Int. J. Inf. Secur. (2017) 16:525–540
DOI 10.1007/s10207-016-0349-6

REGULAR CONTRIBUTION

Mutual authentications to parties with QR-code applications in mobile systems

Cheng-Ta Huang[1] · **Yu-Hong Zhang**[2] · **Li-Chiun Lin**[3] · **Wei-Jen Wang**[2] · **Shiuh-Jeng Wang**[4]

Published online: 14 September 2016

Abstract User authentication over the Internet has long been an issue for Internet service providers and users. A good authentication protocol must provide high security and mutual authentication on both sides. In addition, it must balance security and usability, which has been shown in the literature to be a difficult problem. To solve this problem, we propose a novel mutual authentication protocol with high security and usability. The proposed protocol was developed for quick response code, a type of two-dimensional barcode that can be photographed and quickly decoded by smartphones. We implemented a prototype using the proposed mutual authentication protocol and demonstrated how the prototype improves usability in a mobile communication system. We also used the Gong–Needham–Yahalom logic with several well-known attack models to analyze the security of the proposed protocol, and we obtained satisfactory results. We expect that using the proposed protocol, Internet service providers will be able to provide a mutual authentication mechanism with high security and usability.

✉ Shiuh-Jeng Wang
sjwang@mail.cpu.edu.tw

[1] Department of Information Management, Oriental Institute of Technology, New Taipei City 220, Taiwan

[2] Department of Computer Science and Information Engineering, National Central University, Taoyuan 320, Taiwan

[3] Information Cryptology Construction Lab, Department of Information Management, Central Police University, Taoyuan 333, Taiwan

[4] Department of Information Management, Central Police University, Taoyuan 333, Taiwan

Keywords Authentication · Mutual authentication · Quick response code · Gong–Needham–Yahalom logic · Mobile system

1 Introduction

In computer science, a protocol is a set of rules that determines how messages are transmitted in telecommunications and computer networking. An authentication protocol aims to confirm the identities of each side by exchanging encrypted private data. Such a protocol is usually applied when users want to access limited web services. Park et al. [1] proposed an approach to classify different authentication protocols in 2000. They classified authentication protocols into eight types according to message properties such as identities, key values, and data freshness. A mutual authentication protocol is an advanced type of authentication protocol that is applied when an extra security level is required.

In addition to the authentication protocol concept, we will briefly explain the background of quick response (QR) codes because we added the QR-code technique to implement our protocol. QR codes are two-dimensional barcodes that were developed by Denso Wave in 1994. Compared with other barcodes, QR codes can be not only printed on paper but also displayed on screens. Decoders are no longer confined to special barcode scanners but can be replaced with mobile devices equipped with camera lenses. Because cameras have become standard equipment on smartphones, the popularity of smartphones creates a very favorable environment for using QR codes. In general, two types of security measures are employed for QR codes: their own security and security applications. In 2010, Kieseberg et al. [2] proposed that a QR-code encoding/decoding mechanism can be used to assist in social engineering and to attack automated systems, e.g., in

Table 1 Data capacity of QR code

Version	Modules	ECC Level	Data bits	Numeric	Alpha-numeric	Binary	Kanji
1	21× 21	L	152	41	25	17	10
		M	128	34	20	14	8
		Q	104	27	16	11	7
		H	72	17	10	7	4
40	177× 177	L	23,648	7089	4296	2953	1817
		M	18,672	5596	3391	2331	1435
		Q	13,328	3993	2420	1663	1024
		H	10,208	3057	1852	1273	784

phishing attacks, SQL injections, and command injections. That was the first study to clearly report that the QR-code technique has security vulnerabilities. Oh et al. [3] proposed an authentication system appropriate for a mobile computing environment using QR codes. It is a good security application for QR codes; our scheme also provides this type of QR-code security.

In this study, we present a mutual authentication protocol for QR codes that satisfies the demands of security and convenience. Compared with other security applications for QR codes [4–8], our protocol can classify users' information into public and secret but embedded into a QR code without changing the image quality. Due to its adoption of the QR-code technique, our protocol has advantages for QR codes, such as quick decoding and high capacity, and can be applied to mobile devices. In addition, we adopt the Gong–Needham–Yahalom (GNY) logic to analyze our protocol and to prove that it complies with the demands of logical reasoning. We also consider several attacks to describe our defense strategy.

The remainder of this paper is organized as follows. In Sect. 2, we provide the background to this proposal by mentioning related work, such as work on QR codes and the Needham–Schroeder (N-S) protocol. Our proposed scheme is described in Sect. 3. Implementations are presented in Sect. 4. Evaluations of our scheme are discussed in Sect. 5. Our conclusions are offered in Sect. 6.

2 Related works

This section introduces related work in the literature. First, we describe the QR-code technique in detail. Second, we explain how the N-S protocol works. Finally, we introduce the GNY logic procedure.

2.1 The QR code

A QR code is a type of matrix barcode that was invented by Denso Wave in Japan in 1994 [9]. QR represents quick response, denoting the feature of rapid decoding [10]. The symbol versions of QR codes range from 1 to 40. The larger the version number, the more data capacity that version has. Table 1 shows the minimum and maximum data capacities of QR codes with different data formats.

Table 2 Error correction capability of QR code

Level	Error correction capability (%)
L	7
M	15
Q	25
H	30

In Table 1, a module is a black or white QR-code square containing one or more pixels. The size of the QR code is determined by how many pixels a module consists of, and the data capacity of the QR code is determined by how many modules are contained in the QR code. The error correction capability (ECC) level is a feature of the QR code that denotes how many modules can be dirty or damaged while still enabling the data to be restored. There are four different ECC levels, L, M, Q, and H. Table 2 shows the exact percentages of these levels in the standard definition.

To ensure that the decoder decodes QR codes correctly, some QR-code modules are allocated to have several function patterns other than general data bit patterns. The structure of a QR code is illustrated in Fig. 1. We briefly describe these special function patterns as follows.

- Finder pattern: Three fixed-size squares of size 7× 7 modules are always arranged in the top left, top right, and bottom left in any version. This arrangement enables the scanner to orient the QR code correctly for decoding.
- Separators: Separators are lines of white modules, one module wide that are placed close to the finder patterns inside the QR code. These resemble the edges of the finder patterns and separate the finder patterns from the remaining QR code.

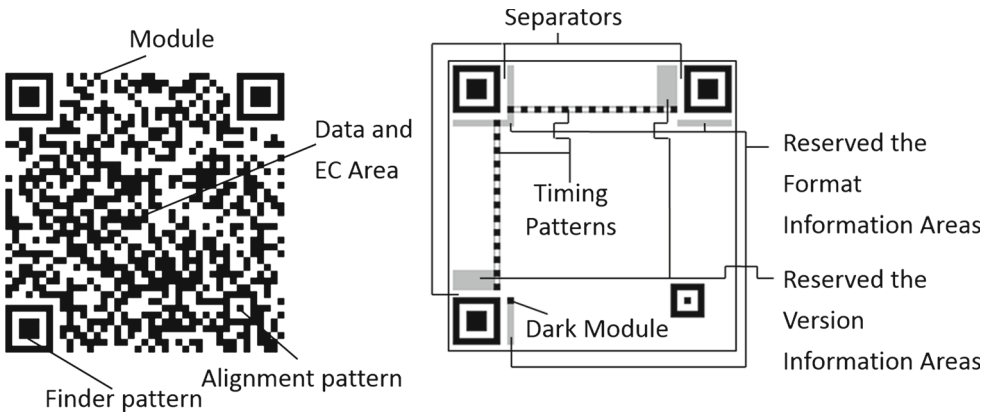

Fig. 1 Structure of QR code

- Alignment pattern: Fixed-size squares of size 5× 5 modules appear in QR codes of version two and larger. The number of alignment patterns depends on the QR-code version number. The greater the version number, the more alignment patterns exist. The patterns have a function similar to that of finder patterns; they assist in positioning and adjusting the QR code when the version number increases.
- Timing pattern: Two lines, one horizontal and one vertical, one module wide and with alternating dark and light modules are placed on the sixth row and sixth column, respectively. These always begin and end with dark modules between the separators. They help the decoder to determine the proportions of black and white modules.
- Dark module: Each of the QR versions has a single dark module near the bottom left of the finder pattern. The precise location of the dark module is ([(4 × Version) + 9], 8), where the top left QR-code module is (0, 0).
- Reserved the Format Information Area: Strips of modules exist next to the separators, except on the left side of top right finder pattern and the upper side of bottom left finder pattern. These strips record information regarding the fault tolerance, stored data format, and data mask.
- Reserved the Version Information Area: QR-code versions 7 and above contain two areas to store version information. The areas are two blocks, one of size 6× 3 that is placed on the upper side of the bottom left finder patter and another of size 3× 6 that is placed on the left side of the top right finder pattern.
- Data and ECC Area: The remaining areas for all function patterns store the general and error correction data bits. All of the data are encoded as bit values and are drawn as white or black modules in a defined order.

