



INTERNATIONAL JOURNAL ON INFORMATICS VISUALIZATION

journal homepage : www.joiv.org/index.php/joiv



Academic Document Authentication using Elliptic Curve Digital Signature Algorithm and QR Code

Theophilus Wellem^{a,*}, Yessica Nataliani^a, Ade Iriani^a

^a Faculty of Information Technology, Satya Wacana Christian University, Diponegoro 52-60 Salatiga, 50711, Indonesia

Corresponding author: *theophilus.wellem@uksw.edu

Abstract— Paper-based documents or printed documents such as recommendation letters, academic transcripts, and diplomas are prone to forgery. Several methods have been used to protect them, such as watermarking, security holograms, or using paper with specific security features. This paper presents a document authentication system that utilizes QR code and ECDSA as the digital signature algorithm to protect this kind of document from counterfeiting. A digital signature is a well-known technique in modern cryptography used for providing data integrity and authentication. The idea proposed herein is to put a QR code in the printed documents where the QR code includes a digital signature. The signature can later be authenticated using the proposed system by uploading the document for authentication or scanning the document's QR code. The proposed system is particularly developed for digital signature generation and verification of students' final project approval documents as the case study. In traditional settings, the approval form is typically signed directly by the student's advisor dan co-advisor using handwritten signatures. However, using the conventional handwritten signature, the signature on the approval form can be falsified. Therefore, a digital signature generation and verification system is implemented herein to avoid handwritten signature falsification. The advisors can use this system to sign the approval form using a digital signature instead of a handwritten one. The signature is stored in a QR code and is generated using ECDSA with SHA-256 as the hash function. The proposed system is evaluated using documents (i.e., approval forms) with genuine and forged QR codes. The evaluation results showed that the system could verify the authenticity of the approval forms, which contain genuine QR codes. The approval forms that contained forged QR codes were correctly identified.

Keywords— Digital signature; QR code; elliptic curve cryptography; authentication.

Manuscript received 29 Mar. 2022; revised 18 Apr. 2022; accepted 5 Jun. 2022. Date of publication 30 Sep. 2022.
International Journal on Informatics Visualization is licensed under a Creative Commons Attribution-Share Alike 4.0 International License.



I. INTRODUCTION

Handwritten signatures are generally used on letters, documents, diplomas, and so forth to prove their authenticity. If someone has signed a document, it means they have approved the paper's contents. For digital data or documents, cryptographic signatures, widely known as digital signatures, can be applied to protect the integrity of the documents and provide authentication [1]. A digital signature is not a digitized handwritten signature scanned with a document scanner or created with an electronic pen. Furthermore, a digital signature is also not a signature replaced with a code (usually a QR code), which is currently used in several documents in Indonesia, such as Household Registration cards, birth certificates, government letters, and so on. In cryptography and data communication, a digital signature is several bytes (e.g., 64 bytes) of data generated and appended by a signatory (or sender) to the original message to provide

authenticity. The signature values depend on the content and the sender of the message.

With the COVID-19 pandemic still in progress and many are working from home, documents to be approved and signed by managers, professors, lecturers, and others are sent electronically via emails or messenger applications (e.g., WhatsApp, Telegram). The signature herein can be a handwritten signature that is scanned using a document scanner, then pasted on the document, or a signature added to the document with an electronic pen or an online signature (e.g., SignNow). Both types of signatures have a low level of security because unauthorized persons can forge them. Several schemes of signature have been proposed in the literature [2]–[4] to solve this problem, and one of them is to use a digital signature combined with barcodes such as QR codes or other appropriate barcode models [5]–[7].

This paper proposes a system for academic document authentication with students' final project approval documents adopted as the case study. The proposed system uses QR

codes that contain digital signatures to replace the conventional handwritten signature of the students' advisors on the approval form. Furthermore, the system can also be used to verify and validate the signature in the documents. Therefore, the system can generate the digital signature, encode the digital signature to create the QR code, and verify the digital signature inside the QR code on the students' documents. The Elliptic Curve Digital Signature Algorithm (ECDSA) is chosen as the digital signature algorithm used in the proposed system.

Some previous research on the use of digital signatures with QR codes are described below. Rochman et al. [8] applied a digital signature with a QR code to validate the student's course plan and transcript. They used digital signatures and QR codes for document validation with the RSA algorithm for encrypting the message digest. The message digest is generated using the RIPEMD-128 hash function. Apriansyah et al. [9] applied QR codes in the attendance system using Reed Solomon codes for the error correction. Nuraeni et al. [10] applied digital signatures to the diploma legalization process and used digital signatures for document validation. No QR code was used in their system. Ardianto and Wakidah [11] used QR codes for diploma authentication without a digital signature. Suratma and Azis [12] applied a digital signature with a QR code to validate goods retrieval documents. The AES algorithm generates a digital signature, and the QR code contains a digital signature resulting from the document's encryption. Lewis and Thorpe [13] used QR codes for authenticating motor insurance documents. Pal and Kumar [14] applied a QR code to design smart documents using digital signatures and blockchain. In their system, the digital signature is embedded in a QR code for document verification purposes. Farooq et al. [15] applied the ECDSA for the authentication mechanism between smart meters in an advanced metering infrastructure of a smart grid. Teraura et al. [16] applied the ECDSA algorithm with QR code for document authentication. The data area of the QR code is divided into three parts: the public data area (unencrypted data), the private data area (encrypted data), and the digital signature area. Unlike the previous research, the research in this paper utilized the ECDSA for signature generation and verification to validate students' final project approval documents. Furthermore, the digital signature is embedded in the QR code. It can be authenticated by the proposed system using a QR code scanner (specifically developed scanner) or uploading the document to the system.

