Recommended Paper

Detecting Fake QR Codes Using Information from Error-Correction

Toshihiro Ohigashi[1,a)] Shuya Kawaguchi[1] Kai Kobayashi[1] Hayato Kimura[1] Tatsuya Suzuki[2] Daichi Okabe[1] Takuya Ishibashi[1] Hiroshi Yamamoto[1] Maki Inui[3] Ryo Miyamoto[3] Kazuyoshi Furukawa[3] Tetsuya Izu[3]

Received: November 30, 2020, Accepted: June 7, 2021

Abstract: In 2018, Takita et al. proposed a construction method of a fake QR code by adding stains to a target QR code, that probabilistically leads users to a malicious website. The construction abused the error-correction of error-correcting code used in the QR code, namely, the added stains induce decoding errors in black and white detection by a camera, so that the decoded URL leads to the malicious website. Also, the same authors proposed a detection method against such fake QR codes by comparing decoded URLs among multiple QR code readings since the decoded URLs may differ because of its probabilistic property. However, the detection method cannot work well over a few readings. Moreover, the proposed detection method does not consider the environmental or accidental changes such as sudden sunshine or reflection, nor recognizes the fake QR code as non-fake when the probability is low. This paper proposes new detection methods for such fake QR codes by analyzing information obtained from the error-correcting process. This paper also reports results from implementing the new detection methods on an Android smartphone. Results show that a combination of these detection methods works very well compared to when using only a single detection method.

Keywords: phishing, QR code, fake, detection, error-correcting code

1. Introduction

The spread of the Internet has made cyber attacks a powerful threat all over the world. Among these cyberattacks, the phishing website is popular for attackers to collect IDs and passwords [16]. In the old days, URLs of such phishing websites were directly included in emails to prompt users to access malicious sites but nowadays few users will click on such dangerous links. Then the attackers embed the URLs of the phishing websites in emails, by using HTML email, shorten URL or QR code to make such URLs invisible to users [11] so that users click these links without being aware that they are accessing a phishing website.

QR code [8] is a two-dimensional barcode developed by the DENSO Corporation in 1994, and an international standard in ISO. QR code has become popular because it can handle large amounts of data compared to previous one-dimensional barcodes, and can be easily used with camera-equipped mobile devices so that it is widely used in many applications such as cashless payment services, airline tickets and accessing websites because of its convenience.

The security threats posed by QR codes have been investigated [4]. Among these, leading to phishing websites [17] and distributing malwares [19] are realistic [2]. The naive approach is to prepare the QR code embedding the link to a malicious website, and use it to replace a legitamate QR code. In fact, a legitimate QR code was covered by a sticker with a malicious QR code on it [15] and a legitimate ticket with a legitimate QR code (issued by a police officer) was replaced by a forged ticket with a malicious QR code [5]. Stickers can be detected physically, but when QR codes are printed on paper or embedded in signboards, detecting such malicious QR codes requires further information beyond the QR code itself. Some countermeasures based on the usable security are studied [1].

A more sophisticated approach is to modify the legitimate QR code printed on a public poster into a malicous one. Kieseberg et al. proposed a construction method by changing white parts to black that deterministically leads users to malicious websites [3]. In 2018, Takita et al. proposed a construction method for a QR code that probabilistically leads users to malicious websites (fake QR code) [6], [12], [14]. The proposed construction abused the error-correction of the error-correcting code used in the QR code, namely, the construction adds stains on a target QR code that induce errors in black and white detection by a camera so that decoding errors are likely (but not always) to occur and the stained QR code may lead users to malicious websites probabilistically. This means that the stains change the luminosity of the center pixels of the modules from the original QR code.

Takita et al. also showed that the probability can be controlled

[1] Tokai University, Minato, Tokyo 108–8619, Japan
[2] University of Tsukuba, Tsukuba, Ibaraki 305–8577, Japan
[3] FUJITSU Ltd., Kawasaki, Kanagawa 211–8588, Japan
[a)] ohigashi@tsc.u-tokai.ac.jp

The preliminary version of this paper was presented at Computer Security Symposium (CSS 2020), October 2020. The paper was recommended to be submitted to the Journal of Information Processing (JIP) by the Program Chair of CSS 2020.

by changing the luminosity of the stains and that the probability can be small enough, say 0.6%, experimentally. Since the fake QR code generated by the proposed construction allows users to access legitimate (non-malicious) websites in most cases and malicious websites with a low probability, such fake QR codes will be left without being recognized as "fake". Even if a user is tricked by the fake QR code, accesses the malicious site and then reports the fake QR code, it will prove difficult to find the fake QR code among the many legitimate QR codes because of the low probability of the fake QR code actually functioning.

The proposed construction adds stains around the center pixels of modules in the target QR code, so that the stains can be recognized visually [6], [12], [14]. In order to avoid being noticed visually, the same authors also proposed to use the designed QR code [9], in which colorful images and texts are allowed, since the stains become less conspicuous in such designed QR code [7]. In addition, the same authors proposed another trick by shifting the positions of modules rather than by adding stains to the QR code in order to cause probabilistic decoding error efficiently [13].

The same authors also proposed a detection method against such fake QR codes by comparing decoded URLs obtained from multiple QR code readings from the same camera [13], [14]. If one reading is different from others, then the QR code is recognized as fake because of its probabilistic property. The increase in the processing time is small so that this detection is very convenient for users in the sense that users can avoid fake codes by themselves. However, some problems occur. The proposed detection method does not work well over a few readings. The proposed detection method does not take the environmental (accidental) change such as sudden sunshine or reflection. Especially, the proposed detection method even recognizes the fake QR code as non-fake when the probability is low. Thus, the fake code might still remain without being recognized as fake by the proposed detection method.

This paper proposes new detection methods to determine whether a QR code is fake or not by using the error-correcting process used in the QR code. Specifically, three detection methods are introduced: (1) a detection method using the number of errors (2) a detection method using the locations of errors and (3) a detection method using the symmetry and locality of errors. In addition, this paper reports results from implementing these detection methods on an Android smartphone. The results show that a combination of these detection methods works very well compared to the cases where a single detection method is used.

The rest of the paper is organized as follows: Section 2 introduces specifications of the QR code, and the detailed construction mechanism of a fake code and a previous detection method are described in Section 3. Then in Section 4, the proposed three detection methods are described while taking some considerations into account. Implementation details and experimental results are described in Section 5, and further discussions are summarized in Section 6.

2. QR Code

This section briefly describes the specifications of the QR code [8]. The QR code is a two-dimensional barcode for embedding information, and is composed of 2-dimensional cells, called modules, which represent 1-bit information for each.

2.1 Module Descriptions

Each QR code has three types of modules: main modules, information modules and supporting modules. Main modules embed the data to be sent to receivers with redundant data for error-correction. Information modules embed the format information and the version information defined in the QR code specifications. The format information (such as L (low), M, Q, H, and S (high)) specifies the error correction level used by the error-correcting code used in the QR code, namely, Reed-Solomon code (RS code), while the version information specifies the number of modules in the QR code and takes a value from 1 (small) to 40 (large). For example, version 2 allows 25×25 modules in a QR code. Supporting modules, such as positioning detection markers, alignment markings, and timing pattern, are modules with fixed patterns for adjusting the direction of the QR code and obtaining the scanning lines of the module when it is scanned. Since we are interested in the data body in this paper, we concentrate on this part in the following.