The ECC feature of QR codes in our authentication protocol is very important. We will describe in detail how ECC is implemented. The ECC mechanism of QR codes is based on Reed–Solomon (RS) codes [11], which were invented by Irving S. Reed and Gustave Solomon in 1960. RS codes are non-binary cyclic error-correcting codes that view the source symbols as coefficients of a polynomial over a finite field. They have been widely applied in communication and data transmission technologies such as Worldwide Interoperability for Microwave Access and digital subscriber lines. In a QR code, the length of the RS code depends on the number of code words that must be corrected. Assume that we have 200 code words in the QR code to be encoded, of which 150 code words are to be corrected. The length of RS code is 300, so 300 additional codewords must be added to the QR code. The total length of code words in the QR code is 500, including the original data and RS code. Thus, this case corresponds to QR-code ECC level H ($150/500 \times 100\,\% = 30\,\%$), because the total number of code words is 500, 150 of which can be corrected. There is a trade-off between ECC and the original data capacity. The higher the ECC level we choose, the less original data capacity the QR code has in the same QR-code version.

2.2 N-S protocol

The concept of mutual authentication protocol was first derived from the N-S protocol that was proposed by Needham and Schroeder in 1978 [12]. It is used to solve the problem of two parties communicating over an insecure network. Figure 2 illustrates the assumptions of the N-S protocol. There are three principals: A and B, two principals wanting mutual communication, and AS, an authentication server. Kasa, Kasb, and Kab are secrets between the two principals. Two types of cryptography techniques are applied in the N-S protocol, symmetric encryption, and public-key cryptography.

The first type of N-S protocol is based on a symmetric encryption algorithm called the N-S symmetric key protocol. This algorithm is the origin of the Kerberos protocol [13]. Its

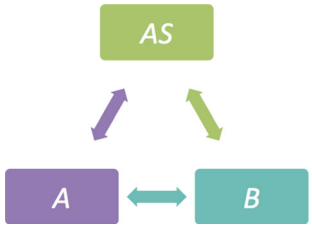

Fig. 2 Assumptions of Needham–Schroeder protocol

principal purposes are to establish a session key between A and B and to use the key to construct a secure communication tunnel on the network. The complete protocol contains five message-exchanging steps. Before beginning message exchanging, the initiator must set up the condition that each principal possesses a secret key that it knows and that is known to its authentication server. We briefly describe the protocol as follows.

Notations

- A: The identity of A which is pre-stored in AS.
- B: The identity of B which is pre-stored in AS.
- I_{A1}: The nonce identifier produced by A and it must be different than others used by A in previous messages of the same type.
- CK: The new key generates by AS when AS confirms both parties identities.
- K_A, K_B: A's and B's secret key.
- $\{\}^X$: The symmetric encryption function using key X.
- I_B: The nonce identifier produced by B.

Messages

1. $A \rightarrow AS : A, B, I_{A1}$
2. $AS \rightarrow A : \{I_{A1}, B, CK, \{CK, A\}^{KB}\}^{KA}$
3. $A \rightarrow B : \{CK, A\}^{KB}$
4. $B \rightarrow A : \{I_B\}^{CK}$
5. $A \rightarrow B : \{I_B - 1\}^{CK}$

The second type of *N-S* protocol, the *N-S public-key* protocol, is based on public-key cryptography. It provides mutual authentication between A and B for communicating on the network. Here, we describe the notation and messages of the protocol; we will explain the details later.

Notations

- A: The identity of A which is pre-stored in AS.
- B: The identity of B which is pre-stored in AS.
- PKB: The B's public key which is pre-store in AS.
- $SKAS$: The AS's secret key. A and B are presumed to possess the AS's public key $PKAS$ to use to decrypt message.
- $\{\}^X$: The asymmetric encryption function using public/secret key X.
- I_A: The nonce identifier produced by A.
- I_B: The nonce identifier produced by B.
- PKA: The A's public key which is pre-store in AS.

Messages

1. $A \rightarrow AS : A, B$
2. $AS \rightarrow A : \{PKB, B\}^{SKAS}$
3. $A \rightarrow B : \{I_A, A\}^{PKB}$
4. $B \rightarrow AS : B, A$
5. $AS \rightarrow B : \{PKA, A\}^{SKAS}$
6. $B \rightarrow A : \{I_A, I_B\}^{PKA}$
7. $A \rightarrow B : \{I_B\}^{PKB}$

There are seven message-exchanging steps in this protocol as compared to five in the *N-S* symmetric key protocol, but steps 1, 2, 4, and 5 are similar key distribution processes. Since public keys are not secret, simply encrypting the data with a public key without considering the data properties can cause vulnerabilities in the encrypted data stream. To prevent leakage, the nonce identifiers I_A and I_B are used to avoid an intruder to catch or count useful information by collecting forged data stream packets.

Some studies have indicated that there are weaknesses in the *N-S* protocols. Denning and Sacco [14] proposed that security based on the assumption that communication keys and private keys are never compromised in the *N-S* protocol is not entirely reasonable. Timestamps must be used to protect against replays of compromised keys. In 1995, Lowe [15] proposed an attack on the *N-S* public-key protocol. It shows that an intruder can impersonate one agent to trick another agent during a run of the protocol in a distributed computer system. To solve the problem, a responder's identity in message 6 of the N-S public-key protocol must be included. In the following year, Lowe [16] used the Failures Divergences Refinement checker [17] to verify the security of the *N-S* public-key protocol. He proved that the same security flaw existed in message 6 and analyzed the full protocol using a logic-based method.

Our proposed protocol is based on the *N-S* protocol including three participants' assumptions and public-key cryptography, but it significantly changes the exchange of messages among the participants and expands the range of applications by adopting the QR-code technique.

2.3 GNY logic

Analyzing a cryptographic protocol using Burrows–Abadi–Needham (BAN) belief logic was first proposed by Burrows et al. [18]. However, BAN logic has defects, such as lack of both flexibility and the ability to reason at more than one level.

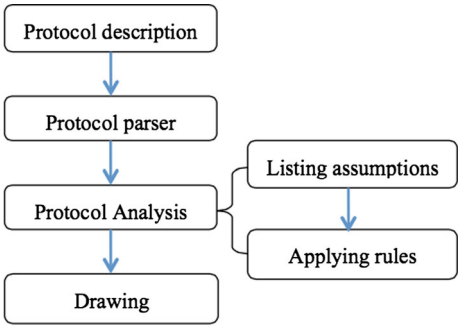

Fig. 3 Overview of the GNY logic

Gong et al. [19] proposed GNY logic for analyzing cryptographic protocols in 1990. Similar to BAN logic, GNY logic offers a procedure for analyzing a protocol, such as creating the assumption list and drawing a conclusion to describe the final position reached by the protocol. We can view GNY logic as an extension of BAN logic that overcomes the flaws of BAN logic by adding more logical rules in its analytic procedure.

Based on the description above, we adopted GNY logic rather than BAN logic to analyze our protocol. Although some modified GNY logic schemes have been proposed [20, 21], the basic structure of GNY logic was not affected. We applied the original GNY logic to analyze our protocol (see Fig. 3).

The first two steps of the GNY logic procedure are the protocol description and parser. These are the processes of interpreting the protocol in terms of the notions and notations of GNY logic. Two kinds of notions, formula and statement notation, are used to describe and parse a protocol. A formula is a particular value in a run. Let X and Y involve formulae, and denote shared secrets S and encryption keys K as two special kinds of formulae. The different types of rules can be described as follows.

Rules

- (X, Y): conjunction of two formulae
- $\{X\}_K$ and $\{X\}_K^{-1}$: conventional encryption and decryption
- $\{X\}_{+K}$ and $\{X\}_{-K}$: public-key encryption and decryption
- $H(X)$: a one-way function of X
- $F(X_1, \ldots, X_n)$: a many-to-one computationally feasible function

A statement reflects some property of a rule. Let P and Q stand for principals. The following are basic statements proposed by GNY logic.

Statements

- $P \triangleleft X$: P is told formula X.
- $P \ni X$: P possesses, or is capable of possessing, X.
- $P \mid\sim X$: P once conveyed formula X.
- $P \mid\equiv \#(X)$: P believes, or is entitled to believe, that X is fresh.
- $P \mid\equiv \phi(X)$: P believes, or is entitled to believe, that X is recognizable.
- $P \mid\equiv P \overset{S}{\leftrightarrow} Q$: P believes, or is entitled to believe, that S is a suitable secret for P and Q.
- $P \mid\equiv \vdash\!\!\rightarrow_{+k} Q$: P believes, or is entitled to believe, that $+K$ is a suitable public key for Q.
- C_1, C_2: conjunction.
- $P \mid\equiv C$: P believes, or P would be entitled to believe, that statement C holds.

