of QRVM security is that, the RV must be correctly inserted before the access is allowed.

As discussed above, there will be three security levels (QR Scan, RV, user id comparison). Thus, the framework of QRVM verification procedure and its relation to the encryption part is illustrated in Figure 5. The proposed framework has three levels of safety against an unauthorized access and/or modification. In case the IoT-based application has been modified in an unwanted manner, a wrong RV will be obvious when a hash function is tested. Hence, the IoT-based application will not allow any access.

Figure 5 shows the QRVM relation to the encryption procedure. Once, an access is needed by the user, the encryption procedure immediately is called to perform the verification process.

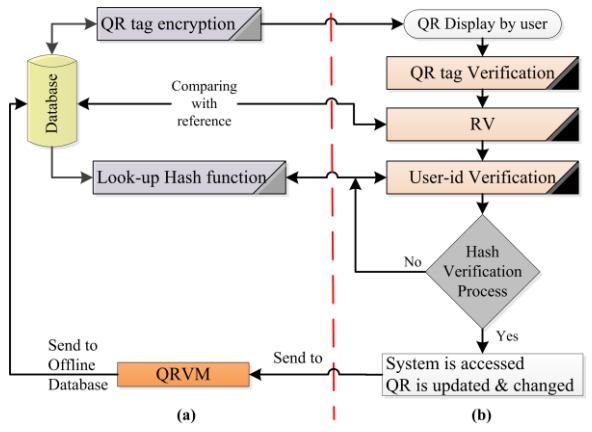

Figure 5. A QRVM Verification Procedure (a) and Encryption (b) Framework

The first security step is to show *QR tag by* using a smart device such as a mobile app to allow the user access the IoT-based application. The pre-generated QR tag which has been stored in the offline database will be used to verify the QR tag with a reference code. If they are identical, the first step is approved; otherwise, the IoT-based application rejects the process. The second step is to insert an RV to make sure that the right user has displayed the correct QR tag (in the previous step). To increase the security, in this step, the RV is inserted manually by using the mobile application. The RV is frequently changed and the offline database storage is updated accordingly. The third security process is *user id* verification; it asks the user to enter the *id* number. This number has been previously generated using a complex process. When it is entered, the IoT-based application uses a special process using a hash function between entered and stored ones.

B. QRVM Flowchart

As discussed earlier, there are three types of verifications which are as follows: a recent face image capturing and fingerprint detection, RV, and *user id*. They are discussed in detail as follows:

Face image and Fingerprint Verification Procedure
The first step in QRVM is to read information from the offline database and look-up table, once the QR tag has been scanned. The purpose is to do a successful and trust comparison between QR tag and original pre-stored information. The comparison could be illustrated as in Figure 6. Once information asked by the user and information stored in the database are identical, the IoT-based application can be accessed. Otherwise, the access is denied.

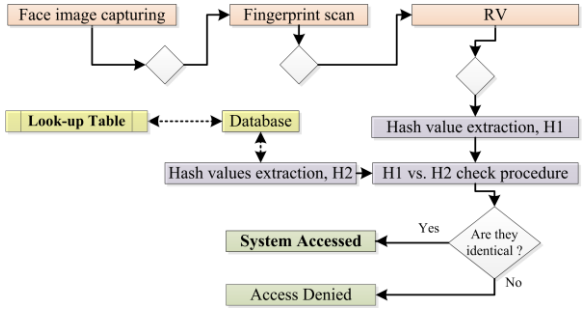

Figure 6. The Proposed QRVM Flowchart

In this figure, certain information stored in the database will be read to check whether they are identical to information collected from the user answer. This comparison simply checks face and fingerprints values with QR tag values. If they are correct, the process goes for the third verification which is the RV; otherwise, the process is halted. The hash value for the RV will be calculated and compared to the look-up table hash value.