The data body consists of six individual data, *Mode Indicator* (*MI*), *Character Count Indicator* (*CCI*), *Data Sequence* (*DS*), *Terminator* (*T*), and *Padding Code* (*PC*), and the redundant data *Error Correction Word* (*ECW*) for error-correction. For example, the data body in 2-M type (with version 2 and error correction level M) QR code consists of the following six individual data:

- *MI* (4-bit): the type of data such as numeric data (0001), alphanumeric data (0010), byte data (0100), or Chinese character data (1000),
- *CCI* (8-bit): the length of *DS* in binary,
- *DS* (at most 26-byte): the body of the data with the size specified by *CCI*,
- *T*: a padding by 0 s for making *DS* as a multiple of 8 in binary,
- *PC*: a padding by repetitions of 236 and 17 if *DS* is smaller than 26-byte,
- *ECW* (16-byte): the redundant data computed by RS code from the above five individual data.

In this case, the data body is represented by *MI*||*CCI*||*DS*||*T* (if necessary)||*PC* (if necessary) ||*ECW*, where || denotes the concatenations.

Figure 1 shows locations of modules corresponding to the above data body, where D1...D28 denotes the first five individual data (in 28-byte), and E1...E16 denotes the error correction words for the 2-M type QR code. Note that each byte has eight modules as shown in the figure. Also note that black and white modules are assigned to information modules and supporting modules.

2.2 Reed-Solomon (RS) Code

The QR code uses the Reed-Solomon (RS) code to correct data, which works well against the burst error since the RS code corrects in a symbol-wise way where each symbol consists of bits. In the QR code, each byte is treated as a symbol in the RS code, and modules corresponding to a symbol are placed next to each other

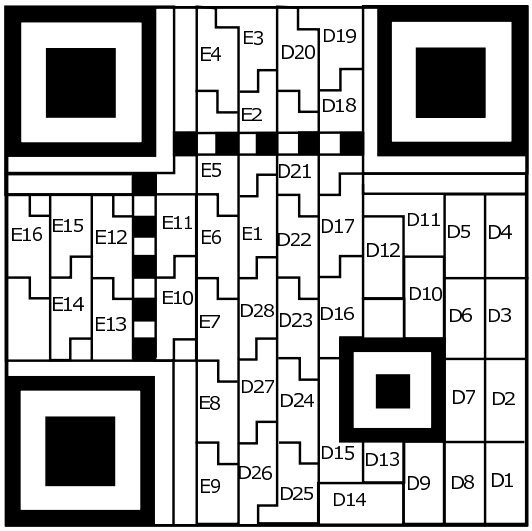


Fig. 1 Module location in a 2-M type QR code.

as in Fig. 1, so that even if multiple adjacent modules are stained by a felt-pen, for example, the errors can be corrected by error-corrections of a small number of symbols. The message length k is the number of symbols in the information part protected by the RS code, and the block length n is the number of symbols in the entire codeword including the redundant symbols added by the code. In an $[n, k]$ RS code, the number of correctable errors is computed by the design distance $d = n-k+1$ using the Euclidean decoding, namely, the number of correctable symbols is bounded by $t = \lfloor \frac{d-1}{2} \rfloor = \lfloor \frac{n-k}{2} \rfloor$.

The RS code used by the QR code is an RS code over $GF(2^8)$, a finite field with 2^8 elements, using $p(x) = x^8 + x^4 + x^3 + x^2 + 1$ as the primitive polynomial. The elements of $GF(2^8)$ are all the binary polynomials of degree 7 or less. The number of binary polynomials of degree 7 or less is 2^8 since eight coefficients of binary polynomials are 0 or 1. The list of coefficients of the 7th order, 6th order, ..., and 0th order of a polynomial is called the vector representation of the polynomial. Since the vector representation consists of a binary sequence of length 7, it is associated with 1 byte of data. Let α be a root of $p(x)$. Using the equation $\alpha^8 = \alpha^4+\alpha^3+\alpha^2+1$, all polynomials in $GF(2^8)$ are represented by $0, \alpha^0, \alpha^1, \alpha^2, \ldots, \alpha^{254}$. Here, 0 (zero) represents the polynomial 0 (vector representation is 00000000), α^0 represents the polynomial 1 (00000001), α^1 represents the polynomial α (00000010), ..., α^7 represents the polynomial α^7 (10000000), α^8 represents the polynomial $\alpha^4 + \alpha^3 + \alpha^2 + 1$ (00011101), and so on.

Encoding is done by using a generator polynomial. The generator polynomial is given by the following formula;

$$g(x) = \prod_{i=0}^{2t-1}(x - \alpha^i). \tag{1}$$

When encoding k message symbols $(m_0, m_1, \ldots, m_{k-1})$, where the message symbols are in $GF(2^8)$, the message polynomial is the following;

$$c(x) = m(x)x^{n-k} + (m(x)x^{n-k} \mod g(x)). \tag{2}$$

Here, $c(x)$ is a polynomial of degree $n-1$, and the series of n coefficients is the codeword.

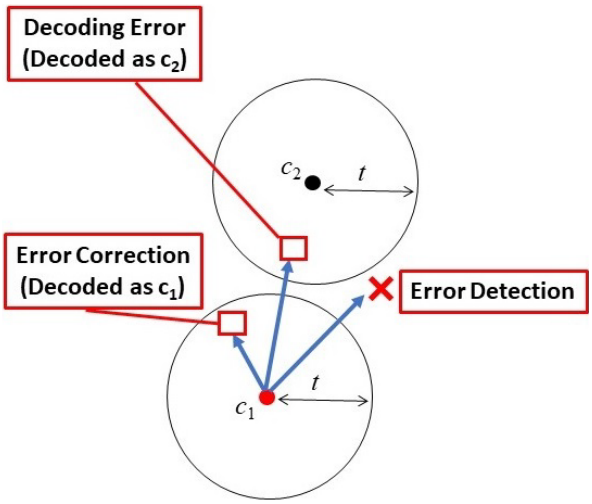


Fig. 2 Error correction, error detection, and decoding error.

The Euclidean decoding is generally used for decoders of QR code.

For a 2-M type QR code, the message is $MI||CCI||DS||T$ (if necessary)$||PC$ (if necessary) and the entire codeword is the concatenation of ECW to the above. We have $n = 44$ and $k = 28$ so that $t = 8$ is obtained from the above discussion. In a QR code with t error-correction capability, up to t errors can be corrected, while only error detection or decoded error will occur with more than t error symbols (see **Fig. 2**). More precisely, if there are more than t symbol errors, a word will be decoded as a wrong symbol if such a codeword exists. If no codeword exists, only error detection is possible.