II. MATERIALS AND METHOD

The research uses a qualitative method consisting of data collection, analysis and interpretation, and validation steps. The data collection step is used to obtain the format of the students' final project approval form and the standard operating procedure for students' final project. Data analysis and interpretation steps are used to identify the requirements to implement the system, study the QR code encoding and decoding methods, hash functions, and suitable cryptographic algorithms to be used in the system. Finally, the data validation step is used to verify the digital signature and the documents.

The digital signature is a well-known technique in modern cryptography used for providing data integrity and

authentication. Currently, there are three digital signature algorithms standards: Digital Signature Algorithm (DSA), RSA Signature Algorithm, and ECDSA [17].

A. Elliptic Curve Digital Signature Algorithm (ECDSA)

Elliptic Curve Cryptography (ECC) is a public-key cryptography scheme proposed by Neal Kobiltz [18] and Victor Miller [19] independently in the 1980s. The approach is based on the algebraic structures of elliptic curves over prime $GF(p)$ and binary finite fields $GF(2^m)$. ECC is attractive because it offers the same level of security as the RSA algorithm with smaller key sizes. ECC is applied to the typical tasks in public-key cryptography, such as key exchange (Elliptic Curve Diffie-Hellman or ECDH) and digital signatures (ECDSA). The ECDSA algorithm can be described as follows.

Alice and Bob agree on two integers (a dan b), a prime number (p) in the elliptic curve equation $y^2 = x^3 + ax + b \pmod{p}$, an elliptic group, and a base point ($B(x_B, y_B)$) selected from the elliptic group [20]. The procedure for generating the public and private keys on ECDSA is as follows.

Alice generates her private and public key in the following way:

- Alice selects an integer x , as the private key.
- Alice calculates $P_A = x.B$, as the public key.

Bob generates his private and public key in the following way:

- Bob selects an integer y , as the private key.
- Bob calculates $P_B = y.B$, as the public key.

The procedure for generating the digital signatures with ECDSA is as follows.

- Alice selects a random integer k , on the interval $[1, p - 1]$.
- Alice calculates $k.B = (x_1, y_1)$ and $r = x_1 \pmod{p}$. If $r = 0$, then return to Step a.
- Alice calculates $k^{-1} \pmod{p}$.
- Alice calculates the hash value of M , $e = H(M)$, where H is the hash function.
- Alice calculates $s = k^{-1} (e + x.r) \pmod{p}$. If $s = 0$, the return to Step a.

The signature for message M is the pair (r, s) . Alice sends both M and (r, s) to Bob.

Meanwhile, the digital signature verification procedure with ECDSA is as follows.

- Bob verifies that r and s are in the interval $[1, p - 1]$.
- Bob retrieves Alice's public key, P_A .
- Bob calculates the hash value of M , $e = H(M)$.
- Bob calculates $w = s^{-1} \pmod{p}$.
- Bob calculates $u_1 = ew \pmod{p}$ dan $u_2 = rw \pmod{p}$.
- Bob calculates $(x_1, y_1) = u_1.B + u_2.P_A$.
- Bob calculates $v = x_1 \pmod{p}$.

If $v = r$, then the signature is valid.

B. Quick Response Code

A Quick Response (QR) code is a 2D barcode initially designed for Japan's automotive industry by Denso Wave in 1994 [21]. The code then was standardized by ISO in 2000, and the current standard for QR codes is ISO/IEC 18004:2015. An example of a QR code is shown in Fig. 1. A QR code can be generated for URL, vCard, text, Wi-Fi access, and so on. The QR code has four standard input encoding modes:

numeric, alphanumeric, byte/binary, and Kanji, as listed in Table I. The code and character for the alphanumeric encoding mode are given in Table II.



Fig. 1 Example of a QR code for the message "Hello World"

TABLE I
ENCODING MODE [22]

| Encoding mode | Bits/char | Character |
|---------------|-----------------------------|--|
| Numeric | 10-bit per three characters | 0-9 (10 characters) |
| Alphanumeric | 11-bit per two characters | 0-9, A-Z, space, \$, %, *, +, -, ., /, : (45 characters) |
| Byte/biner | 8-bit per character | ISO 8859-1 (256 characters) |
| Kanji | 13-bit per character | Shift JIS X 0208 |

TABLE II
ALPHANUMERIC CHARACTER CODE [23]

| Code | Character | Code | Character |
|------|-----------|------|-----------|
| 00 | 0 | 23 | N |
| 01 | 1 | 24 | O |
| 02 | 2 | 25 | P |
| 03 | 3 | 26 | Q |
| 04 | 4 | 27 | R |
| 05 | 5 | 28 | S |
| 06 | 6 | 29 | T |
| 07 | 7 | 30 | U |
| 08 | 8 | 31 | V |
| 09 | 9 | 32 | W |
| 10 | A | 33 | X |
| 12 | C | 34 | Y |
| 13 | D | 35 | Z |
| 14 | E | 36 | space |
| 15 | F | 37 | \$ |
| 16 | G | 38 | % |
| 17 | H | 39 | * |
| 18 | I | 40 | + |
| 19 | J | 41 | - |
| 20 | K | 42 | . |
| 21 | L | 43 | / |
| 22 | M | 44 | : |

In data encoding, the bitstream sequence consists of a mode indicator, a message length indicator, and a message. The mode indicator is an indicator for the encoding mode type, shown in Table III, while the message length indicator is an indicator for the message length in bits. Each version of the QR code with a different encoding mode has an additional message length indicator, as shown in Table IV. QR code has various versions, depending on the size, from Version 1 to Version 40. For example, Version 1 consists of 21x21 modules, Version 2 consists of 25x25 modules, and Version

40 have 177x177 modules. The black modules represent the value of 1, while the white modules represent the value of 0.