Hash based Reference Value (HRV)
HRV is designed to guarantee that RV is periodically changed and it looks like a one-session key. The RV is expired once it has been used for a short period of time. The RV is dependent on a

variable (e.g., *i*) which updates its value frequently; *i*++ or *i*--. This is considered for the correct RV the next time is used. The QRVM will update RV in the database therefore, the field of RV in the database is changed. Then, the RV encryption produces a different hash value. The new generated QR tag will be updated accordingly. Additionally, the hash value for the user's RV will be compared to the one already updated in the IoT-based application database.

In this step, the encryption process is applied. The procedure is designed as follows:

Two references values are randomly generated using a pseudorandom number generator (*PRNG*); i.e., RV1 and RV2 using a algebraic formula denoted by: $RV1 \circledcirc RV2$ to produce a different number. Then, 1-session secret key, k_p is generated using Secure Hash Algorithm 1 (SHA-1). Finally, Eq. (1) is applied.

$$h'_{RV} = hash(RV1 \circledcirc RV2, k_p, M) \quad (1)$$

whereas

- h'_{RV} represents the first calculated hash value
- $RV1 \circledcirc RV2$ is the hash function input and is a produced mathematically value
- $\circledcirc$ represents the mathematical operations applied on $RV1$ and $RV2$
- k_p is the 2nd input of hash function and is a private key used only one time.
- M is the 3rd input of the hash function and is the message obtained from the user's pre-entered data

A further encryption process is used with a rolling function to produce a new hash value, H_{RV} as expressed in Eq. (2):

$$H_{RV} = E(h'_{RV}, k_s) \quad (2)$$

whereas

- H_{RV} is the 2nd hash value and is the encrypted value for $RV1$ and $RV2$
- $E(h'_{RV}, k_s)$ encryption procedure applied on h'_{RV} by using a different secret key, k_s.

The proposed algorithm for this procedure is shown in Algorithm 1:

- Start
- Apply PRNG on $RV1$ and $RV2$ values
- Apply various mathematical operations to produce $RV1 \circledcirc RV2$ value
- Produce k_p
- Implement a hash function to calculate h'_{RV}
- Implement an encryption algorithm to produce H'_{RV}
- Apply a rolling function
- End

Algorithm 1: HRV algorithm

User id Verification

A similar verification procedure is applied on *user id* values. The related pseudo-code is briefly given in Algorithm 2.

- Start
- Do initialization for the following values:
  -Scan QR tag for user #*i*
  -Ask the user to enter a correct id
  -Call stored hash value for the entered id(i) → $hash_{id_{LOOKUP}}$
- Do a Hash function for id → $hash_{id_U}$
- Extract the id_{QR} value stored inside the QR tag
  -Do Hash function for this id → $hash_{id_{QR}}$
- Compare $hash_{id_U}$ *AND* $hash_{id_{QR}}$ to $hash_{id_{LOOKUP}}$
- Return a comparison value (True OR False)
- End

Algorithm 2: User Id Verification

This algorithm performs a comparison between data collected from the user $hash_{id_U}$ and user's QR code $hash_{id_{QR}}$ to be compared to the user id pre-stored in IoT-based application's database $hash_{id_{LOOKUP}}$.

Any mismatch between the three values will detect some possibilities such as:

- if the user has entered a wrong id, it might be that the user has used a different QR tag; and/ or
- if the QR tag value is wrong, the QR tag might be expired.

A simple flowchart is illustrated in Figure 7 to provide a more explanation of Algorithm 2.

IV PERFORMANCE ANALYSIS AND EVALUATION

This section performs an analysis and evaluation procedure. The performance of the proposed research work is discussed. To achieve a high level of performance of the proposed algorithm, various parameters and factors have been considered.