The protocol analysis step is the principal reasoning process of GNY logic. It contains two sub-steps, listing assumptions and applying rules. The former is used to assume that each of the participants possesses a secret and believes it is a secret shared between/among them. For example, if we assume that P and Q possess a secret S and that each believes that it is for himself and the other, then we can list these assumptions: $P \ni S$; $Q \ni S$; $P \mid\equiv Q \overset{S}{\leftrightarrow} P$; and $Q \mid\equiv Q \overset{S}{\leftrightarrow} P$. The latter is the most important step in the GNY logic procedure. We examine each message in the protocol to determine whether it satisfies any rule. If it does, then the message conforms to the property of the rule. There are six categories of logical postulates in GNY logic. We explain each category and list portions of postulates that we can apply to our protocol, as follows.

- **Being-Told Rules**: The set of rules dealing with rules a principal receives.
  T1. $\frac{P \triangleleft * X}{P \triangleleft X}$: Being told that a not-originated-here formula is a special case of being told a fact.
  T2. $\frac{P \triangleleft (X,Y)}{P \triangleleft X}$: Being told that a fact implies being told each of its concatenated components.
  T4. $\frac{P \triangleleft \{X\}_{+K}, P \ni -K}{P \triangleleft X}$: If a principal is told a fact that is encrypted with a public key and he possesses the corresponding private key, then he is considered to have also been told the decrypted contents of that rule.
  T5. $\frac{P \triangleleft F(X,Y), P \ni X}{P \triangleleft Y}$: If a principal is told the result of a function F and he possesses one of two arguments, then he is considered to have been told the other argument as well.
- **Possession Rules**: The set of rules specifying the rules a principal can possess by handling rules he already holds.
  P1. $\frac{P \triangleleft X}{P \ni X}$: A principal can possess anything he is told.
  P3. $\frac{P \ni (X,Y)}{P \ni X}$: If a principal possesses a rule, then he can possess any one of the concatenated components of that rule.
  P4. $\frac{P \ni X}{P \ni H(X)}$: If a principal possesses a rule, then he can

possess a one-way computationally feasible function of that rule.

P8. $\frac{P \ni -K, P \ni X}{P \ni \{X\}_{-K}}$: If a principal possesses a rule and a private key, then he can possess the decryption of that rule with the key.

– **Freshness Rules**: The set of rules specifying the formulae a principal can be believe to be fresh.

F1. $\frac{P \mid \equiv \#(X)}{P \mid \equiv \#(X,Y), P \mid \equiv \#(F(X))}$: If P believes that a rule X is fresh, then he is entitled to believe that any rule of which X is a component is fresh and that a computationally feasible one-to-one function F of X is fresh.

F4. $\frac{P \mid \equiv \#(X),\ P \ni -K}{P \ni \#(\{X\}_{-K})}$: If P believes that a formula X is fresh and possesses a private key, then P is entitled to believe that the decryption of X with that key is fresh as well.

– **Recognizability Rules**: The set of rules specifying the rules a principal can believe to be recognizable.

R4. $\frac{P \mid \equiv \phi(X),\ P \ni -K}{P \ni \phi(\{X\}_{-K})}$: If P believes that a rule X is recognizable and P possesses a private key, then $P$is entitled to believe that the decryption of X with that key is recognizable.

R5. $\frac{P \mid \equiv \phi(X),\ P \ni X}{P \mid \equiv \phi(H(X))}$: If P believes that a rule X is recognizable and possesses X, then he is entitled to believe that a one-way function of X is recognizable.

– **Message Interpretation Rules**: The set of rules enabling principals to advance their beliefs about other principals by examining messages they receive.

I2. $\frac{P \lhd * \{X, <S>\}_{+K},\ P \ni (-K, S),\ P \mid \equiv \overset{+K}{\rightarrow} P,\ P \mid \equiv P \overset{S}{\leftrightarrow} Q,\ P \mid \equiv \phi(X,S),\ P \mid \equiv \#(X,S,+K)}{P \mid \equiv Q \mid \sim (X, <S>),\ P \mid \equiv Q \mid \sim \{X, <S>\}_{+K},\ P \mid \equiv Q \ni +K}$:

Suppose that for principal P all of the following conditions hold:

1. P receives a rule consisting of X concatenated with S, encrypted with a public key, and marked with a not-originated-here mark;
2. P possesses S and the corresponding private key;
3. P believes the public key is his own;
4. P believes S is a suitable secret for himself and Q;
5. P believes that X concatenated with S is recognizable;
6. P believes that at least one of S, X, or $+K$is fresh.

Then, P is entitled to believe that

1. Q once conveyed the formula X concatenated with S;
2. Q once conveyed the formula X concatenated with S and encrypted with the public key;
3. Q possesses the public key.

I3. $\frac{P \lhd * H(X, <S>),\ P \ni (X,S),\ P \mid \equiv P \overset{S}{\leftrightarrow} Q,\ P \mid \equiv \#(X,S)}{P \mid \equiv Q \mid \sim (X, <S>),\ P \mid \equiv Q \mid \sim H(X, <S>)}$: Suppose that for principal P all of the following conditions hold:

1. P receives a rule consisting of a one-way function of X and S marked with a not-originated-here mark;
2. P possesses S and X;
3. P believes S is a suitable secret for himself and Q;

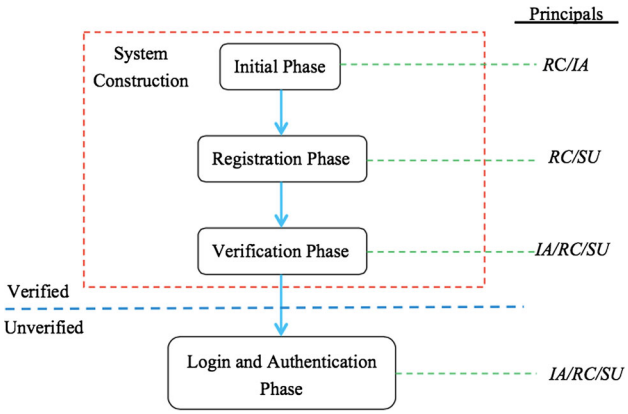

Fig. 4 Protocol architecture

4. P believes that either S or X is fresh.

Then, P is entitled to believe that

1. Q once conveyed the rule X concatenated with S;
2. Q once conveyed the one-way function of X concatenated with S.

– **Jurisdiction Rules**: The set of rules used to represent trust and delegation between two principals.

J2. $\frac{(P \mid \equiv Q \mid \Rightarrow Q \mid \equiv *,\ P \mid \equiv Q \mid \equiv Q \mid \equiv C)}{(P \mid \equiv Q \mid \equiv C)}$: If P believes that Q is honest and competent and Preceives a message X $\sim>$ C that he believes Q conveyed, then P can be expected to believe that Q really believes C.

3 Our scheme

This section discusses how we designed our mutual authentication protocol for QR codes step by step. First, we present an overview of the proposed protocol. Next, we explain the details of our system construction procedure. Last, we describe the login and authentication phase to illustrate how each participant works in the real world.

3.1 Protocol overview

The architecture of the proposed protocol is presented in Fig. 4. There are three principals in the protocol, service user, institution agency, and registration center. A service user (SU) is anyone who wants to access a variety of services, such as an e-bank, a Wi-Fi network, or cloud services. An institution agency (IA) is responsible for verifying whether SU can access the service provider. A registration center (RC) plays an unique role in the system in accepting registering from SUs and IAs and stores related information provided by SUs and IAs.

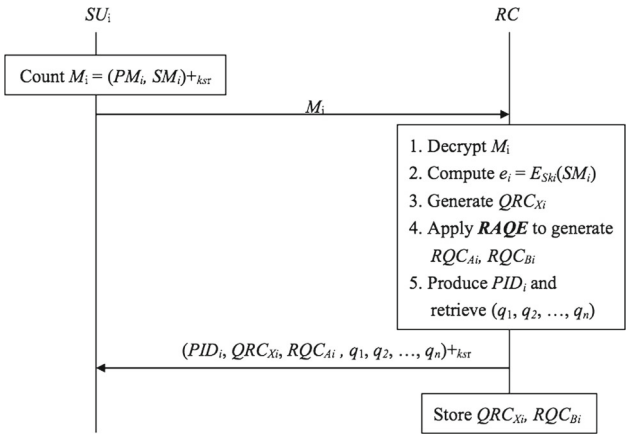

Fig. 5 Registration phase

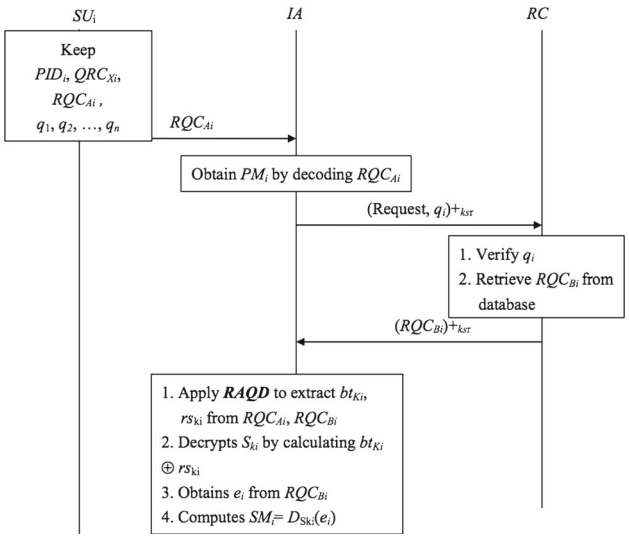

Fig. 6 Verification phase

The entire protocol consists of four phases that can be divided into the first three phases and the last phase. The former phases, which comprise the system construction stage, are used to establish relationships among *RC*, *IA*, and *SU*. After finishing that stage, all of the involved *IA*s and *SU*s will be verified by *RC*. The login and authentication phase, the last phase in the protocol, executes in an unverified circumstance. The original purpose of this phase was to achieve mutual authentication between *IA* and *SU* by *RC* exchanging prestored authentication information to *IA* and *SU*, respectively. If *IA* and *SU* have acquired expected authentication information, then *SU* believes that *IA* is verified and is allowed to access *IA*'s service.