As a kind of data storage, QR code also employs error correction code to correct data errors in the decoding process (by a QR code scanner or reader). The error correction codes used in QR codes are Reed-Solomon codes, BCH codes (for format information), and Golay codes (for Format information). The error correction level consists of four levels, i.e., L, M, Q, and H. The estimated error correction capability of each correction level is described in Table V.

TABLE III
INDICATOR MODE [23]

| Indicator | Information |
|-----------|--------------------------------------|
| 0001 | Numeric encoding |
| 0010 | Alphanumeric encoding |
| 0100 | Byte encoding |
| 1000 | Kanji encoding |
| 0011 | Structured append mode |
| 0111 | Extended channel interpretation mode |
| 0101 | FNC1 encoding (position 1) |
| 1001 | FNC1 encoding (position 2) |
| 0000 | The end of message/terminator |

TABLE IV
MESSAGE LENGTH INDICATOR (NUMBER OF BITS) [23]

| Encoding mode | Version | Version | Version |
|---------------|---------|---------|---------|
| | 1-9 | 10-26 | 27-40 |
| Numeric | 10 | 12 | 14 |
| Alphanumeric | 9 | 11 | 13 |
| Byte/biner | 8 | 16 | 16 |
| Kanji | 8 | 10 | 12 |

TABLE V
QR CODE ERROR CORRECTION LEVEL [22]

| Correction Level | Error Correction Ability |
|-----------------------|----------------------------------|
| L (<i>low</i>) | 7% of codewords can be returned |
| M (<i>medium</i>) | 15% of codewords can be returned |
| Q (<i>quartile</i>) | 25% of codewords can be returned |
| H (<i>high</i>) | 30% of codewords can be returned |

The number of data stored in a QR code depends on the input data type, version, and error correction level. The higher the error correction level, the less the data storage capacity. Version 1 with error correction level H has the least data storage capacity, while version 40 with error correction level L has the largest storage capacity. The total capacity of storage for version 1-H (version 1 with error correction level H), 15-Q (version 15 with error correction level Q), 25-M (version 25 with error correction level M), and 40-L (version 40 with error correction level L), are shown in Table VI.

TABLE VI
SAMPLE OF MAXIMUM CAPACITY FOR 1-H, 15-Q, 25-M, 40-L [9]

| Encoding mode | Maximum Number of Characters | | | |
|---------------|------------------------------|------|-------|-------|
| | 1-H | 15-Q | 25-M | 40-L |
| Numeric | 17 | 703 | 2,395 | 7,089 |
| Alphanumeric | 10 | 426 | 1,451 | 4,296 |
| Byte/biner | 7 | 292 | 997 | 2,953 |
| Kanji | 4 | 180 | 614 | 1,817 |

III. RESULTS AND DISCUSSION

This section presents the design and implementation results of the proposed system. First, the description of the signature generating and verifying using ECDSA is described, followed by an example of QR code encoding process. Finally, the implementation of the proposed system is described along with evaluation results

A. Signature Generation and Verification

The flowcharts for generating and verifying a digital signature with QR code and ECDSA are shown in Figs. 2 and 3, respectively.

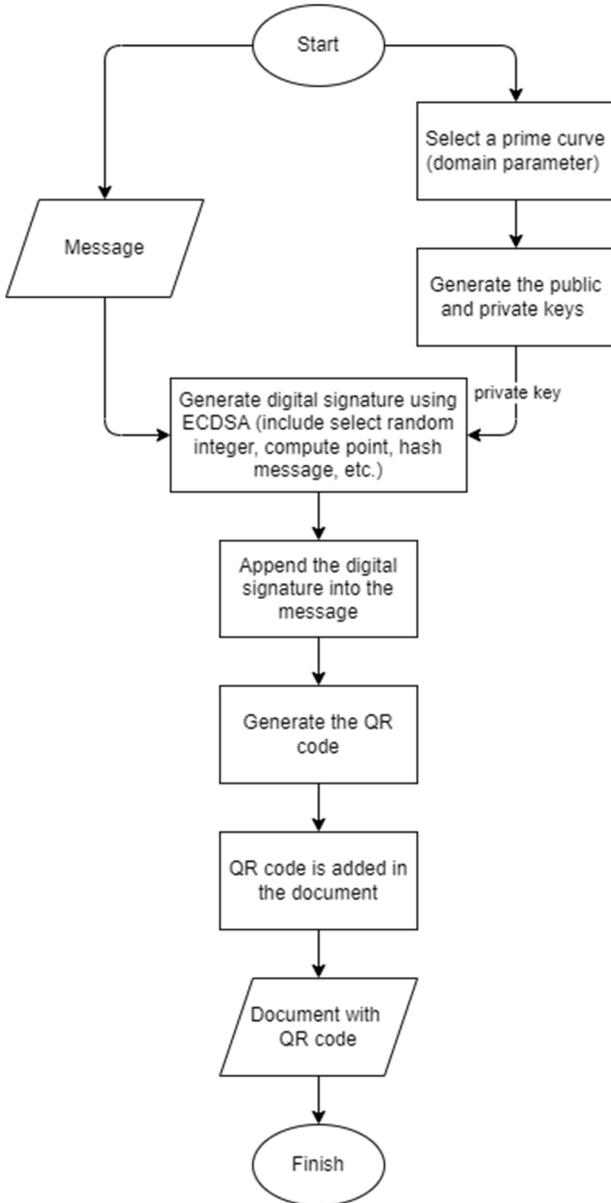


Fig. 2 Flowchart of the generating digital signature in the proposed system

The process of generating the digital signature is described below.

Input: message.

1. Select domain parameters used in the signature process (e.g., curve name, hash function).
2. Generate the public and private keys using the domain parameters from Step 1.

3. Hash the message using a hash function (e.g., SHA-256) to obtain the hash value.
4. Generate the digital signature using the private key from Step 2 and the hash value from Step 3.
5. Append the digital signature obtained in Step 4 to the message.
6. Generate a QR code for the result from Step 5.
7. Put the QR code into the document.