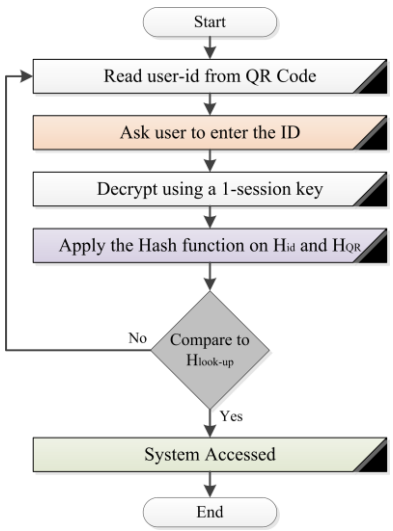

Figure 7. Hash-based Comparison for Verification

A. Security Factors Analysis

Confidentiality

Information is transmitted between two authorized parties, using a strong encryption algorithm. However, information being transmitted between these two parties is only not so much; whereas most information is stored in an offline database. Additionally, the QR tag is periodically re-generated using a 1-session key to increase information confidentiality and keep it secure.

Integrity

To achieve a high level of QRVM integrity, the details and patterns in QR tag image will be scanned and verified. Additionally, face image and fingerprints are verified. If there is any mismatch when comparing to the original values, the answer will be wrong. In that case, the QRVM has no integrity and the verification process will make the process halted. Thus, a third party has modified the QR tag contents or it might be the original QR tag is not currently scanned. Both possibilities will protect the original user QR tag and other contents.

Availability

There will be no access by a third party but only one authorized source is allowed to access thru the offline database given a certain period of time.

B. **Other Factors**

Authentication

In this evaluation, the authentication factor to be verified and tested is denoted by: $F_{authentication}$. To fulfill so, a small portion of grey patterns will be merged with a hash value obtained from the user. The right user will be required to enter a special value predefined earlier. The hash value then will be calculated and compared to the original one stored in the offline database. If the entered value is correct, a certain hash value will be fit with those grey patterns to produce a new QR tag. The QRVM is going to scan the new resulted QR tag. This procedure is performed periodically every 24 hours to guarantee the authority of QR tag issue. Any mistake of the entered value will appear in this step; whereas Eq. (3) is applied for QR tag authentication.

$$F_{authentication} = \begin{cases} 1 & Hash(user) == Hash(db) \\ 0 & Hash(user) \neq Hash(db) \end{cases} \quad (3)$$

whereas

- $F_{authentication}$ is a logical returned value after the comparison is done
- $Hash(user)$ is the hash value given for the user (old value)
- $Hash(db)$ is the new updated hash value stored in the database

If Eq. (3) has been applied, and returned the value "1"; as: $F_{authentication} = \{1\}$, then, the QR tag is generated by the original source. Otherwise, it is generated by an unauthorized source. That means there is no authenticity for the generated QR tag.

Robustness

In order for the QRVM to allow an access to the related IoT-based application or physical device by using the application, three entries which are the QR tag scan, RV, and *user id* will be tested. However, the application might ask the user to re-capture Face image and Fingerprint.

Thus, all three entries must be correct. Otherwise, the IoT-based application does not allow any attempt for access. If the QR tag has been used successfully and one or more of other entries was wrong or mismatched to the original values, the user can't access. This policy of verification is to increase the security thru threats prevention.

Hence, it is clear that any error in the input of the QRVM will lead to the same output. Meaning, any possibility of error existence during access's attempts, it is rejected. That can give the QRVM more robustness against unauthorized attempts. So that threats and actions are prevented.

Computation Time based Efficiency

To verify the efficiency of the proposed algorithm, the selected factor for this verification is the computation time. The computation time is calculated and compared to other existing algorithms.

As discussed earlier on how the encryption process is done, plaintexts (taken from information) have been already hashed, (H_p). Firstly, by using SHA-1, the 1-session private key is generated. The hash value extracted from the QR tag (H_{QR}) and compared to the original hash value stored in the offline database; i.e., H_p and H_{QR}. Then, the user will be asked to enter a RV and user id value as the message being used for this evaluation. After that, the message, session key, and hash values will be encrypted to create a public key. This value is being hashed. Finally, the QR tag will be generated using these information and values.

In this part, the average computation time of QRVM compared to other similar schemes and techniques is shown in Table I.

Table 1. QRVM Computation Time Compared to Other Techniques.