3.2 System construction

The system construction stage is the process of constructing trusted relationships among *SU*, *IA*, and *RC*. The stage contains three phases, initial, registration, and verification. All of the phases are performed under the assumption that *RC* has been verified and trusted. First, *IA* and *SU* go to *RC* to register and exchange the related authentication information in the initial and registration phases, respectively. *IA* obtains *SU*'s portion of the registration information from *RC* to verify *SU* in the verification phase later. All of the phases in the system construction stage are executed again whenever a new *IA* or *SU* wants to join the system. The detailed descriptions are provided in the following list, and we illustrate the registration and verification phases in Figs. 5 and 6, respectively.

Initial Phase

Goal: Construct relationships between *IA*s and *RC*.

Step 1 Each IA_i asks to join the system and provide data, such as physical address and network domain, to *RC*, where $i = 1, 2, \ldots, n$.

Step 2 *RC* verifies the legitimacy and configuration requirements of each IA_i. If holds, then generates (M_{IAi}, r_{IAi}), where M and r denote a unique identity message and a random number, respectively, for each IA_i.

Step 3 *RC* computes and stores $q_i = E_{xi}(M_{IAi}, r_{IAi})$, where $E_{xi}(.)$ represents a symmetric encryption function using the random key x_i, and then assigns q_i to IA_i in a secure manner.

Step 4 IA_i stores q_i secretly in the host machine.

Registration Phase

Goal: *SU* registers with *RC*.

Step 1 SU_i prepares public message PM_i and secret message SM_i as registration information and then transmits $M_i = (PM_i, SM_i)+_{ksr}$ to *RC*, where $(.)+_{ksr}$ denotes an asymmetric encryption function using *RC*'s public key $+ksr$ and private key $-ksr$. The public messages have low privacy demands, such as username or ID number; otherwise, the secret messages are highly sensitive data such as credit card numbers or conviction or medical records.

Step 2 *RC* acquires (PM_i, SM_i) by decrypting M_i and then computes $e_i = E_{Ski}(SM_i)$, where S_{ki} is a random key.

Step 3 *RC* randomly chooses a bit string bt_{ki} to encode as the QR code QRC_{Xi}, where the length of bt_{ki} is equal to the length of S_{ki}.

Step 4 RC applies ***RAQE*** to generate a revised QR code RQC_{Ai} by inputting PM_i and bt_{ki}.

Step 5 RC computes $rs_{ki} = (S_{ki} \oplus bt_{Ki})$ and applies ***RAQE*** again to generate a revised QR code RQC_{Bi} by inputting e_i and rs_{ki}.

Step 6 RC passes $(PID_i, QRC_{Xi}, RQC_{Ai}, q_1, q_2, \ldots, q_n)+_{ksr}$ to SU_i where PID_i is an unique identity to identify a SU_i and stores QRC_{Xi} and RQC_{Bi} in the host database.

Verification Phase

Goal: *IA* verifies *SU* by *RC* and obtains *SU*'s public and secret messages.

Step 1 *IA* obtains RQC_{Ai} from SU_i, when SU_i first accesses *IA*'s services.
Step 2 *IA* directly decodes RQC_{Ai} using a computer or a mobile device equipped with cameras to obtain PM_i.
Step 3 *IA* sends a message (Request, q_i)+$_{ksr}$ to *RC* to obtain RQC_{Bi}, where ^{+}kir is the public key, and the private key ^{-}kir is shared between *IA* and *RC*.
Step 4 *IA* applies ***RAQD*** to extract bt_{Ki} by inputting RQC_{Ai} and to extract rs_{ki} by inputting RQC_{Bi}. *IA* then decrypts S_{ki} by calculating $bt_{Ki} \oplus rs_{ki}$.
Step 5 *IA* obtains e_i from RQC_{Bi} using a computer or a mobile device equipped with cameras and then computes $SM_i = D_{Ski}(e_i)$, where $D_{Ski}(.)$ represents a symmetric decryption function using the key S_{ki}.

Revised Algorithm of QR-code Encoding (RAQE)
Input: A Message 1 M_1, A Message 2 M_2
Output: A revised QR code RQC
Step 1 According to the lengths of M_1 and M_2, determine the QR-code version and error correlation level that are to be used. The normal and error correlation capacities of selected QR-code settings are greater than the lengths of M_1 and M_2, respectively.
Step 2 Encode M_1 as a QR code QC using the normal QR-code algorithm with a preselected setting.
Step 3 Convert M_2 into a string of bits sb.
Step 4 Generate a revised QR code RQC by revising QR-code modules as follows:

$$module = \begin{cases} black, & \text{if } sb = 0 \\ white, & \text{if } sb = 1 \end{cases} \quad (1)$$

The locations of the revised modules are limited to data and error correction code words and are chosen not to be repeated by the random generator $R(s)$, where s is a random seed equal to $H(M_1)$.
Step 5 Output RQC.

Revised Algorithm of QR-code Decoding (RAQD)
Input: A revised QR code RQC
Output: A Message 2 M_2
Step 1 Decode RQC using the normal QR-code algorithm to obtain a message M_1.
Step 2 Extract a string of bits sb in order by referring to RQC's module as follows:

$$sb = \begin{cases} 0, & \text{if } module\ is\ black \\ 1, & \text{if } module\ is\ white \end{cases} \quad (2)$$

The order of extraction is determined by the random generator $R(s)$, where s is a random seed equal to $H(M_1)$.
Step 3 According to the format of M_1, convert sb to a message M_2.
Step 4 Output M_2.

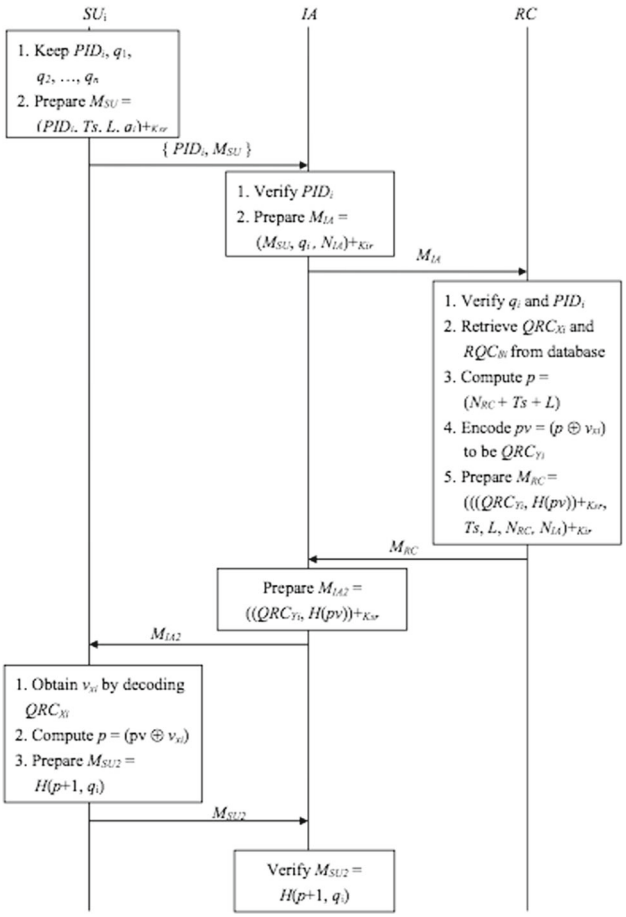

Fig. 7 Login and authentication phase

3.3 Login and authentication phase

The login and authentication phase is performed whenever an unverified *SU* wants to access services provided by *IA*. A detailed description of this phase is shown in Fig. 7. Figure 8 illustrates actual application scenarios for *IA*, *SU*, and *RC*, with the numbers indicating the message passing order. *IA* and *SU* achieve the purpose of mutual authentication by exchanging messages with *RC*. All of the processes can be conducted on a mobile or desktop device with computing and network capacities. To avoid a replay attack, two unique random values, N_{IA} and N_{RC}, are applied to each login session. The timestamp Ts and the length of valid time L for each authentication are set when *SU* initiates a login request. In addition, an appropriate hash function $H(.)$ is used to generate a portion of the authentication messages. We explain the steps of this phase in the following list.
Step 1 SU_i asks *IA* to log in by sending PID_i and $M_{SU} = (PID_i, Ts, L, q_i)+_{Ksr}$.
Step 2 *IA* receives PID_i and M_{SU} and then sends $M_{IA} = (M_{SU}, q_i, N_{IA})+_{Kir}$ to *RC*, after the verification of PID_i, where N_{IA} is a nonce for *IA*.