Output: A document with a QR code that contains the message and digital signature.

The process of verifying the digital signature is described below.

Input: QR code or document with QR code.

1. Scan the QR code or upload the document (in .pdf) for the decoding process to obtain the message and the digital signature.
2. Hash the message (obtained from Step 1) using the same hash function in the signature generation process to get the hash value.
3. Use the public key and the hash value (obtained from Step 2) to verify the digital signature (obtained from Step 1).

Output: Signature verification result (valid or not valid).

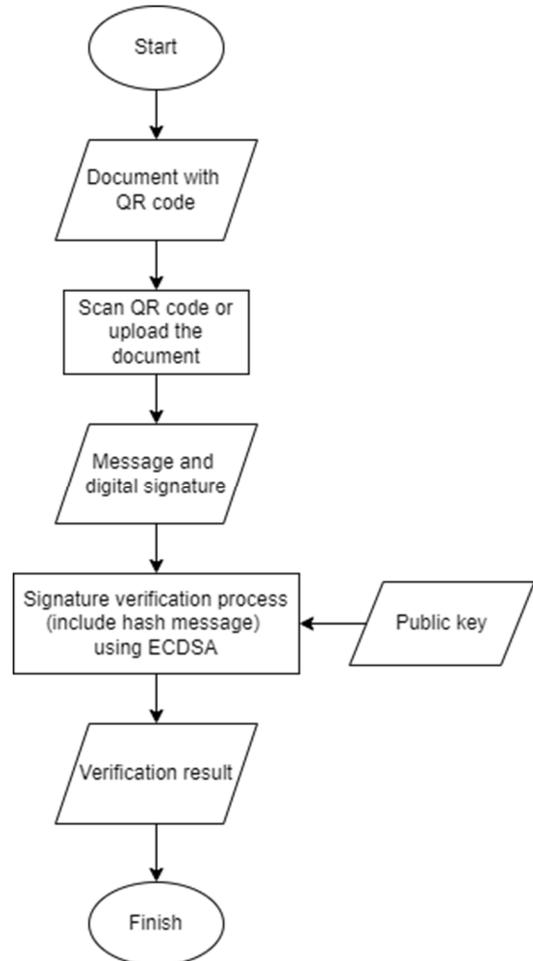


Fig. 3 Flowchart of the verifying digital signature in the proposed system

An example of the output document generated by the proposed system is shown in Fig. 4. The QR code contains the digital signature generated using ECDSA.

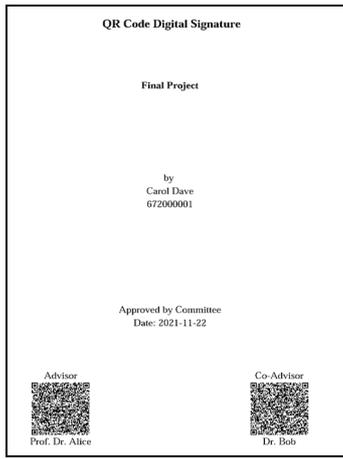


Fig. 4 Example of the output document

B. QR code Encoding and Decoding

A QR code is generated by a QR code encoder. An example of QR code encoding is presented below, and a short message is used to simplify the presentation. The message to be encoded is FTI-UKSW, which consists of alphanumeric data with a length of 8, and a high error correction level is selected. The QR code generated for this short message with a high error correction level. For this short message, the QR code's version generated is Version 1. Following the QR code specification, a Version 1 QR code with a high error correction level (1-H), the total number of codewords (n) = 26, the number of codewords for the message (k) = 9, and the number of error correction codewords ($n-k$) = 17. The calculation for the FTI-UKSW encoding process is described in Table VII.

TABLE VII
QR CODE ENCODING CALCULATION EXAMPLE

| No | Tasks | Results |
|----|---|---|
| 1 | Break the message every two characters. According to Table 1, the alphanumeric encoding mode has a format of 11-bit per two characters. | FT I- UK SW |
| 2 | Code each character according to Table 2. | 15 29 18 41 30 20 28 32 |
| 3 | Multiply the first number by the constant 45, then add it by the second number. | $(45*15) + 29 = 704$ $(45*18) + 41 = 851$ $(45*30) + 20 = 1370$ $(45*28) + 32 = 1292$ |
| 4 | Form in 11-bit binary format. According to Table 1, the alphanumeric encoding mode has a format of 11-bit per two characters. | 01011000000 01101010011 10101011010 10100001100 |
| 5 | Add a terminator at the end of the message with "0000", according to Table 3. | 0101100000 01101010011 10101011010 10100001100 0000 |
| 6 | Add message length = 8 at the beginning of the codewords. According to Table 4, the number of bits to represent the message length in alphanumeric encoding mode is 9 bits, and in binary 9-bits, eight is 000001000. | 000001000 01011000000 01101010011 10101011010 10100001100 0000 |
| 7 | Add the alphanumeric encoding mode, namely "0010", at the beginning of the codewords, according to Table 3. | 0010 000001000 01011000000 01101010011 10101011010 10100001100 0000 |
| 8 | Group into 8-bit. | 00100000 01000010 11000000 01101010 01110101 01101010 10000110 00000 |
| 9 | Add "0" at the end of the codewords (zero padding) so that each block becomes 8-bit. | 00100000 01000010 11000000 01101010 01110101 01101010 10000110 00000000 |
| 10 | Add "11101100" and "00010001" at the end of the codewords alternately so that the new codewords are nine blocks long. | 00100000 01000010 11000000 01101010 01110101 01101010 10000110 00000000 11101100 |
| 11 | Convert each block in decimal. | 32 66 192 106 117 106 134 0 236 |
| 12 | The message is formed into a message polynomial. The result is $m(x)$. | $m(x) = 32x^8 + 66x^7 + 192x^6 + 106x^5 + 117x^4 + 106x^3 + 134x^2 + 0x^1 + 236$ |
| 13 | Multiply $m(x)$ with x^{17} . The result is $m'(x)$. | $m'(x) = 32x^{25} + 66x^{24} + 192x^{23} + 106x^{22} + 117x^{21} + 106x^{20} + 134x^{19} + 0x^{18} + 236x^{17}$ |
| 14 | Determine the polynomial generator, $g(x)$ according to the QR code specification. | $g(x) = x^{17} + 119x^{16} + 66x^{15} + 83x^{14} + 120x^{13} + 119x^{12} + 22x^{11} + 197x^{10} + 83x^9 + 249x^8 + 41x^7 + 143x^6 + 134x^5 + 85x^4 + 53x^3 + 125x^2 + 190x + 79$ |
| 15 | Compute $r(x) = m'(x) \text{ mod } g(x)$. | $r(x) = 211x^{16} + 101x^{15} + 197x^{14} + 180x^{13} + 243x^{12} + 0x^{11} + 68x^{10} + 55x^9 + 103x^8 + 208x^7 + 225x^6 + 200x^5 + 87x^4 + 135x^3 + 223x^2 + 190x + 213$ |
| 16 | Concatenate the coefficient of $m(x)$ with the coefficient of $r(x)$. | 32 66 192 106 117 106 134 0 236 211 101 197 180 243 0 68 55 103 208 225 200 87 135 223 190 213 |
| 17 | Convert to 8-bit binary. The final codewords will be generated into QR code. | 00100000 01000010 11000000 01101010 01110101 01101010 10000110 00000000 11101100 11010011 01100101 11000101 10110100 11110011 00000000 01000100 00110111 01100111 11010000 11100001 11001000 01010111 10000111 11011111 10111110 11010101 |
| 18 | QR code result. |  |