No. Tests	Password; ms	Certificate; ms	(Kim & Jun, 2011); ms	QRVM; ms
10	24.9	45.2	31.4	28.6
20	43.7	91.2	63.1	50.2
50	115.6	212.6	149.0	132.1
100	205.1	452.2	313.3	226.7

This comparison shows that the proposed QRVM is faster than Certificate and the work proposed in (Kim & Jun, 2011). However, the Password technique is faster because the time needed for encryption and verification process of QRVM takes more time for a more secure procedure. Another reason is that, the QRVM has applied the hash function more than one time; that is to increase the IoT-based application safety being accessed.

In Figure 8, the QRVM computation time is on the second rank whereas a less computational time is achieved with a good level of safety and security. In Figure 9, the average computation time is shown whereas its value ranges between 23.8% and 27.2% among others. (4) has been used:

$$Computation\_time_{Avrg} = \frac{T_{QRVM}}{T_{Password}+T_{Certificate}+T_{[Kim]}} \quad (4)$$

whereas

- $Computation\_time_{Avrg}$ is the average computation time for QRVM per each test
- $T_{Password}, T_{Certificate}, T_{[Kim]},$ and T_{QRVM} represent the computation time calculated for Password, Certificate techniques, the method in (Kim & Jun, 2011), and the proposed QRVM, respectively.

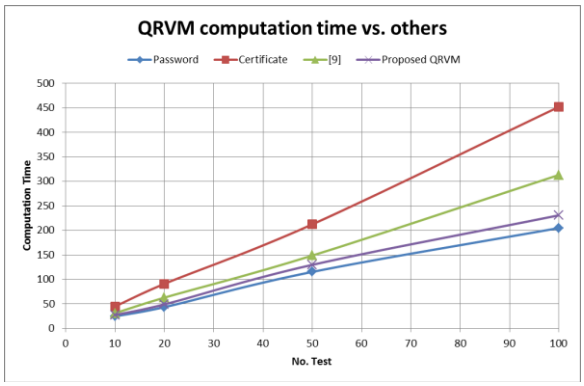

Figure 8. QRVM computation time compared to other methods

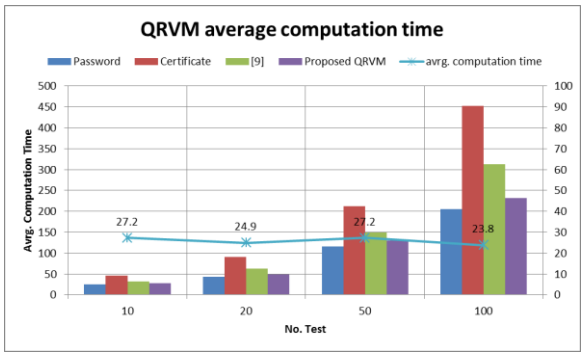

Figure 9. QRVM average computation time

The cumulative average computation time is 25.7%.

Key Length against brute force attack.

Table 2. Key Length based Decryption-time Evaluation.

Key length; size in bits	Number of alternative keys	Time required; 10^6 decryption/µs time in years
168	3.7×10^{50}	6×10^{30}
320	2.1×10^{96}	3.4×10^{76}
384	3.9×10^{115}	6.3×10^{95}

This table shows that a very long time is needed to decrypt a message with different key sizes. The last column mentions the time needed for a IoT-based application in which 10^6 keys could be processed in 1 µs. This is computationally secure.

V CONCLUSION

This paper proposes a simple algorithm used as an access procedure for smart applications and IoT applications such as smart home applications and security gates. The proposed QRVM collects user information and encrypts them in order to create a secure QR tag image. The design of QRVM has considered the security factors and mechanisms. Results and evaluation have discussed different factors such as integrity and availability. Results have confirmed that QRVM is strong against brute force attack in terms of time required to crack it. The performance has confirmed it is real-time responsive, reliable, and useable. To overcome an attack, QRVM produces a QR tag periodaically every a short period of time so that an attack needs further time whereas the original QR-tag is no more used.

ACKNOWLEDGEMENTS

This research is funded by FGRS/1/2017/SS09/UNITEN/02/3, Ministry of Higher Education Malaysia; which is supported by Universiti Tenaga Nasional.