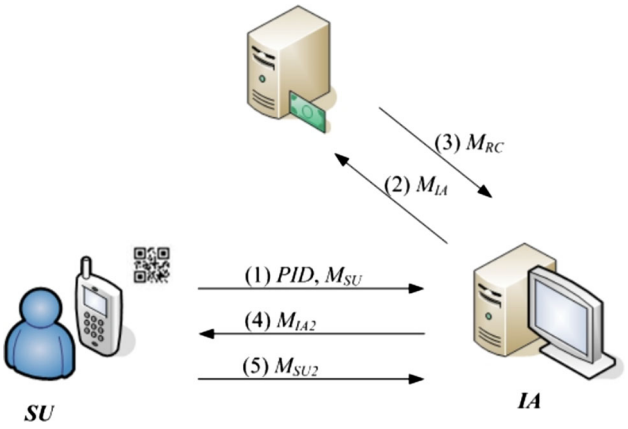

Fig. 8 Diagram of proposed mutual authentication protocol

Step 3 *RC* decrypts M_{IA} and M_{SU} to verify q_i and PID_i. If holds, then retrieves QRC_{Xi} and RQC_{Bi} from the database provided by SU_i in the registration phase.
Step 4 *RC* decodes QRC_{Xi} to obtain value v_{xi} and generates a nonce N_{RC} equivalent in length to v_{xi} using a one-time password (OTP) generator.
Step 5 *RC* computes $p = (N_{RC} + Ts + L)$ and modifies p to be equivalent in length to v_{xi} by abandoning its highest bits. Then, *RC* encodes $pv = (p \oplus v_{xi})$ as a QR code QRC_{Yi}.
Step 6 *RC* sends $M_{RC} = ((QRC_{Yi}, H(pv))+_{Ksr}, Ts, L, N_{RC}, N_{IA})+_{Kir}$ to *IA*.
Step 7 *IA* receives M_{RC} from *RC* and sends $M_{IA2} = (QRC_{Yi}, H(pv))+_{Ksr}$ to SU_i.
Step 8 SU_i receives M_{IA2} from *IA*. Then, SU_i decodes prestored QRC_{Xi} using a computer or mobile device equipped with cameras to obtain v_{xi} and retrieves p by calculating $(pv \oplus v_{xi})$. SU_i sends $M_{SU2} = H(p + 1, q_i)$ to *IA*.
Step 9 *IA* verifies SU_i by computing whether M_{SU2} is equal to $H(p + 1, q_i)$.

4 Architecture simulations

We present the basic implementations using simulation models to implement the authentication system process partially and demonstrate how it works step by step. The entire authentication system is the process in which all of the principals generate and exchange messages. Since the authentication system can use all of the popular hash and cryptography techniques that meet the security requirements, we focus on how to apply the QR-code technique in the system. First, we describe the implementation environment, including the programming language, tools, and relevant applications. Then, we present the design of an application to show how to encode and decode a revised QR code. Finally, we use an example to describe how *SU* executes the proposed protocol.

4.1 Implementation environment

There are three operations in the implementation environment, QR-code generation, QR-code decoding using a mobile device equipped with a camera, and revised-QR-code processing. First, we used a popular QR-code barcode generator developed by RACO Industries to generate a normal QR code. The generator's user interface is presented in Fig. 9. Any user can operate all of the functions, such as inputting encoding messages or choosing a QR-code version or EC level, directly on the web page. The settings of our implementation environment are described in Table 3.

To perform normal QR-code decoding, we used a mobile device equipped with a camera and an installed QR-code decoder application called QR Droid. The specifications for the operations are given in detail in Table 4.

For the revised-QR-code processing, we designed an application called *Revised_QR_Code* using VB.net and the development tool Microsoft Visual Studio 2010. These both were executed on a computer with Microsoft Windows 7 Home Premium ×64 Service Pack 1, an Intel Core i5 2.67 GHz processor, and 8 GB of RAM.

4.2 Authentication system implementation

In this section, we present an example to explain how to use the QR-code technique in the proposed authentication protocol. The implementation system includes two phases, generating a revised QR code and extracting messages from the revised QR code. The flowchart of the implementation system is presented in Fig. 10.

In Fig. 10a, PM is first encoded as a QR code using the QR-code generating operation. Then, SM is added to the QR code to perform the revised-QR-code processing operation and to produce a revised QR code. In Fig. 10b, the revised QR code is first decoded with QR-code decoding using a mobile device operation to produce PM. Then, PM is added to the revised-QR-code processing operation to assist in extracting SM from the revised QR code.

Suppose that we have PM = PDCLab 2014/03/30 A123456789 and SM = 01302968 15:30:28. The processes of encoding messages to generate a revised QR code are as follows:
Step 1 Input *PM* into the *RACO QR-Code Barcode Generator* to generate a QR code image, as shown in Fig. 11.
Step 2 Start the *Revised_QR_Code* application, as shown in Fig. 12.
Step 3 Load the QR code image encoded from *PM* and input *PM* to the textbox, as shown in Fig. 13.
Step 4 Input *SM* into the textbox and press the *Process* button to execute the *Revised-QR-Code processing* operation. The output result of the revised QR code image is shown in Fig. 14.

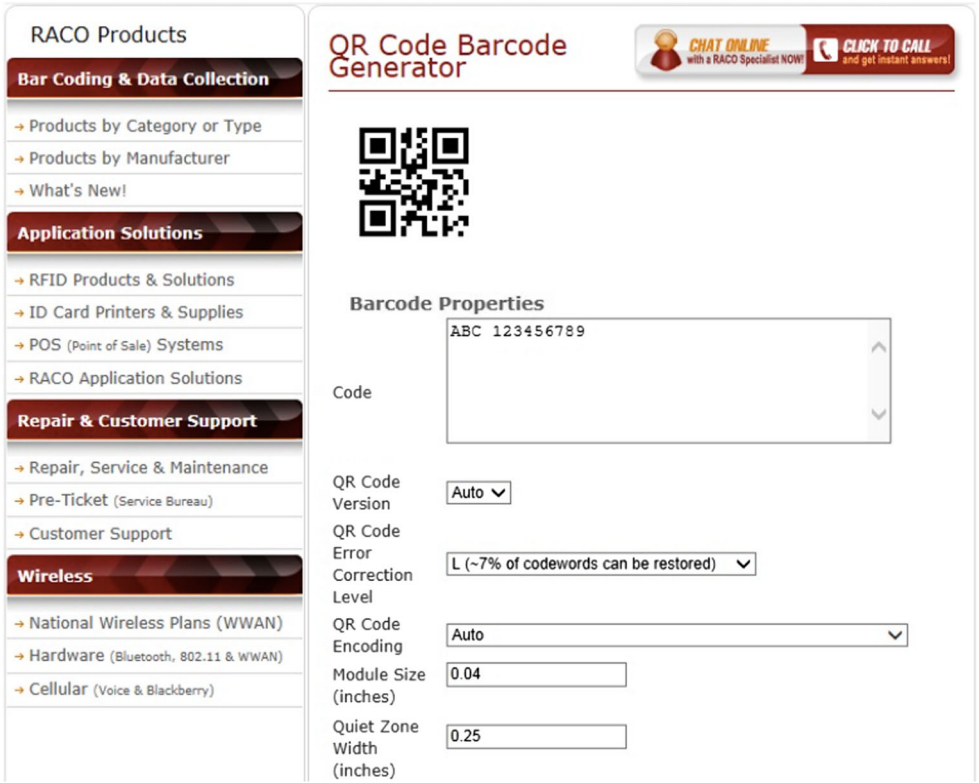

Fig. 9 RACO QR code barcode generator (http://www.racoindustries.com/barcodegenerator/2d/qr-code.aspx)

Table 3 Settings of RACO QR-code barcode generator

Function name	Description	Implement setting
QR-code version	QR-code version from V01 to V40 or automatically set according to the length of messages	V06
QR-code error correction level	QR-code error correction level including L, M, Q, and H	Q
QR code encoding	The encoding message format setting including numeric, alphanumeric, Kanji, and byte, or automatically set by detecting messages	Auto
Module size	The QR-code module size (0.04 in. denotes each module including 4 × 4 pixels	0.04
Quiet zone width	The QR-code quiet zone width (0.01 in. denotes one pixel)	0.25
Top margin	The top margin height of a QR-code image (0.01 in. denotes one pixel)	0.25
Bottom margin	The bottom margin width of a QR-code image (0.01 in. denotes one pixel)	0.25
Image format	The QR-code image format including BMP, GIF, JPG, and PNG	BMP

Table 4 Specifications in decoding a QR code

Component	Specification
Mobile device	HTC new one
Camera	Ultra pixel camera
Application	QR Droid

Suppose that we have a revised QR code. The processes involved in extracting messages from the revised QR code are as follows:

Step 1 Use a mobile device equipped with a camera and with the application *QR Droid* installed to extract *PM*, as shown in Fig. 15.