An example of the QR code generated by the proposed system is shown in Fig. 5. Using a high error correction level (H), the QR code version is 12 (65x65 modules). It can store 158 data codewords and 308 error correction codewords. In contrast, the QR code version is 7 (45x45 modules) if a low error correction level (L) is used. The QR code can store 156 data codewords and 40 error correction codewords. The content of the QR code is shown below.

```
672000001|Carol Dave|67001|2021-11-20
14:20:46|MEUCIQDZRNbIOPQf99WYUyKha73/OU5
WUUGrwar7rM8PQIGxQwIgbP4NKj31khJNJdeZbJL
kq95S7i876+afu1gdhCXY1Tc=
```



Fig. 5 Example of QR code with error correction level: (a) High and (b) Low

A QR code is typically read using a scanner that decodes the QR code for the decoding process. The decoding process starts with recognizing the black/white modules, decoding the format information, determining the version, and releasing the masking [23]. Next, the codewords are processed for error detection and correction. If there are errors in the codewords, the errors are corrected. Otherwise, the data part in the codewords is extracted as the output. QR code encoder and decoder libraries are widely available in various programming languages to simplify software development that requires QR code processing. The `phpqrcode` library [24] and `ZXing` library [25] are used in the proposed system to implement the encoder and the decoder, respectively.

C. System Implementation

The proposed system is designed and implemented as a web-based application using PHP programming language. The system users are categorized as Administrator, Advisor, and Student. The administrator has the task of managing all data and users in the system, including verifying of signature in students' documents. Users with the role of Advisors can view and sign the documents submitted by students. Finally, users with the role of Students can input their final project data and submit their documents for approval. The workflow of the proposed system for signature generation is as follows.

1. The administrator creates user accounts for advisors and students.
2. When a student needs his or her advisor's signature, the student inputs the title of the final project and selects the advisor in the system.
3. If the advisor is accessing the system for the first time, the system will generate a public key and a private key automatically, then these keys will be stored in the system to be used in the future.
4. The system generates a message (P) containing the student number, name, advisor ID, signature date, and timestamp.

5. Sign the message (P) by executing the sign function. The private key is used as one of the parameters in the sign function. This step creates a digital signature (S).
6. S is appended to P to create a string T. The system then generates a QR code for T.
7. The system put the generated QR code to the approval form.
8. The student can now download the signed approval form.

The digital signature generation process is described below. Students log in to the system, input their final project data, and select their advisor(s), as shown in Fig. 6.

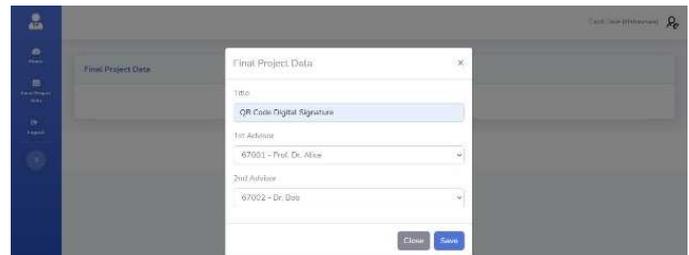


Fig. 6 Student's portal to input the paper title and choose the advisor

When an advisor logs in, the system displays the student data that requires the advisor's signature, as shown in Fig. 7.

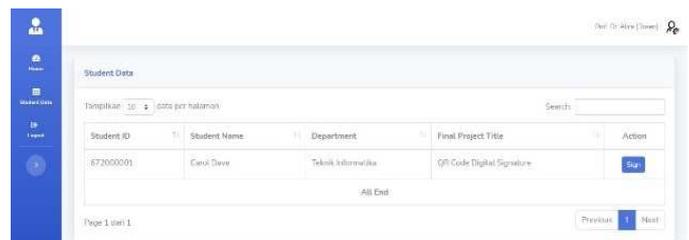


Fig. 7 Advisor's portal that displayed signature's request from the student

The advisor can then sign the document. The digital signature and the QR code generation are executed in this step—the advisor inputs the password to sign the document, as shown in Fig. 8.