2.3 Example

Let us represent a legitimate URL http://www.u-tokai.ac.jp/ in a 2-M type QR code. The length of the data is 25, which is smaller than 26, so that *Padding Code* (*PC*) is necessary and the number of the message symbols is 28. Individual data in the data body are as follows:

- *MI* (4-bit): 0100 (byte data)
- *CCI* (8-bit): 0001 1001 (25 in decimal)
- *DS* (25-byte): http://www.u-tokai.ac.jp/
- *T* (4-bit): 0000
- *PC* (1-byte): 236

16-byte *ECW* are computed from the above message symbols by RS code as described in the previous section. Thus we have the final encoded data as in **Table 1**, and the corresponding QR code as in **Fig. 3**.

3. Constructing Fake QR Codes

This section describes how to construct a fake QR code that can probabilistically lead users to malicious websites proposed by Takita et al. [6], [12], [14].

3.1 Principle

Let c_1 be a codeword for the QR code corresponding to a legitimate URL, c_2 be a codeword for the QR code corresponding to a malicious URL, and $d(c_1, c_2)$ be the distance (the number of different symbols) between these codewords.

Table 1 QR code example.

MI	*CCI*		*DS*												
	25		h		t		t		p		:		/		/
0100	0001	1001	0110	1000	0111	0100	0111	0100	0111	0000	0011	1010	0010	1111	0010
65		150	135		71		71		3		162		242		
DS															
/	w		w		w		.		u		-		t		o
1111	0111	0111	0111	0111	0111	0111	0010	1110	0111	0101	0010	1101	0111	0100	0110
247		119	119		114		231		82		215		70		
DS															
o	k		a		i		.		a		c		.		j
1111	0110	1011	0110	0001	0110	1001	0010	1110	0110	0001	0110	0011	0010	1110	0110
246		182	22		146		230		22		50		230		
DS					*T*	*PC*									
j	p		/												
1010	0111	0000	0010	1111	0000	1110	0100								
167		2	240			236									
ECW															
36	111	113	168	84	82	21	223	143	148	4	29	86	247	145	151

Fig. 3 Legitimate QR code.

Fig. 4 Fake QR code.

Firstly, we construct a new sequence c_1' with $d(c_1', c_2) = d(c_1, c_2) - a$ for a pre-determined parameter a ($\leq t$) by adding a errors to the sequence c_1. Though the error tolerance of the QR code corresponding to the sequence c_1' becomes low, the sequence c_1' will be decoded as c_1 since a is smaller than t, so that, as QR code, embedded URL will be decoded successfully if there are no other environmental errors. Furthermore, we construct another new sequence c_1'' with $d(c_1'', c_2) = d(c_1', c_2) - b$ for a pre-determined parameter b satisfying $d(c_1, c_2) - t \leq a + b \leq d(c_1, c_2)$, where a fake QR code is constructed by the sequence c_1''. Then, the sequence c_1'' will be decoded as c_2 by error correction (decoding error), and thus, as QR code, embedded malicious website will be provided to the user probabilistically. **Figure 5** shows relations among these sequences. With this approach, users can be led to a legitimate website in some cases, and to a malicious website in other cases.

Takita et al. add a errors by reversing the black and white of the corresponding modules deterministically, and b errors by changing the luminosity of the center pixel in a module in order to make errors probabilistically [14]. In particular, Takita et al. provide an experimental result in which the relation between the error probability and the degree of the luminosity is shown. The probability of the error is 65.8% for the luminosity 0, while the probability is 0.6% for the luminosity 160, where 255 corresponds to white. Since the error probability depends on not only the luminosity but the medium on which the QR code is printed or displayed, the brightness of the surroundings, and the performance of the camera, the result does not hold for all circumstances. However, the fact that the error probability may be low makes it difficult for the user to detect such fake QR codes.

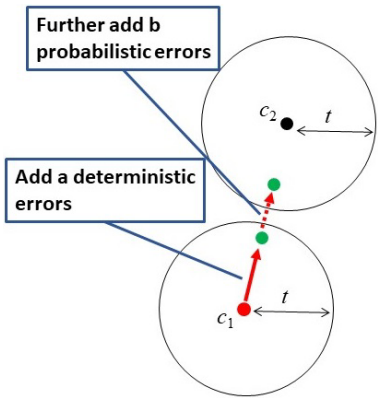


Fig. 5 Fake QR code principle.

3.2 Example of a Fake QR Code

As an example, we embed a malicious URL http://www.u-yokai.ac.jp/ into a QR code corresponding to a legitimate URL http://www.u-tokai.ac.jp/. As introduced in Section 2, the corresponding sequence for the legitimate URL is

$$\begin{aligned} c_1 = (&65, 150, 135, 71, 71, 3, 162, 242,\\ &247, 119, 119, 114, 231, 82, 215, 70,\\ &246, 182, 22, 146, 230, 22, 50, 230,\\ &167, 2, 240, 236,\\ &36, 111, 113, 168, 84, 82, 21, 223,\\ &143, 148, 4, 29, 86, 247, 145, 151), \end{aligned}$$

where the last row is the redundant data for the error-correction. Similarly, the sequence corresponding to the malicious website is

$$c_2 = (65, 150, 135, 71, 71, 3, 162, 242,$$

247, 119, 119, 114, 231, 82, 215, **150**,
246, 182, 22, 146, 230, 22, 50, 230,
167, 2, 240, 236,
207, **252**, **132**, **239**, **96**, **205**, **117**, **61**,
38, **171**, **42**, **232**, **175**, **206**, **111**, **215**),

where **bold** symbols are different from c_1 (17 symbols). Since 1 letter is different in the URLs and thus all 16 error-correction symbols are changed, the distance between c_1 and c_2 is $d(c_1, c_2) = 1 + 16 = 17$, which coincides with the design distance $d = n - k + 1 = 28 - 12 + 1 = 17$, namely the distance is minimum as RS code, so that this sequence pair is likely to provide decoding errors.

Next, we determine locations to set a deterministic errors and b probabilistic errors in order to construct a fake QR code. In the following, we use $a = 8$ and $b = 1$. Since we need to change more modules when the Hamming distance between the corresponding symbols is large, we want to find a symbol pair between c_1 and c_2 whose Hamming distance is minimum (> 0). Because of this condition, we set deterministic errors on E1 to E8, while the probabilistic error on E16 (whose Hamming distance is 1). In fact, E16 of c_1 is 151 = (10010111) and E16 of c_2 is 215 = (11010111). Thus, we set X = (1?010111) for E16 of c_2, where ? denotes the probabilistic bit. Then the sequence c_1'' for the fake QR code is given as shown below.

c_1'' = (65, 150, 135, 71, 71, 3, 162, 242,
247, 119, 119, 114, 231, 82, 215, $\underline{70}$,
246, 182, 22, 146, 230, 22, 50, 230,
167, 2, 240, 236,
$\underline{207}$, $\underline{252}$, $\underline{132}$, $\underline{239}$, $\underline{96}$, $\underline{205}$, $\underline{117}$, $\underline{61}$,
143, 148, 4, 29, 86, 247, 145, $\underline{X}$)

Note that the underlined symbols are different from c_1.