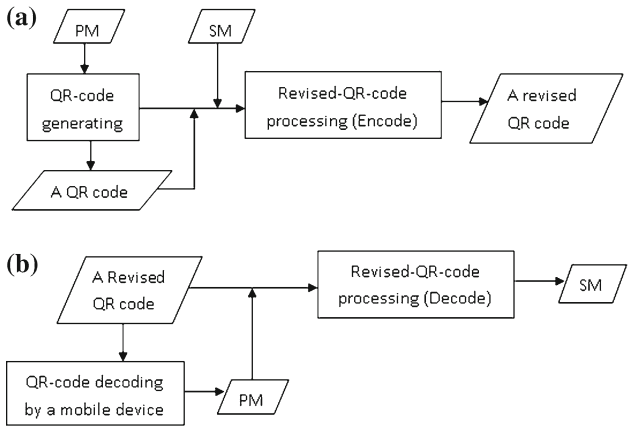

Fig. 10 Implement system with **a** embedding and **b** extracting messages using QR-code technique)

Fig. 11 Generate a QR code

Step 2 Start the *Revised_QR_Code* application and load the revised-QR-code image, as shown in Fig. 16.
Step 3 Input *PM* into the textbox and press the *Process* button to execute the *Revised-QR-Code processing* operation. The output result of *SM* is shown in Fig. 17.

5 Performance evaluations

In this section, we propose three methods of evaluating our authentication protocol. First, we explain data capacity, including *PM* and *SM*, with different QR-code versions. Second, we draw the protocol analysis process using GNY logic. Finally, we consider four normal attacks against the security of a general authentication protocol and analyze whether our protocol can combat such attacks effectively.

5.1 Data capacity with different QR-code versions

Several factors impact the data capacity of a QR code, including its version number, ECC level, and module size in a limited area. Basically, the larger the version number of the QR code, the more data capacity it has. For the same version number, the higher the ECC level, the less pure data capacity the QR code has. In our proposed method, the original data are classified as *PM* and *SM*, and each takes its own approach to encoding but is employed in the same QR code. The capacity of *PM* is equal to the specific pure data capacity of the QR code, because it is encoded using the original QR-code method. In contrast, the capacity of SM is determined by the ECC level, because it is embedded using our revised embedding algorithm. According to the description of the ECC mechanism in QR code, the formula for calculating the capacity of *SM* can be written as

$$\mathrm{SM}_{\mathrm{capacity}} = \left\lfloor \frac{\mathrm{PM}_{\mathrm{capacity}} \times \mathrm{ECC\ Level}}{1 - 2 \times \mathrm{ECC\ Level}} \right\rfloor,$$

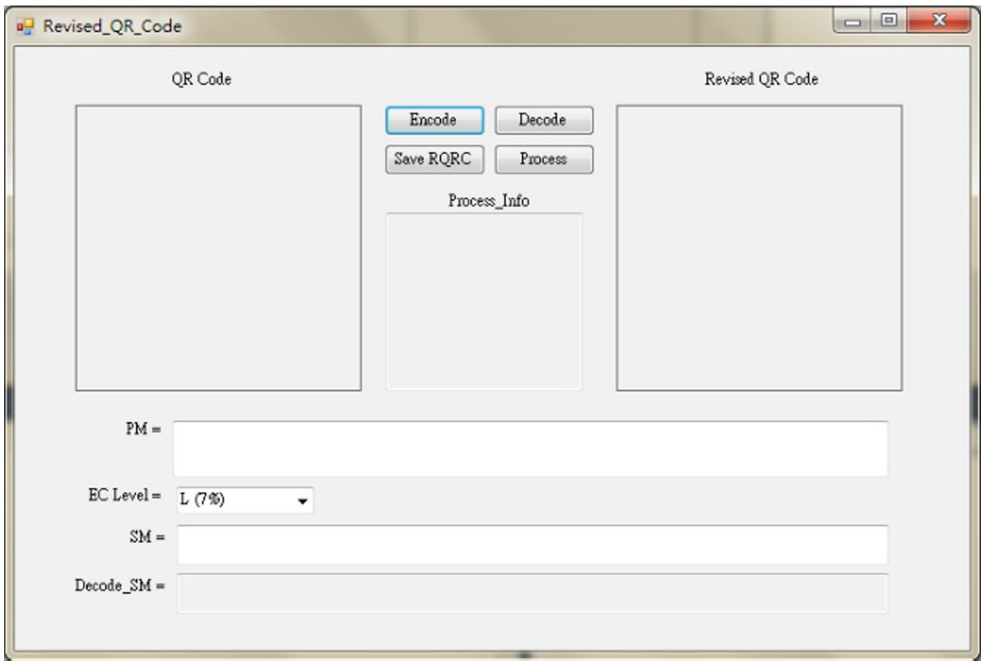

Fig. 12 Startup form of *Revised_QR_Code*

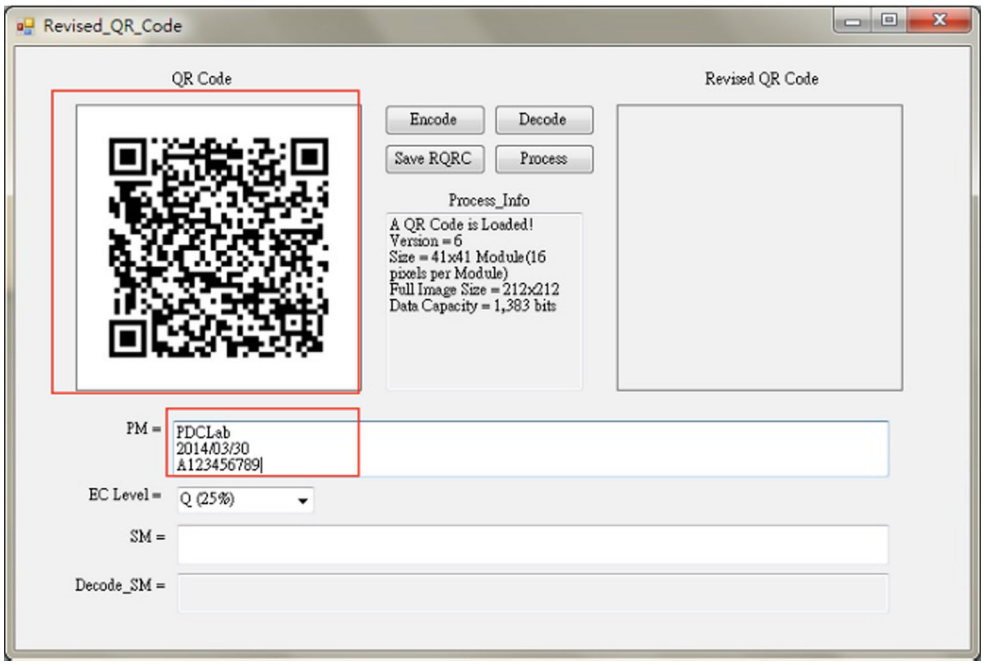

Fig. 13 Load a QR code and input *PM*

Fig. 14 Input *SM* and produce a revised QR code

where the ECC level can be selected from Table 2.

The *PM* and *SM* capacities practically observed in the minimum and maximum QR-code versions are listed in Table 5. We can see that there is inherently a trade-off between the *PM* and *SM* capacities. However, we must emphasize that all of the values of the *SM* data bits in Table 5 are theoretical maxima, and the actual *SM* capacities are generally less. This difference results from a variety of factors that influence the *SM* capacity, such as the manner in which the *SM* data are embedded, the camera's resolution, and the quality of the QR-code image, in conditions in which *PM* can be correctly decoded.