Fig. 8 Advisor's portal to sign the document

The implementation of ECDSA in this system uses functions from the OpenSSL library [26]. The `openssl_get_curve_names()` function is used to obtain a list of curves supported by OpenSSL. The curve adopted herein is `prime256v1` (NIST P-256). After selecting the curve, the process for generating the private key and the public key is executed (the `openssl_pkey_new()` and `openssl_pkey_get_details()` functions are used here). The generated private key can be stored in a `.pem` file or a `.der` file and loaded from the file when used, while

the public key is generally stored in the form of a digital certificate or can be stored in a .pem file. An example of a private key and a public key is as follows:

```
-----BEGIN EC PRIVATE KEY-----
MHcCAQEIEILwCVMMEyC/cajYjaPXbxZRhgStlK
xs0jBvvh7JBDBD7oAoGCCqGSM49
AwEHoUQDQgAEEvZShMEplL5yx+vFodjO+cKiy
qf/5FS4RL9sSB2SpM0FgrTPTRjd
Ebqaula+tgvh3cvcohwf0OfKP7R2koFBmA==
-----END EC PRIVATE KEY-----

-----BEGIN PUBLIC KEY-----
MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEEv
ZShMEplL5yx+vFodjO+cKiyqf/
5FS4RL9sSB2SpM0FgrTPTRjdE
bqaula+tgvh3cvcohwf0OfKP7R2koFBmA==
-----END PUBLIC KEY-----
```

After the key generation process, the digital signature is generated using the specified parameters (domain parameters), the private key, and the hash value. An example of the resulting digital signature (in base64 encoding format) is shown below.

```
MEUCIQDZRNbIOPQf99WYUykha73/0U5WUUGrwa
r7rM8PQIqXQwIgbP4NKj31khJNJdeZbJLkq95S
7i876+afu1gdhCXylTc=
```

The student can now log in to the system and observe that the signature request has been completed, as shown in Fig. 9. Finally, the student can download the signed approval form and put it in the final project document. An example of the signed document is shown in Fig. 10.



Fig. 9 Student's portal when the signature request has been completed

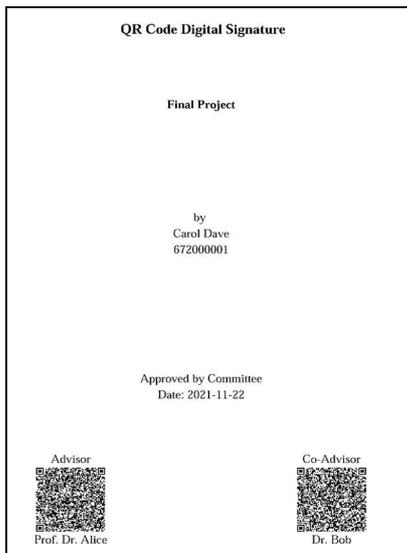


Fig. 10 Signed approval form example

The administrator performs the digital signature verification process by uploading the approval form to the system. Verification is carried out to check the validity of the signatures. The workflow for signature verification is as follows.

1. The administrator uploads the approval form (in .pdf).
2. The system decodes the QR code into a string T, consisting of the message and the digital signature.
3. The system separates T into a message (P) and a digital signature (S).
4. The system verifies the digital signature (S) by executing the verify function. The public key is used as one of the parameters in the verify function.
5. If the call to verify function returns 1, the digital signature is authentic (valid).

The verification part on the administrator's portal is shown in Fig. 11. Fig. 12 shows the verification result for the approval form shown in Fig. 10, which is valid.



Fig. 11 Administrator's portal to verify the document



Fig. 12 Valid verification

D. System Testing and Evaluation

As shown in the previous section, an example is given for the generation and verification of digital signatures for a student named Carol Dave (M1) with two supervisors, namely Prof. Dr. Alice (P1) and Dr. Bob (P2). In the system evaluation, four scenarios were tested.

1) *Scenario 1 (data integrity)*: After M1 got the approval form, for some reason, the form was damaged. M1 scans the QR code from the two advisors to get the message and digital signature contained in the QR code. M1 then generates the QR code from P1 and P2 using another QR code generator (e.g., the one available on the internet). After that, M1 creates a new approval form and puts the new QR code on the M1's approval form.

Result: The QR code of P1 on the original approval form is shown in Fig. 13(a). It contains 672000001|CarolDave|67001|2021-11-20 14:20:46|MEUCIQDZRNbIOPQf99WYUykha73/0U5WUUGrwar7rM8PQIqXQwIgbP4NKj31khJNJdeZbJLkq95S7i876+afu1gdhCXylTc=

The QR code of P1 from another QR code generator is shown in Fig. 13(b). It contains 672000001|Carol Dave|67001|2021-11-20 14:20:46|MEUCIQDZRNbIOPQf99WYUykha73/0U5

WUUGrwar7rM8PQIgXQwIgbP4NKj31khJNJdeZbJLkq95S7i876+afu1gdhCXY1Tc=



Fig. 13 QR code of P1: (a) original P1; (b) from another QR code generator

Both QR codes contain the same messages and digital signatures. The verification result for the M1 new approval form is valid, as shown in Fig. 14. Therefore, if there are no changes to both the message and digital signature, the validation will be successful, even though the QR code is generated with other generators. Note that the QR code encoder in the proposed system is not equipped with its own certificate (public key) and private key. Thus, any QR code encoder can re-create the QR code as long as the message and digital signature are not changed. The development of a certified encoder/decoder is left for future work.