By then encoding this sequence into the QR code, we obtain the false QR code as shown in the example in **Fig. 4**. Here, the probabilistic error is provided by changing the luminosity of a module, namely, a pixel with the given luminosity is placed at the center of a white module so that this (almost) white module (0) can be misidentified as black (1) probabilistically.

In the above example, only one probabilistic error was added in order to occur the decoding error surely. When more probabilistic errors are added, the generated sequence is likely to be out of the sphere with center c_1 or c_2, and the decoding error may not work. However, since some QR code readers are designed to retry when the error is detected, it is possible to construct a fake QR code even if probabilistic errors are added to multiple symbols. On the other hand, more reading time will be needed.

3.3 Detection Methods by Takita et al.

A naive countermeasure for fake QR codes is to erase recognized stains by image processing. However, this approach cannot counter other tricks such as shifting module positions rather than adding stains to the QR code as proposed in Ref. [13]. In addition, since this cannot distinguish whether the stains came from the fake QR code or other reasons, we cannot detect the fake QR code.

Takita et al. also proposed detection methods for a fake QR code [13], [14]. The first detection method utilizes additional information in addition to the QR code. For example, when a QR code is printed on a brochure, the embedded URL will be also printed in letters. Takita et al. proposed comparing the decoded URL and the printed URL, where the fake code can be detected manually. This may work to a certain degree but is a very limited method. Especially, the detection will not be possible when no additional information is provided.

Another detection method is scanning the QR code multiple times and jump to the link when all decoded URLs are the same. This works better than the above detection method and is convenient for users. Still, some problems remain. The first problem is that the detection does not work well for a few readings. Since the error probability can be controlled and can be 10%, for example, at least 10 scans are required to detect the fake QR code. Secondly, the detection is affected by environmental factors. The detection method is susceptible to an environmental change such as weather, sunshine, or change of the angle. The last and the greatest problem is that if the fake QR code is not detected by the detection method, which is likely to occur especially when the error probability is low, then the fake QR code will remain.

4. New Detection Methods

As introduced in Section 3.3, previous detection methods proposed by Takita et al. have limitations. Since the construction of the fake code is artificial, the locations of corrected symbols have common characteristics. In this section, we propose new detection methods against fake QR codes by using information obtained from error-correction. More precisely, we propose three detection methods: (1) a detection method using the number of errors (2) a detection method using the locations of errors and (3) a detection method using the symmetry and locality of errors. In addition, we propose a combined detection method of these detections.

4.1 Detection Method Using the Number of Errors

The decoding process of the RS code used in the QR code consists of the following steps:

Step 1 Compute the syndrome from the received codeword,
Step 2 Compute the error-polynomial from the syndrome,
Step 3 Compute the error locations from the error-polynomial,
Step 4 Compute the number of corrected symbols from the error evaluating polynomial,
Step 5 Output the decoded result with error-corrections,

so that the number of the corrected symbols (errors) is obtained from Step 4. Therefore, this number can be used for the detection against the fake QR code.

In the fake QR code construction introduced in Section 3, deterministic and probabilistic errors are added to the codeword corresponding to the legitimate URL. In order to make the decoding error occur, the number of the probabilistic errors b is chosen to be small, while the number of the deterministic errors a is large. Thus the number of corrected symbols equals or is larger than a

when the fake QR code is scanned, which can be used to distinguish between legitimate or fake QR codes. One way is to prepare a threshold a' in advance, and make a warning when the number of corrected symbols in the decoding process is larger than a' in order to detect the fake QR code. This detection method will be suitable for cases where the environmental errors are unlikely to occur such as in the daytime, with little stains and tears on the QR code.

When the number of deterministic errors a is small and the probabilistic errors b is large, the above approach does not work well. In this case, another threshold works as well. For b probabilistic errors, denote the each error probability as $p_1, p_2, \ldots, p_b$. Assume that a deterministic errors and b probabilistic errors are added to the legitimate sequence and $a + b$ errors are required to make the detection error occur. The probabilty of making the decoding error occur or namely leading the user to a malicious website is $\prod_{i=1}^{b} p_i$, which is much smaller than each of $p_1, p_2, \ldots, p_b$ since they all are probabilistic variables. For example, when $b = 4$, and each error probability is set to be 0.2, the probability of occurring the decoding error is $0.2^4 = 0.0016$. In such cases, the number of corrected symbols is likely to be different in each scan, which can be used as another threshold for detecting the fake QR code. In fact, in the above setting, the number of errors can be $a + 0$ with a probability of 0.4096, $a + 1$ with 0.4096, $a + 2$ with 0.1536, $a + 3$ with 0.0256, and $a + 4$ with 0.0016, so that the number of errors will vary from $a + 0$ to $a + 4$ in each scan. Thus, the detection method based on the number of corrected symbols is valid even for a smaller a.

4.2 Detection Method Using Locations of Errors

As described in the previous subsection, not only the number of corrected symbols, but the locations of corrected symbols (errors) can be obtained from the error-correcting process. Therefore, it is possible to use the location information for detecting fake QR codes as well.

Let us observe the sample fake QR code described in Section 3.2. When the fake QR code is scanned, and if the probabilistic error does not occur, the legitimate sequence c_1 will be decoded from the fake sequence c_2. This is because error symbols are corrected from other symbols of namely the underlined symbols in c_1''. If the probabilistic error does occur, the malicious sequence c_1' will be decoded from the fake sequence c_2, by correcting error symbols by other symbols, namely, bold symbols in c_2 but not underlined in c_1''. In any cases, most error symbols are in the error-correction part. In fact, this is one of the distinguishing features of the fake QR codes. In the construction of the fake QR codes by Takita et al., the difference between a legitimate URL and a malicious URL is set to be small in order to make the fake URL hard to notice. Thus the error symbols are generally gathered in the error-correction part.

For detecting the fake QR codes, we count the numbers of corrected symbols in the error-correction part and in other parts, and compare with a threshold (determined in advance) to decide whether the scanned QR code is fake or not.

Note that, in practice, since the error-correction modules are located together in a QR code, there will be a case when natural errors are gathered to the error-correction part, for example, a burst error on the error-correcting modules by a felt pen. The above detection method might possibly recognize this natural error as a fake. However, in such cases, the ratio of errors in the error-correction part will be very high so that treating a too high ratio case as non-fake by another threshold will work.

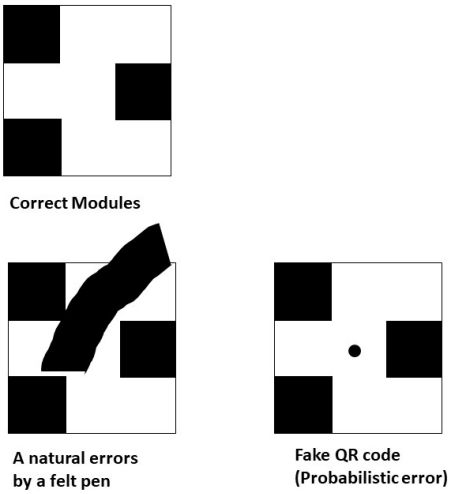


Fig. 6 Difference of errors.