Fig. 15 Decode a QR code by a mobile device

5.2 Protocol analysis by GNY logic

In this section, we present an analysis of our protocol based on GNY logic. We write the login and authentication phase of our mutual authentication protocol in the form $P \rightarrow Q : X$, where P and Q range over the principals, and X is send-

Fig. 16 Load a revised QR code

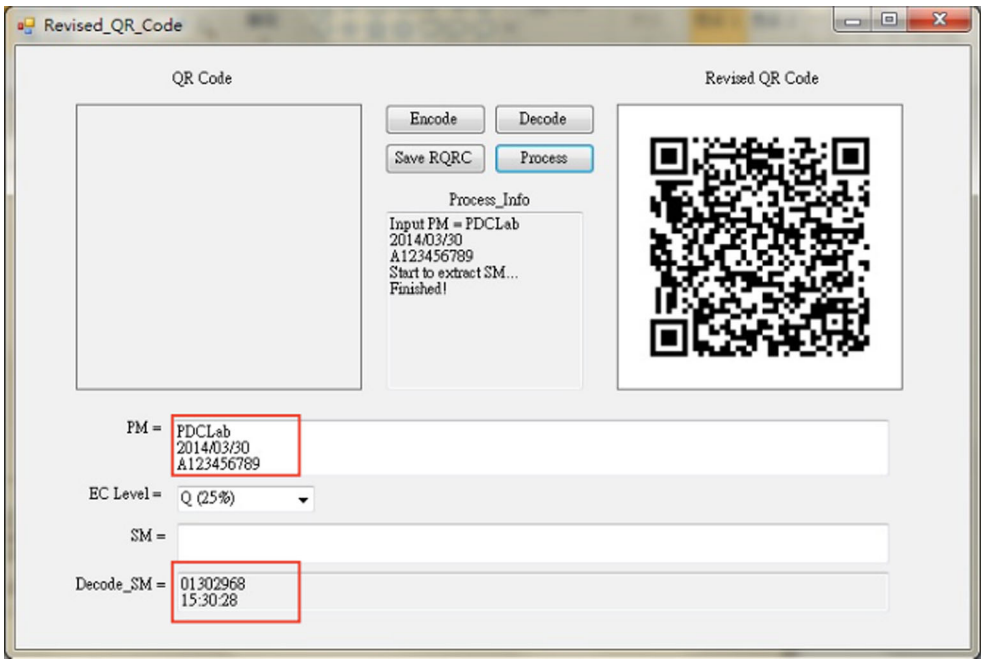

Fig. 17 Decoding result of SM

Table 5 *PM* and *SM* capacities of QR code

Version	ECC level	PM data bits	SM data bits	Total modules
1	L	152	12	441 (21×21)
	M	128	27	
	Q	104	52	
	H	72	54	
40	L	23,648	1924	31,329 (177×177)
	M	18,672	4001	
	Q	13,328	6664	
	H	10,208	7656	

ing messages from P to Q. All notations are the same as in Sect. 3.3. This results in the following five messages:

1. $SU \rightarrow IA : PID, \{PID, Ts, L, q\}+_{Ksr}$
2. $IA \rightarrow RC : \{\{PID, Ts, L, q\}+_{Ksr}, q, N_{IA}\}+_{Kir}$
3. $RC \rightarrow IA : \{\{pv, H(pv)\}+_{Ksr}, Ts, L, N_{IA}, N_{RC}\}+_{Kir}$
4. $IA \rightarrow SU : \{pv, H(pv)\}+_{Ksr}$
5. $SU \rightarrow IA : H(p+1, q)$

N_{IA} and N_{RC} are nonces for IA and RC, respectively. $^{+}K_{sr}$ and $^{+}K_{ir}$ are the public keys between SU and RC and between IA and RC, respectively. According to the above depiction, we parse the result using the following rules. For each line, the parser transforms $P \rightarrow Q : X$ into $P\triangleleft : X$ to express the meaning that P is told formula X. For each pattern Y in X, if P is told Y, which he did not convey previously in the current run, then the parser will place a star in front of Y. For each $P\triangleleft : X$, the parser adds $\sim>$ to reflect the verbal explanation of the protocol execution. The parser would produce the following output.

1. $IA\triangleleft : {}^{*}PID, {}^{*}\{PID, {}^{*}Ts, {}^{*}L, {}^{*}q\}+_{Ksr}$
2. $RC\triangleleft :^{*} \{{}^{*}\{{}^{*}PID, {}^{*}Ts, {}^{*}L, {}^{*}q\}+_{Ksr}, {}^{*}q, {}^{*}N_{IA}\}+_{Kir}$
3. $IA\triangleleft : {}^{*}\{{}^{*}\{{}^{*}pv, {}^{*}H(pv)\}+_{Ksr}, Ts, L, N_{IA}, {}^{*}N_{RC} \sim> RC \mid\equiv RC \overset{{}^{-}K_{sr},q}{\longleftrightarrow} SU\}+_{Kir} \sim> RC \mid\equiv RC \overset{{}^{-}K_{ir},q}{\longleftrightarrow} IA$
4. $SU\triangleleft : {}^{*}\{{}^{*}pv, {}^{*}H(pv)\}+_{Ksr} \sim> IA \mid\equiv IA \overset{{}^{-}K_{ir},q}{\longleftrightarrow} RC$
5. $IA\triangleleft : {}^{*}H(p+1, q) \sim> SU \mid\equiv IA \overset{p,q}{\longleftrightarrow} RC$

We begin our protocol analysis by listing the initial assumptions: $SU \ni^{-} K_{sr}; SU \ni Ts, L$
$SU \mid\equiv SU \overset{{}^{-}K_{sr}}{\longleftrightarrow} RC; SU \mid\equiv \#(Ts, L); SU \mid\equiv \phi(pv)$
$IA \ni^{-} K_{ir}; SU \ni N_{IA}$
$IA \mid\equiv IA \overset{{}^{-}K_{ir}}{\longleftrightarrow} RC; IA \mid\equiv \#(N_{IA}); IA \mid\equiv \phi(p)$
$RC \ni PID; RC \ni N_{RC}$
$RC \mid\equiv \phi(PID); RC \mid\equiv \phi(N_{RC}); RC \mid\equiv \#(N_{RC})$

According to the assumptions above, all of the principals possess secrets, and SU and IA believe that they share a private key between themselves and RC. In addition, the basis of SU and IA believing in the jurisdiction of RC is having shared secrets between them. The related assumptions are as follows:

$SU \mid\equiv RC \Rightarrow (SU \overset{q}{\leftrightarrow} IA); SU \mid\equiv RC \Rightarrow RC \mid\equiv^{*};$
$SU \mid\equiv IA \Rightarrow IA \mid\equiv^{*}$
$IA \mid\equiv RC \Rightarrow (SU \overset{q}{\leftrightarrow} IA); IA \mid\equiv RC \Rightarrow RC \mid\equiv^{*};$
$IA \mid\equiv SU \Rightarrow SU \mid\equiv^{*}$

Based on the assumptions above, SU and IA believe that RC is honest and competent. Furthermore, SU and IA believe in each other's competence and honesty. To increase trust among all of the principals, we add two assumptions, as follows:

$RC \ni {}^{-}K_{sr}; RC \ni {}^{-}K_{ir}; RC \ni q$
$RC \mid\equiv SU \overset{{}^{-}K_{sr}}{\leftrightarrow} RC; RC \mid\equiv IA \overset{{}^{-}K_{ir}}{\leftrightarrow} RC; RC \mid\equiv SU \overset{q}{\leftrightarrow} IA$

RC believes that he possesses suitable private keys with SU and IA. He also believes that q is a proper secret for SU and IA.

Now we begin to apply all of the GNY logic rules to each message. For any run of the protocol:

Message 1:
Applying T1 and P1, we obtain $IA \ni PID$. In other words, IA possesses PID.

Message 2:
Applying T1, T4, and P1, we obtain that
$RC \ni (\{\{PID, Ts, L, q\}+_{Ksr}, q, N_{IA}\}+_{Kir})$. RC possesses all of the message contents.
Applying T2, we obtain that $RC \ni q$. RC possesses the secret q.
Applying R4, we obtain that
$RC \mid\equiv \phi(\{\{PID, Ts, L, q\}+_{Ksr}, q, N_{IA}\}+_{Kir})$. RC believes that the contents of the message are recognizable.