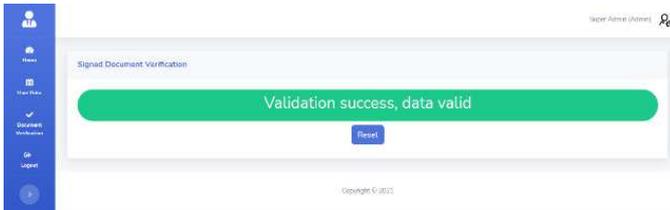


Fig. 14 Success validation for Scenario 1

2) *Scenario 2 (authentication, forged signature)*: After M1 gets the approval form, another student, Eve Adam (M2), who is also P1's student, takes the QR code of P1 from the M1's approval form. M2 then creates a new approval form by himself and puts the QR code of P1 on his approval form. In this scenario, M2 tries to forge the approval form using a QR code obtained from another student with the same advisor.

Result: The QR code of P1 on M2's approval form contains 672000001|Carol Dave|67001|2021-11-20 14:20:46|MEUCIQDZRNbIopQf99WYUykha73/0U5WUUGrwar7rM8PQIgXQwIgbP4NKj31khJNJdeZbJLkq95S7i876+afu1gdhCXY1Tc=

The verification result for M2's approval form is failed (shown in Fig. 15) because the message on the QR code has M1's identity. In contrast, the message in the newly created approval form has M2's identity.

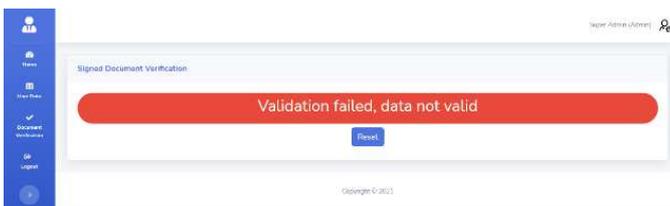


Fig. 15 Failed validation for Scenario 2

3) *Scenario 3 (authentication, forged signature with newly generated QR code)*: After M1 gets the approval form, M2, who is also P1's student, replaces the student ID and name

(obtained by scanning the QR code of P1) with his identity and generates a new QR code. M2 then creates a new approval form and puts his version of P1's QR code on the approval form. This scenario is similar to that of Scenario 1. However, a new QR code is generated in this scenario. M2 tries to forge the approval form using his version of P1's QR code.

Result: The new version of P1's QR code on M2's approval form with M2's identity is shown in Fig. 16. It contains 672000002|Eve Adam|67001|2021-11-20 14:20:46|MEUCIQDZRNbIopQf99WYUykha73/0U5WUUGrwar7rM8PQIgXQwIgbP4NKj31khJNJdeZbJLkq95S7i876+afu1gdhCXY1Tc=



Fig. 16 QR code of P1 with M2's identity as the message

The message part on the QR code is replaced with the M2's identity. The verification result for M2's approval form failed, as shown in Fig. 17, because the digital signature does not match the message.

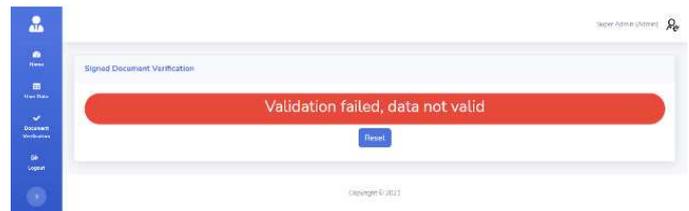


Fig. 17 Failed validation for Scenario 3

4) *Scenario 4 (forged approval form)*: M2 is not registered in the database but has an approval form (created by himself). The verification result for M2's approval form is failed, as shown in Fig. 18, because M2's data are not found in the database.

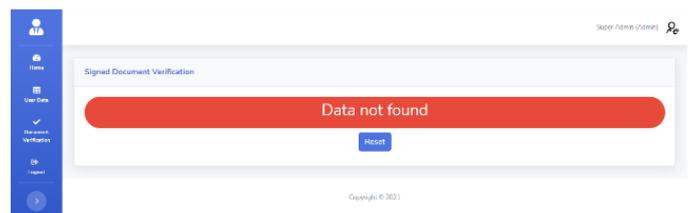


Fig. 18 Data not found for Scenario 4

IV. CONCLUSION

This paper proposes the usage of QR codes and ECDSA for document authentication to prevent signature or document forgery. A system is implemented for students' final project document authentication using a case in academic settings. The system was evaluated using both genuine and false QR code signatures. Based on the evaluation and testing results, it is found that: (1) As long as there are no changes to the message and digital signature, the verification results are valid even though another QR code generator generates the QR code. This can be improved using a certified QR code encoder

and decoder, (2) If the message on the QR code is not the same as the message on the document, then the validation is failed, and (3) If the message on the QR code does not match with the digital signature, then the validation is failed. The proposed method and system can be extended for any documents that require handwritten signatures. A study and implementation of a certified QR code encoder and decoder is a potential future work. Using a certified encoder and decoder can improve the security and reliability of the system.

ACKNOWLEDGMENT

This research is funded by the Office of the Vice Rector V for Research and Community Service, Satya Wacana Christian University, with Decree Number 196/Pen./Rek./V/2021.