4.3 Detection Method Using the Symmetry and Locality of Errors

When errors occur on a QR code printed on a sheet of paper, there may be a series of errors, such as a specific region is stained by a felt pen, a part of the paper is torn, a finger is captured when scanned (see **Fig. 6**, for example). On the other hand, in the construction of the fake QR code, locations of errors are generally scattered. The difference between such natural and artificial error patterns can be used as a detection of fake QR codes. Especially, in the natural case, all erroneous modules are likely to be all black (1) or white (0) because of its nature, which can be used as a detection method against the fake QR code.

We demonstrate a typical example of a detection method using the error pattern. Here, $E1$ of the fake QR code in Section 3.2 is $207 = (11001111)_2$. If the Fake QR code is decoded to the original QR code, then the $E1$ is decoded from 207 to $36 = (00100100)_2$. Then, the modules corresponding to two of the most significant bits are changed from black (1) to white (0), and the modules corresponding to the 3rd, 5th, 7th, and 8th are changed from white (0) to black (1). The mixed error patterns are not natural error patterns for a QR code printed on a sheet of paper. Similarly, when the fake QR code is decoded to the QR code of the malicious URL, then the $E1$ is decoded from $243 = (10001111)_2$ to $38 = (00100110)_2$. This case also occurs in the mixed error patterns. Therefore, as a detection against the fake QR codes, we count the numbers of symbols in which the mixed error patterns have occurred, and compare with a threshold (determined in advance) to decide whether the scanned QR code is fake or not.

5. Implementation

This section describes implementation details and results of

the proposed detection methods. We actually implemented a QR code reader application on Android with ZXing library [18], which is an open-source, multi-format 1D/2D barcode image processing library.

5.1 Implementation Details

5.1.1 Detection Method Using the Number of Errors

This detection method uses the number of corrected symbols (errors) which has appeared in the error-correction process in the RS code or namely the fake QR code is detected if the number of corrected symbols is larger than a threshold. This threshold is bounded by the number of correctable symbols t, and as introduced in Section 2, t is computed by the number of entire symbols n and the number of symbols in the information part k by $t = \lfloor \frac{n-k}{2} \rfloor$. Since n, k are embedded in the QR code, the values of these parameters are obtained and computed easily. Implementation results are as follows: note that we use the ratio r_1, the number of corrected symbols over t, is used as a threshold for simplicity.

1. Set a value of the threshold T_1 from 0 to 1.
2. Obtain n and k from the QR code.
3. Compute t from n and k.
4. Obtain the number of corrected symbols n_1 from the error-correction process.
5. Compute the ratio $r_1 = n_1/t$.
6. Output "fake" if $r_1 > T_1$ and "non-fake" otherwise.

5.1.2 Detection Method Using Locations of Errors

This detection method uses locations of corrected symbols (errors) which have appeared in the error-correction process in the RS code. In other words, the method detects the fake QR code if the number of corrected symbols in the error-correction part is larger than a threshold, which is bounded by $n - k$. Similar to the previous case, values of these parameters are obtained and computed easily. Implementation results are as follows: note that we use the ratio r_2, the number of corrected symbols in error-correction part over the entire number of corrected symbols n_1, is used as a threshold for simplicity.

1. Set a value of the threshold T_2 from 0 to 1.
2. Obtain the entire number of corrected symbols n_1 and the number of corrected symbols in error-correction part n_2 from the error-correction process.
3. Compute the ratio $r_2 = n_2/n_1$.
4. Output "fake" if $r_2 > T_2$ and "non-fake" otherwise.

In order to obtain the number of corrected symbols and their locations, we modify the `decode` function in the `ReedSolomonDecoder` class provided by the ZXing library (written in Java language). We show the modified `decode` function in the below. The number of entire corrected symbols is obtained by `sigma.getDegree()` in line 29, and locations of corrected symbols are obtained in line 37.

```
1  public void decode(int[] received, int twoS) throws ReedSolomonException {
2    MyGenericGFPoly poly = new MyGenericGFPoly(field, received);
3    int[] syndromeCoefficients = new int[twoS];
4    boolean noError = true;
5    for (int i = 0; i < twoS; i++) {
6      int eval = poly.evaluateAt(field.exp(i + field.getGeneratorBase()));
7      syndromeCoefficients[syndromeCoefficients.length - 1 - i] = eval;
8      if (eval != 0) {
9        noError = false;
10     }
11   }
12   byte[] rawByte = new byte[received.length];
13   for (int i = 0; i < received.length; i++){
14     rawByte[i] = (byte) received[i];
15   }
16   if (noError) {
17     MyDataClass.setErrorCount(0);
18     return;
19   }
20   MyGenericGFPoly syndrome = new MyGenericGFPoly(field,
           syndromeCoefficients);
21   MyGenericGFPoly[] sigmaOmega =
22           runEuclideanAlgorithm(field.buildMonomial(twoS, 1), syndrome,
                 twoS);
23   MyGenericGFPoly sigma = sigmaOmega[0];
24   MyGenericGFPoly omega = sigmaOmega[1];
25   int[] errorLocations = finderrorLocations(sigma);
26   int[] errorMagnitudes = findErrorMagnitudes(omega, errorLocations);
27   int[] errorPositions = new int[errorLocations.length];
28
29   int errorCount = sigma.getDegree();
30   MyDataClass.setErrorCount(errorCount);
31
32   for (int i = 0; i < errorLocations.length; i++) {
33     int position = received.length - 1 - field.log(errorLocations[i]);
34     if (position < 0) {
35       throw new ReedSolomonException("Bad error location");
36     }
37     errorPositions[i] = position;
38     received[position] = MyGenericGF.addOrSubtract(received[position],
             errorMagnitudes[i]);
39   }
40   for (int i = 0; i < received.length; i++){
41     rawByte[i] = (byte) received[i];
42   }
43   Arrays.sort(errorPositions);
44   MyDataClass.setErrorPositions(errorPositions);
45 }
```

5.1.3 Detection Method Using the Symmetry and Locality of Errors

This detection method uses the symmetry and locality of corrected symbols (errors) which have appeared in the error-correction process in the RS code. In other words, the fake QR code is detected if the number of symbols which has occurred the mixed error patterns is larger than a threshold. Implementation results are as follows: note that we use the ratio r_3, the number of symbols occurred the mixed error patterns over the entire number of corrected symbols n_1 is used as a threshold for simplicity.

1. Set a value of the threshold T_3 from 0 to 1.
2. Obtain the number of entire corrected symbols n_1 and their locations.
3. Obtain a decoded cordword of the QR code by using the error-correction process.
4. Compare the decoded codeword with the sequence obtained from a camera, and count the number of symbols occurred the mixed error patterns n_3.
5. Compute the ratio $r_3 = n_3/n_1$.
6. Output "fake" if $r_3 > T_3$ and "non-fake" otherwise.