Message 3:
Applying T1, T4, P1, P3, and P8, we obtain that $IA \ni (Ts, L, N_{IA}, N_{RC})$. *IA* possesses the contents, including Ts, L, N_{IA} *and* N_{RC}.
Applying F4, we obtain that
$IA \mid\equiv \#(\{\{pv, H(pv)\}+_{Ksr}, Ts, L, N_{IA}, N_{RC}\}+_{Kir})$. *I A* believes that the message is not a replay.
Applying R4, we obtain that
$I A \mid\equiv \phi(\{\{pv, H(pv)\}+_{Ksr}, Ts, L, N_{IA}, N_{RC}\}+_{Kir})$. *I A* believes that the contents of messages are recognizable.
Applying I2, we obtain that
$IA \mid\equiv\sim (\{\{pv, H(pv)\}+_{Ksr}, Ts, L, N_{IA}, N_{RC}\}+_{Kir})$. *I A* believes that the message originated from *RC*.
Applying J2, we obtain that $IA \mid\equiv RC \mid\equiv IA \overset{q}{\leftrightarrow} SU$. *IA* believes that *RC* believes that *q* is a proper key for *IA* and *SU*.
Message 4:
Applying T1, T4, P1, and P4, we obtain that $SU \ni (\{pv, H(pv)\}+_{Ksr})$. *SU* possesses the contents of *pv* and H(*pv*) and can verify the integrity of *pv* by calculating H(*pv*).
Applying F4 and F10, we obtain that $SU \mid\equiv \#(\{pv, H(pv)\}+_{Ksr})$. *SU* believes that the message is not a replay.
Applying R4 and R5, we obtain that $SU \mid\equiv \phi(\{pv, H(pv)\}+_{Ksr})$. *SU* believes that the message contents are recognizable.
Applying I2 and I3, we obtain that $SU \mid\equiv RC \mid\equiv IA \mid\sim (\{pv, H(pv)\}+_{Ksr})$. *SU* believes that the message originated from *RC*.
Applying J2, we obtain that $SU \mid\equiv SU \overset{q}{\leftrightarrow} IA$. *SU* believes that *IA* believes that *q*is a proper key for *SU* and *IA*.
Message 5:
Applying T5 and P4, we obtain that $IA \ni H(p+1, q)$. *IA* possesses the contents of $H(p+1, q)$. That is, he is able to verify the truth of the message from *SU*.
Applying F1, we obtain that $IA \mid\equiv \#(H(p+1, q))$. *IA* believes that the message is fresh.
Applying R5, we obtain that $IA \mid\equiv \phi(H(p+1, q))$. *IA* believes that the message contents are recognizable.
Applying I3, we obtain that $IA \mid\equiv SU \mid\sim (H(p+1, q))$. *IA* believes that the message originated from *SU*.
Applying J2, we obtain that $IA \mid\equiv IA \overset{q}{\leftrightarrow} SU$. *IA* believes that *SU* believes that *q* is a proper key for *IA* and *SU*.

In conclusion, we have analyzed all of the messages in our protocol using GNY logic. The results show that *SU*, *IA*, and *RC* possess the secret key and believe in it and that each believes that the other possesses it and believes in it under the initial assumptions. The reasoning leads to these results given that the five messages have no significant weaknesses and attain the rule targets of GNY logic.

5.3 Security analysis

In this section, we consider some possible attacks against our authentication protocol. The security risk analysis and our prevention strategies are described as follows.

- **Eavesdropping**
  The security risk of eavesdropping is very high for any authentication protocol. To prevent this risk, our protocol encrypts all of the exchanged messages using a public-key system. Even though the eavesdropper acquires *pv*, he cannot retrieve *p* because QRC_{Xi} has not been transmitted.
- **Back-shoulder attack**
  It is not feasible for an attacker to obtain authentication information through a backdoor, because our protocol does not involve typing a password. If the authentication data have been stored on a token, it must be well protected.
- **Man-in-the-middle (MITM) attack**
  Since our protocol can achieve mutual authentication between *SU* and *IA*, an MITM attack [22] is not feasible. The RC plays an important role in helping *SU* and *IA* authenticate each other, similarly to a public-key infrastructure [23]. In addition, all of the exchanged messages are encrypted using pre-shared keys, and the *pv* generation method is based on an OTP generator; hence, they both provide defenses against MITM attacks.
- **Replay attack**
  Suppose that an intruder attempts to replay a legal request between *SU* and *IA* or between *IA* and *RC* [24]. The timestamp Ts and the random numbers N_{IA} and N_{RC} will prevent a receiver from accepting this request.

6 Conclusions

In this paper, we first presented our authentication protocol in detail. Then, we described the implementation to show that our protocol is feasible. From the implementation, *PM* and *SM* can be embedded in the same QR-code image and then can be extracted properly and separately. This feature is a unique innovation. In addition, we applied formal analysis using GNY logic to analyze our protocol. The results showed that the protocol has no significant weaknesses and attains the rule targets of GNY logic. Finally, we considered possible attacks against our protocol and argued that they can be prevented. In future work, we plan to develop a complete system and to expand the applications of QR codes in daily life. Since security risks are constantly increasing and being updated, we will keep track of new types of security risks to determine whether they could cause security vulnerabilities in our protocol and devise appropriate modifications.

Acknowledgements This research was partially supported by the National Science Council of the Republic of China under the Grant MOST 105-2221-E-008-070-MY2, MOST 104-2221-E-015-001-, NSC 101-2218-E-008-003-, and the Software Research Center, National Central University, Taiwan.

References

1. Park, D., Boyd, C., Dawson, E.: Classification of authentication protocols: a practical approach. In: The 3rd International Workshop, ISW 2000 Wollongong, Australia. Lecture Notes in Computer Science, vol 1975, Springer, Berlin, pp. 194–208 (2000)
2. Kieseberg, P., Leithner, M., Mulazzani, M., Munroe, L., Schrittwieser, S., Sinha, M., Weippl, E.: QR code security. In: The 8th International Conference on Advances in Mobile Computing and Multimedia (MoMM'10), ACM-proceedings, pp. 430–435 (2010)
3. Oh, D.S., Kim, B.H., Lee, J.K.: A study on authentication system using QR code for mobile cloud computing environment. In: The 6th International Conference on FutureTech 2011, Loutraki, Greece. Communications in Computer and Information Science, vol. 184, Springer-Verlag GmbH, Berlin, pp. 500–507 (2011)
4. Liao, K.C., Lee, W.H.: A novel user authentication scheme based on QR-code. Journal of Networks **5**(8), 937–941 (2010)
5. Liao, K.C., Lee, W.H., Sung, M.H., Lin, T.C.: A one-time password scheme with QR-code based on mobile phone, IEEE-Proceedings, The 5th International Joint Conference on Networked Computing and Advanced Information Management (NCM'09), pp. 2069–2071 (2009)
6. Sahu, S.K., Gonnade, S.K.: Encryption in QR code using steganography. Int. J. Eng. Res. Appl. **3**(4), 1738–1741 (2013)
7. Chung, C.H., Chen, W.Y., Tu, C.M.: Image hidden technique using QR-barcode. In: The 5th International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP'09), IEEE-Proceedings, pp. 522–525 (2009)
8. Vongpradhip, S., Rungraungsilp, S.: QR code using invisible watermarking in frequency domain. In: 2011 9th International Conference on ICT and Knowledge Engineering, pp. 47–52 (2012)
9. Denso Wave, the Inventor of QR Code. http://www.qrcode.com/en/ (1994)
10. Chang, Y.H., Chu, C.H., Chen, M.S.: A General scheme for extracting QR code from a non-uniform background in camera phones and applications. In: IEEE-proceedings, The 9th IEEE international symposium on multimedia (ISM 2007), pp. 123–130 (2007)
11. Reed, I.S., Solomon, G.: Polynomial codes over certain finite fields. J. Soc. Ind. Appl. Math. **8**(2), 300–304 (1960)
12. Needham, R.M., Schroeder, M.D.: Using encryption for authentication in large networks of computers. Commun. ACM **21**(12), 993–999 (1978)
13. Neuman, B.C., Ts'o, T.: Kerberos: an authentication service for computer networks. IEEE Commun. Mag. **32**(9), 33–38 (1994)
14. Denning, D.E., Sacco, G.M.: Timestamps in key distribution protocols. Commun. ACM **24**(8), 533–536 (1981)
15. Lowe, G.: An attack on the Needham–Schroeder public-key authentication protocol. Inf. Process. Lett. **56**(3), 131–133 (1995)
16. Lowe, G.: Breaking and fixing the Needham–Schroeder public-key protocol using FDR. In: The 2nd International Workshop, TACAS 1996 Passau, Germany, Lecture Notes in Computer Science, vol. 1055, Springer, Berlin, pp. 147–166 (1996)
17. Formal Systems (Europe) Ltd. Failures Divergence Refinement-User Manual and Tutorial ver. 1.3 (1993)
18. Burrows, M., Abadi, M., Needham, R.M.: A logic of authentication. Proc. R. Soc. A Math. Phys. Sci. **426**(1871), 233–271 (1989)
19. Gong, L., Needham, R., Yahalom, R.: Reasoning about belief in cryptographic protocols. In: IEEE-Proceedings, Computer Society Symposium on Research in Security, pp. 234–248 (1990)
20. Ding, Y.: An improvement of GNY logic for the reflection attacks. J. Comput. Sci. Technol. **14**(6), 619–623 (2010)
21. Mathuria, A.M., Safavi-Naini, R., Nickolas, P.R.: On the automation of GNY logic. In: IEEE Computer Society Press Los Alamitos, Australian Computer Science Communications, pp. 370–379 (1995)
22. Asokan, N., Niemi, V., Nyberg, K.: Man-in-the-middle in tunnelled authentication protocols. In: The 11th International Workshop, Cambridge, UK, Lecture Notes in Computer Science, vol. 3364, Springer, Berlin, pp. 28–41 (2005)
23. Perlman, R.: An overview of PKI trust models. IEEE Netw. **13**(6), 38–43 (1999)
24. Syverson, P.: A taxonomy of replay attacks [rcryptographic protocols]. In: IEEE-Proceedings, Computer Security Foundations Workshop VII (CSFW 7), pp. 187–191 (1994)