REFERENCES

- [1] N. Dlamini, S. Mthethwa, and G. Barbour, "Mitigating the Challenge of Hardcopy Document Forgery," *2018 Int. Conf. Adv. Big Data, Comput. Data Commun. Syst. icABCD 2018*, Sep. 2018, doi: 10.1109/ICABCD.2018.8465401.
- [2] A. Singhal and R. S. Pavithr, "Degree Certificate Authentication using QR Code and Smartphone," *Int. J. Comput. Appl.*, vol. 120, no. 16, pp. 38–43, 2015.
- [3] H. A. Ahmed and J.-W. Jang, "Higher Educational Certificate Authentication System Using QR Code Tag," *Int. J. Appl. Eng. Res.*, vol. 12, no. 20, pp. 9728–9734, 2017.
- [4] S. K. Thamer and B. N. Ameen, "A New Method for Cipherring a Message Using QR Code," *Comput. Sci. Eng.*, vol. 6, no. 2, pp. 19–24, 2016.
- [5] R. Focardi, F. L. Luccio, and H. A. M. Wahsheh, "Usable cryptographic QR codes," *Proc. IEEE Int. Conf. Ind. Technol.*, vol. 2018-February, pp. 1664–1669, Apr. 2018, doi: 10.1109/ICIT.2018.8352431.
- [6] A. Wibiyanto and I. Afrianto, "QR code and transport layer security for licensing documents verification," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 407, pp. 1–8, 2018, doi: 10.1088/1757-899X/407/1/012069.
- [7] M. A. Sadikin and S. U. Sunaringtyas, "Implementing digital signature for the secure electronic prescription using QR-code based on android smartphone," *Proc. - 2016 Int. Semin. Appl. Technol. Inf. Commun. ISEMANTIC 2016*, pp. 306–311, Mar. 2017, doi: 10.1109/ISEMANTIC.2016.7873856.
- [8] F. F. Rochman, I. K. Raharjana, and T. Taufik, "Implementation of QR Code and Digital Signature to Determine the Validity of KRS and KHS Documents," *Sci. J. Informatics*, vol. 4, no. 1, pp. 8–19, May 2017, doi: 10.15294/SJI.V4I1.7198.
- [9] A. Apriansyah, F. Fauziah, and N. Hayati, "Implementasi Algoritma Reed Solomon Codes Pada Proses Encoding QR Code pada Sistem Absensi," *J. Infomedia Teknik Inform. Multimed. Jar.*, vol. 4, no. 2, pp. 75–80, 2019, doi: 10.30811/JIM.V4I2.1572.
- [10] F. Nuraeni, H. Agustin, I. M. Muharam, T. Informatika, and T. Tasikmalaya, "Implementasi Tanda Tangan Digital Menggunakan RSA dan SHA-512 Pada Proses Legalisasi Ijazah," in *Konferensi Nasional Sistem Informasi (KNSI) 2018*, Mar. 2018, pp. 864–869.
- [11] E. Ardianto and N. Wakhidah, "Pengembangan Metode Otentikasi Keaslian Ijasah dengan Memanfaatkan Gambar QR Code," *J. Transform.*, vol. 13, no. 2, pp. 35–41, Jun. 2016, doi: 10.26623/TRANSFORMATIKA.V13I2.325.
- [12] A. Suratma, A. G. P. Suratma, and A. Azis, "Tanda Tangan Digital Menggunakan QR Code dengan Metode Advanced Encryption Standard," *Techno (Jurnal Fak. Tek. Univ. Muhammadiyah Purwokerto)*, vol. 18, no. 1, pp. 59–68, Jun. 2017, doi: 10.30595/techno.v18i1.1482.
- [13] O. Lewis and S. Thorpe, "Authenticating Motor Insurance Documents using QR Codes," Apr. 2019, doi: 10.1109/SOUTHEASTCON42311.2019.9020614.
- [14] K. Pal and C. R. S. Kumar, "QR Code Based Smart Document Implementation Using Blockchain and Digital Signature," *Adv. Intell. Syst. Comput.*, vol. 1174, pp. 449–465, 2021, doi: 10.1007/978-981-15-5616-6_32.
- [15] S. M. Farooq, S. M. Suhail Hussain, and T. S. Ustun, "Elliptic Curve Digital Signature Algorithm (ECDSA) Certificate Based Authentication Scheme for Advanced Metering Infrastructure," *2019 Innov. Power Adv. Comput. Technol. i-PACT 2019*, Mar. 2019, doi: 10.1109/I-PACT44901.2019.8959967.
- [16] N. Teraura, I. Echizen, and K. Iwamura, "A QR Symbol with ECDSA for Both Public and Secret Areas using Rhombic Sub-cells," 2020.
- [17] C. F. Kerry and P. D. Gallagher, "Digital Signature Standard (DSS)," 2013. doi: 10.6028/NIST.FIPS.186-4.
- [18] N. Koblitz, "Elliptic Curve Cryptosystems," *Math. Comput.*, vol. 48, no. 177, pp. 203–209, 1987.
- [19] V. S. Miller, "Use of Elliptic Curves in Cryptography," *Adv. Cryptol.*, vol. 218, pp. 417–426, 1986.
- [20] R. Munir, *Kriptografi*, 1st ed. Bandung: Informatika, 2019.
- [21] D. Wave, "QR Code development story." [Online]. Available: <https://www.denso-wave.com/en/system/iot/support/>. [Accessed: Mar. 02, 2022].
- [22] E. F. Nurdiansyah and I. Afrianto, "Implementasi QR Code Sebagai Tiket Masuk Event dengan Memperhatikan Tingkat Koreksi Kesalahan," *J. Teknol. dan Inf.*, vol. 7, no. 2, pp. 25–44, Sep. 2017, doi: 10.34010/JATI.V7I2.491.
- [23] ISO/IEC 18004:2015, "ISO - ISO/IEC 18004:2015 - Information technology — Automatic identification and data capture techniques — QR Code bar code symbology specification," 2015..
- [24] "PHP QR Code - QR code generator, an LGPL PHP library." [Online]. Available: <https://phpqrcode.sourceforge.net/>. [Accessed: Mar. 02, 2022].
- [25] "GitHub - zxing/zxing: ZXing ('Zebra Crossing') barcode scanning library for Java, Android." [Online]. Available: <https://github.com/zxing/zxing>. [Accessed: Mar. 02, 2022].
- [26] "OpenSSL: Cryptography and SSL/TLS Toolkit." [Online]. Available: <https://www.openssl.org/>. [Accessed: Mar. 02, 2022].