5.1.4 Combined Detection Method

Each of the proposed detection methods can potentiall detect the fake QR code. However, it may fail depending on the stains. For example, when a felt pen gives stains on a legitimate QR code, the detection method using the number of errors may judge it as the fake. In order to avoid such false positives, we also implement a combination of the detection methods. In fact, the combined detection method judges as a fake if all three detection methods judge as a fake. This combined detection method will likely avoid occurrence of false positives.

5.2 Discussions

In this subsection, we discuss the limitations of the proposed detection methods and possible workarounds.

5.2.1 Fake QR Code based on Design QR Code

The "design QR code" is a variation of the QR code with illustrations, letters or logos (QRgraphy is one of the examples of the design QR code [10]). Since such irregular data are recognized

Table 2 Device Specifications.

Device Name	HUAWEI nova lite 2
OS	EMUI9.1.0 (Build 9.1.0.205 (C635E6R1P5)) (Original OS based on Android 9.0)
CPU	Hisilicon Kirin 659
Memory	3.0 GB

Table 3 Conditions for Experiments.

Illuminance	69 lux
Distance between target QR code with the camera	30 cm
Size of target QR code	$12 \times 12\,\text{cm}^2$

as errors in a QR code, a certain number of errors will occur even when the code is not faked. Therefore, when countering such fake QR codes it may seem difficult to distinguish whether the QR code is fake or not by using the number of errors. However, in this case, the detection method by the error locations is effective.

5.2.2 Changing Error Locations and Countermeasures

In order to avoid the fake QR code detection method using the error locations, we consider changing the locations of errors from the error-correction part to other parts in a QR code.

Firstly, we consider a method to use the padding part. The padding is used to fill in the data with a specific pattern when the length of the data is not sufficient. According to the specifications of the QR code, 236 and 17 are used to fill in the data. Since these data are included in the data body of the QR code, and can be corrected by the RS code, these data can be used instead of the error-correction part to add errors in order to decrease the ratio of the error-correction part in errors. However, such additions can be detected easily by checking the padding part since it must be a repetition of 236 and 17 in the format.

Another trick is to use the parameter part of the GET method in a target URL. For example, when the legitimate website is http://www.u-tokai.ac.jp/?ABCDEF, it is natural that an adversary set a malicious website as http://www.u-yokai.ac.jp/?ABCDEF. However, in the GET method, if such a parameter is invalid, it is automatically moved to a home page such as http://www.u-yokai.ac.jp/ so that the adversary can change the parameter part in the fake QR code. In other words, the adversary can choose a sequence with suitable error numbers and locations to deceive the above detection methods. Alternatively, the HTML anchor # can be used as such a trick. Currently, it appears difficult to detect such intelligent tricks.

5.3 Experimental Results

In order to show the effectiveness of our detection methods and the implementation, we conducted three experiments by using an Android device where our implementation is installed (see **Table 2** for detail specifications). Note that we used ZXing 3.4.0 for the implementation.

In the following experiments, we print QR codes/Fake QR codes on paper, and scan the QR codes by the camera of a smartphone from the front with the conditions in **Table 3**.

5.3.1 Experiment 1

In the first experiment, we prepare 10 fake QR codes in 2-M type whose legitimate and malicious URLs (lead by the fake QR codes) are shown in **Table 4**. Then, we scan the fake QR codes and collect the decoded URL, the ratio values r_1, r_2, r_3 in order to evaluate the thresholds. **Table 5** summarizes the collected results. As in this table, when we set the threshold values as $T_1 = 0.8$, $T_2 = 0.8$, $T_3 = 0.4$, we can say that all detection methods for all fake QR codes detect the fake successfully independent from the decoded URLs.

Table 4 List of pairs of legitimate and malicious URLs for Fake QR codes.

No.	Legitimate URL	Malicious URL
1	`http://www.u-tokai.ac.jp/`	`http://www.u-yokai.ac.jp/`
2	`http://www.tokai-adm.jp/`	`http://www.yokai-adm.jp/`
3	`http://www.fujitsu.com/`	`http://www.fukitsu.com/`
4	`http://www.tsukuba.ac.jp/`	`http://www.tsukubi.ac.jp/`
5	`http://www.kobe-u.ac.jp/`	`http://www.kabe-u.ac.jp/`
6	`http://www.google.com/`	`http://www.googlo.com/`
7	`http://www.yahoo.co.jp/`	`http://www.yagoo.co.jp/`
8	`http://www.goo.ne.jp/`	`http://www.hoo.ne.jp/`
9	`http://www.nifty.com/`	`http://www.nefty.com/`
10	`http://zoom.us/`	`http://zaom.us/`

Table 5 Experimental results of detecting Fake QR codes made by Table 4.

	Legitimate URL is decoded			Malicious URL is decoded		
No.	r_1	r_2	r_3	r_1	r_2	r_3
1	1.00	0.88	0.88	1.00	0.88	0.75
2	1.00	0.88	0.75	1.00	0.88	0.63
3	1.00	0.88	0.75	1.00	0.88	0.75
4	1.00	0.88	0.88	1.00	0.88	0.75
5	1.00	0.88	1.00	1.00	0.88	0.75
6	1.00	0.88	0.63	1.00	0.88	0.63
7	1.00	0.88	0.88	1.00	0.88	0.88
8	1.00	0.88	0.88	1.00	0.88	0.63
9	1.00	0.88	0.88	1.00	0.88	1.00
10	1.00	0.88	0.88	1.00	0.88	0.63

Table 6 Processing Times for Scanning QR Codes.

	Original	Our Implementation
Legitimate QR Code	40.8 msec	42.2 msec
Fake QR Code	44.4 msec	49.8 msec

5.3.2 Experiment 2

In order to evaluate the processing overhead, we measure the processing times of the reading by the original ZXing and by our implementation with the modified ZXing. The two targets of this experiment are the legitimate QR code (Fig. 3) and the fake QR code (Fig. 4). We scanned each QR code 10 times and the averaged times are summarized in **Table 6**. As seen in this table, the overhead of our methods is sufficiently smaller than the processing of the original ZXing.

5.3.3 Experiment 3

In this experiment, we add stains to the legitimate QR code artificially. We used the following four QR codes in 2-M type for evaluating the implementation. The first QR code is the legitimate QR code (Fig. 3). The other three QR codes are legitimate with three patterns of stains. Pattern 1 (**Fig. 7**) has small stains in black only, while Pattern 2 has small stains in black and white (**Fig. 8**). Pattern 3 (**Fig. 9**) has stains in black and white on the error-correction part, whose number of errors are the maximum number that the RS code can correct. Patterns 1 and 2 are natural stains rather than artificial. The characteristics of Pattern 3 are set similar to that of the fake QR code so that it seems hard to distinguish fake or not.

- Legitimate QR code (Fig. 3).
- Legitimate QR code with stains: Pattern 1 (Fig. 7).