REFERENCES

Chen, Y. H., Tsai, M. J., Fu, L. C., Chen, C. H., Wu, C. L., & Zeng, Y. C. (2015, 9-12 Oct. 2015). *Monitoring Elder's Living Activity Using Ambient and Body Sensor Network in Smart Home.* Paper presented at the 2015 IEEE International Conference on Systems, Man, and Cybernetics.

Ghaffari, M., Ghadiri, N., Manshaei, M. H., & Lahijani, M. S. (2017). P4QS: A Peer-to-Peer Privacy Preserving Query Service for Location-Based Mobile Applications. *IEEE Transactions on Vehicular Technology, 66*(10), 9458-9469. doi: 10.1109/TVT.2017.2703631

Huang, H.-F., Liu, S.-E., & Chen, H.-F. (2013). Designing a new mutual authentication scheme based on nonce and smart cards. *Journal of the Chinese Institute of Engineers, 36*(1), 98-102. doi: 10.1080/02533839.2012.726329

Kanaris, L., Kokkinis, A., Fortino, G., Liotta, A., & Stavrou, S. (2016). Sample Size Determination Algorithm for fingerprint-based indoor

localization systems. *Computer Networks, 101*, 169-177. doi: https://doi.org/10.1016/j.comnet.2015.12.015

Kim, Y. G., & Jun, M. S. (2011, Nov. 29 2011-Dec. 1 2011). *A design of user authentication system using QR code identifying method.* Paper presented at the 2011 6th International Conference on Computer Sciences and Convergence Information Technology (ICCIT).

Kirkham, T., Armstrong, D., Djemame, K., & Jiang, M. (2014). Risk driven Smart Home resource management using cloud services. *Future Generation Computer Systems, 38*, 13-22. doi: https://doi.org/10.1016/j.future.2013.08.006

Lin, P. Y. (2016). Distributed Secret Sharing Approach With Cheater Prevention Based on QR Code. *IEEE Transactions on Industrial Informatics, 12*(1), 384-392. doi: 10.1109/TII.2015.2514097

Lin, S. S., Hu, M. C., Lee, C. H., & Lee, T. Y. (2015). Efficient QR Code Beautification With High Quality Visual Content. *IEEE Transactions on Multimedia, 17*(9), 1515-1524. doi: 10.1109/TMM.2015.2437711

Liu, Z., Choo, K. K. R., & Grossschadl, J. (2018). Securing Edge Devices in the Post-Quantum Internet of Things Using Lattice-Based Cryptography. *IEEE Communications Magazine, 56*(2), 158-162. doi: 10.1109/MCOM.2018.1700330

Nazemzadeh, P., Fontanelli, D., Macii, D., & Palopoli, L. (2017). Indoor Localization of Mobile Robots Through QR Code Detection and Dead Reckoning Data Fusion. *IEEE/ASME Transactions on Mechatronics, 22*(6), 2588-2599. doi: 10.1109/TMECH.2017.2762598

Rane, S., Dubey, A., & Parida, T. (2017, 18-19 July 2017). *Design of IoT based intelligent parking system using image processing algorithms.* Paper presented at the 2017 International Conference on Computing Methodologies and Communication (ICCMC).

Samuel, S. S. I. (2016, 15-16 March 2016). *A review of connectivity challenges in IoT-smart home.* Paper presented at the 2016 3rd MEC International Conference on Big Data and Smart City (ICBDSC).

Tkachenko, I., Puech, W., Destruel, C., Strauss, O., Gaudin, J. M., & Guichard, C. (2016). Two-Level QR Code for Private Message Sharing and Document Authentication. *IEEE Transactions on Information Forensics and Security, 11*(3), 571-583. doi: 10.1109/TIFS.2015.2506546

Xiao-Long, W., Chun-Fu, W., Guo-Dong, L., & Qing-Xie, C. (2017, 20-22 Oct. 2017). *A robot navigation method based on RFID and QR code in the warehouse.* Paper presented at the 2017 Chinese Automation Congress (CAC).