Fig. 7 Legitimate QR code (with small black stains).

Fig. 8 Legitimate QR code (with small black and white stains).

Fig. 9 Legitimate QR code (with black and white correctable stains).

Table 7 Experimental Results ($T_1 = 0.8$, $T_2 = 0.8$, $T_3 = 0.4$).

	r_1	r_2	r_3	Combination
Legitimate (Fig. 3)	0.00	0.00	0.00	Legitimate
Legitimate (pattern 1, Fig. 7)	0.50	0.25	0.00	Legitimate
Legitimate (pattern 2, Fig. 8)	**1.00**	0.13	0.13	Legitimate
Legitimate (pattern 3, Fig. 9)	**1.00**	**1.00**	**0.75**	**Fake**

- Legitimate QR code with stains: Pattern 2 (Fig. 8).
- Legitimate QR code with stains: Pattern 3 (Fig. 9).

Table 7 shows the results of our experiments, where the second column denotes the ratio $r_1 = n_1/t$, for the number of errors n_1 and the number of correctable symbols t, the third denotes the ratio $r_2 = n_2/n_1$, for the number of errors in error-correction part n_2 and the above n_1, the fourth denotes the ratio $r_3 = n_3/n_1$, for the numbers of symbols occurred the mixed error patterns n_3 and the above n_1, and the last column denotes the output of the combinatorial approach. **Bold** values are beyond the thresholds so that the QR codes are judged as fake. Here, we use values $T_1 = 0.8$, $T_2 = 0.8$, $T_3 = 0.4$ for the thresholds.

According to the experimental results, the fake QR code was detected by all three detection methods and thus by the combination of methods. On the other hand, Pattern 2 was misdetected by the number of errors method, and Pattern 3 was misdetected by all methods. For Pattern 2, the other detection methods work well so the combination of 3 detection methods is effective. However, for Pattern 3, it seems hard to avoid the misdetection in the current approach.

6. Concluding Remarks

This paper proposed three detection methods for fake QR codes and a combination of these detection methods. Specifically, by observing the information obtained from the error-correction, we proposed the use of the number of corrected symbols (errors), locations of corrected symbols (errors), the symmetry and the locality of errors (error patterns), and their combination. As far as we were able to observe, our detection methods performed well in the experiments. However, the issue remains that none of the proposed detection methods correctly detected certain stained QR code. Further investigation is required to clarify the performance of our detection methods.

In this paper, we only discuss the case when providing probabilistic errors by adding a stain at the center of a module with different luminosity values. We believe that both the detection method using the number of errors and the detection method using the error locations are valid when probabilistic errors are provided by shifting the position of the module.

Research Ethics and Disclosure

Since this relates to attacks against the QR code which is widely used in the real world, we discuss the impact of our research on services and products that utilize the QR code. The construction method of fake QR codes has been published in Refs. [6], [12], [14], and the detection method for the attack, which directs users to the malicious website, was given in Refs. [13], [14]. The purpose of this paper is to provide advanced security against fake QR codes. Namely, the proposed detection methods determine whether a QR code is fake or not in order to remove potential threats from fake QR codes. We therefore believe that none of the contributions provided by this paper pose additional threats to currently used QR codes or services and associated products.

References

[1] Krombholz, K., Frühwirt, P., Kieseberg, P., Kapsalis, I., Huber, M. and Weippl, E.: QR Code Security: A Survey of Attacks and Challenges for Usable Security, *Proc. 2nd International Conference on Human Aspects of Information Security, Privacy, and Trust* (*HAS 2014*), Lecture Notes in Computer Science, Vol.8533, pp.79–90, Springer (2014).

[2] Kharraz, A., Kirda, E., Robertson, W., Balzarotti, D. and Francillon, A.: Optical Delusions: A Study of Malicious QR Codes in the Wild, *Proc. 44th IEEE/IFIP International Conference on Dependable Systems and Networks* (*DSN 2014*), pp.192–203, IEEE (2014).

[3] Kieseberg, P., Schrittwieser, S., Leithner, M., Mulazzani, M., Weippl, E., Munroe, L. and Sinha, M.: Malicious Pixels Using QR Codes as Attack Vector" in Trustworthy Ubiquitous Computing, Khalil, I. and Mantoro, T. (Eds.), pp.21–38, *Atlantis Ambient and Pervasive Intelligence*, Vol.6, Atlantis Press (2012).

[4] Kieseberg, P., Leithner, M., Mulazzani, M., Munroe, L., Schrittwieser, S., Sinha, M. and Weippl, E.: QR Code Security, *Proc. 8th International Conference on Advances in Mobile Computing and Multimedia* (*MoMM 2010*), pp.430–435, ACM (2010).

[5] Nikkei Asia: QR code scams strike China, from merchants to traffic tickets, available from ⟨https://asia.nikkei.com/Economy/QR-code-scams-strike-China-from-merchants-to-traffic-tickets⟩ (accessed 2021-03-28).

[6] Okuma, H., Takita, M. and Morii, M.: The Existence of the QR Code Guiding to a Malicious Web Site and a Counterfeit Attack Using It (in Japanese), IEICE Technical Report, ICSS 2018-6, Vol.118, No.109, pp.33–38, IEICE (2018).

[7] Okuma, H., Takita, M. and Morii, M.: On a Construction of Fake QR Code, its Effect, and a Countermeasure (in Japanese), *Proc. Computer*

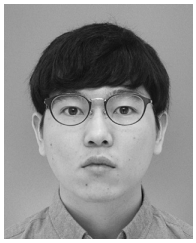
Security Symposium 2018 (*CSS 2018*), pp.987-992, IPSJ (2018).
[8] Information technology - Automatic identification and data capture techniques - QR Code bar code symbology specification, ISO/IEC 18004:2015, ISO (2015).
[9] QR Code Generator, available from ⟨https://www.unitag.io/qrcode⟩ (accessed 2020-11-29).
[10] QRgraphy, available from ⟨https://qrgraphy.com/⟩ (accessed 2020-11-29).
[11] Sucuri: How Modern Web Phishing Works, available from ⟨https://blog.sucuri.net/2016/08/modern-web-phishing-works.html⟩ (accessed 2020-11-29).
[12] Takita, M., Okuma, H. and Morii, M.: A Construction of Fake QR Codes Based on Error-Correcting Codes and Its Threat (in Japanese), *Proc. 17th Forum on Information Technology 2018* (*FIT 2018*), Vol.4, pp.2–6, IPSJ and IEICE (2018).
[13] Takita, M., Okuma, H. and Morii, M.: On a Construction of Fake QR Code using Erasure Decoding (in Japanese), *Proc. 2019 Symposium on Cryptography and Information Security* (*SCIS 2019*), 8 pages, IEICE (2019).
[14] Takita, M., Okuma, H. and Morii, M.: A Construction of Fake QR Codes Based on Error-Correcting Codes, *Proc. 6th International Symposium on Computing and Networking* (*CANDAR 2018*), pp.188–193, IEEE Computer Society (2018).
[15] Tech in Asia: Thieves are pickpocketing wallet apps in China, available from ⟨https://www.techinasia.com/fake-qr-code-scams-china⟩ (accessed 2021-03-28).
[16] Trend Micro: Trend Micro Cloud App Security Report 2019 (2019), available from ⟨https://www.trendmicro.com/vinfo/us/security/research-and-analysis/threat-reports/roundup/trend-micro-cloud-app-security-report-2019⟩ (accessed 2020-11-29).
[17] Vidas, T., Owusu, E., Wang, S., Zeng, C., Cranor, L.F. and Christin, N.: QRishing: The susceptibility of smartphone users to QR code phishing attacks, *Proc. International Conference on Financial Cryptography and Data Security* (*FC 2013*), Lecture Notes in Computer Science, Vol.7862, pp.52–69, Springer, Heidelberg (2013).
[18] ZXing, available from ⟨https://github.com/zxing/zxing⟩ (accessed 2020-11-29).
[19] Zhou Y. and Jiang, X.: Dissecting Android Malware: Characterization and Evolution, *Proc. IEEE Symposium on Security and Privacy* (*S&P 2012*), pp.95–109, IEEE (2012).

Editor's Recommendation

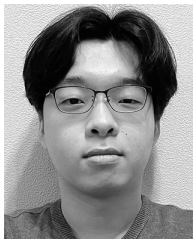
The authors of this paper proposed, implemented, and evaluated defense schemes against existing spoofed QR code attacks. Their experimental results clearly demonstrated that the proposed schemes can detect the attacks with high accuracy. The authors also evaluated the detection accuracy in a realistic situation where stains are added to the QR code, deriving the practical findings. The theme addressed in this paper is about attacks using QR codes, which are widely used in our life, and the research ethics that should be taken into consideration as security research are fully discussed in the paper. For these reasons, we have selected this paper as our recommended paper.

(Program Chair of the Computer Security Symposium 2020, Tatsuya Mori)

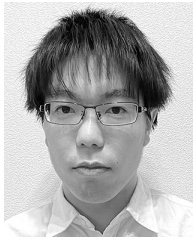
Toshihiro Ohigashi received his B.E. and M.E. degrees from the University of Tokushima, and his Ph.D. degree from Kobe University in 2002, 2004, and 2008, respectively. From 2008 to 2015 he was an Assistant Professor in the Information Media Center, Hiroshima University. From 2015 to 2018, he was a Junior Associate Professor in Tokai University. He is currently an Associate Professor in Tokai University. His current research interests include cryptography, information security, and network protocol. He received the SCIS 20th Anniversary Award and SCIS 2013 Innovation Paper Award from ISEC group of IEICE in 2003 and 2014, respectively. He also received the CSS 2014 best paper award from CSEC group of IPSJ in 2014, the IPSJ Yamashita SIG Research Award from IPSJ in 2015, and the Best Paper Award from IEICE in 2015. He is a member of IPSJ and IEICE.



Shuya Kawaguchi received his B.E. degree from Tokai University in 2021. Since 2021, he has been a master's student in the Graduate School of Information and Telecommunication Engineering, Tokai University. His current research interests include information security and usable security. He received the CSS 2020 best student paper award from CSEC group of IPSJ in 2020.



Kai Kobayashi received his B.E. degree from Tokai University in 2020. He joined Internet Initiative Japan Inc. in 2020. His current research interest is information security.



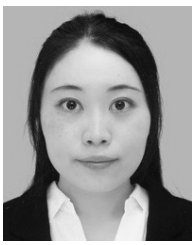
Hayato Kimura received his B.E. and Master of Information and Telecommunication Engineering degrees from Tokai University in 2019 and 2021, respectively. He joined LINE Corporation in 2021. His current research interests include cryptography, information security, network security, and network protocol. He received the IWSEC 2017 Best Poster Award in 2017. He is a member of IPSJ.

Tatsuya Suzuki received his B.E. and Master of Information and Telecommunication Engineering degrees from Tokai University in 2018 and 2020, respectively. Since 2020, he has been a doctoral candidate in the Graduate School of Science and Technology Doctoral Program in Risk and Resilience Engineering, University of Tsukuba. His current research interests include cryptography and information security. He received the IWSEC 2017 Best Poster Award in 2017.

Daichi Okabe received his B.E. degree from Tokai University in 2020. Since 2020, he has been a master's student in the Graduate School of Information and Telecommunication Engineering, Tokai University. His current research interest is information security.

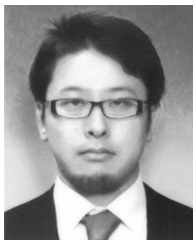
Takuya Ishibashi received his B.E. and Master of Information and Telecommunication Engineering degrees from Tokai University, Japan in 2018 and 2020, respectively. Since 2020, he has been a doctoral candidate in the Graduate School of Science and Technology, Tokai University. His current research interest is information security.

Hiroshi Yamamoto received his B.E., M.E., and Ph.D. degrees from Osaka University in 1991, 1993, and 1996, respectively. From 1996 to 2000, he was an Assistant Professor in Osaka University. From 2000 to 2004, he was a Junior Associate Professor in Tokai University. From 2004 to 2019, he was an Associate Professor in Tokai University. He is currently a Professor in Tokai University. His current research interests include coding theory, cryptography, and web system. He is a member of IEICE.



Maki Inui received her B.E. and M.E. degrees in mathematical engineering in 2016, 2018 respectively from Osaka Prefecture University. Since 2018, she has been engaged in research and development on network security and cyber security at Fujitsu Laboratories Ltd. and FUJITSU Ltd.

Ryo Miyamoto received his B.E. degree in information engineering in 2004 from Nagoya University. He received his M.S. degree in information science in 2006 from Nagoya University. He has been engaged in research and development on information security at Fujitsu Laboratories Ltd. and FUJITSU Ltd.



Kazuyoshi Furukawa received his B.E. degree in information science in 2002 from Hiroshima City University. He received his M.E. degree in science and engineering in 2004 from Tokyo Institute of Technology. Since 2004, he has been engaged in research and development on cryptography and information security at Fujitsu Laboratories Ltd. and FUJITSU Ltd.

Tetsuya Izu received his B.S. and M.S. degrees in mathematics from the University of Tokyo in 1992, and from Rikkyo University in 1994, respectively. He received his Ph.D. in engineering from the University of Electro-Communications in 2007. He has been engaged in research on cryptography, information security, network security and cyber security at FUJITSU Laboratories Ltd. and FUJITSU Ltd. since 1997. He was awarded the Young Scientists' Prize by the Minister of Education, Culture, Sports, Science and Technology Research on side channel attacks and countermeasures in information security in 2007. He was awarded the IPSJ Kiyasu Special Industrial Achievement Award on the development of the dedicated integer factoring devices in 2008. He was also awarded the IPSJ Kiyasu Special Industrial Achievement Award on the development of the coutermeasures against side channel attacks in 2013. He is a senior member of IPSJ. He is also a member of IACR and IEICE.