DOI: 10.2478/ijasitels-2023-0008

Methods for data validation using QR codes

Oleksik Vlad Andrei[1], Pitic Elena Alina[2], Crețulescu Radu George[1]
[1]Computer Science and Electrical and Electronics Engineering Department, Faculty of Engineering, "Lucian Blaga" University of Sibiu, Romania
[2]Department of Mathematics and Informatics, Faculty of Sciences, "Lucian Blaga" University of Sibiu, Romania
{vladandrei.oleksik, alina.pitic, radu.cretulescu} @ulbsibiu.ro

Abstract

This article aims to elaborate on a few strategies designed for validating data encoded as a QR code. As the use-cases for such validation schemes vary widely as the demand for such practices is continuously increasing, the present article will represent a comparative study of checksum, hashing, as well as asymmetric encryption algorithms for the offline validation of claims transmitted through the payload of the exchanged data. Results are evaluated based on different criteria impacting the reliability, such as the size of the QR code and the resulting security.

Keywords: Data validation, Data encoding, Checksum algorithms, Hashing Algorithms, Signature algorithms

1. Introduction

In today's context of constant technological evolution and free access to digital communication tools, data is one of the main resources for research, development, innovation, and economic growth [8]. Open access to technological solutions and communication networks contributes to the increasing amounts of data generated and transmitted daily.

Each owner of an internet-enabled device can access dozens of information resources, which in turn collect and transmit data that we consciously and unconsciously expose, such as location including place of work and/or residence, race, gender, age, contact details, personal interests, marital status, or religious beliefs [5].

This data is collected and stored by organisations for the purpose of further use to identify opportunities or threats, thus preparing their decision-making strategy and adapting to current or future trends.

As mentioned, we all expose huge amounts of data, and organisations don't need all the data we expose, whether it is real or not. What data collectors need to ensure is that this information meets their collection requirements. This is where ***data validation*** comes in.

DOI: 10.2478/ijasitels-2023-0008

Data validation is a crucial practice in ensuring the integrity, accuracy, and structural soundness of data before it is used in various business operations [6]. It encompasses a range of validation types, including data type, constraint, structure, consistency, and code validation, each of which is tailored to check that the data meets the necessary criteria for further processing [10].

At the same time, data validation can be defined as a form of data cleansing, which involves examining source data for accuracy and quality before any processing, import or use takes place [24]. This definition aligns with the perspective presented above, where the importance of ensuring that data meets specific standards and objectives within the constraints of the destination was also emphasized.

Given the enormous amounts of data that are transported daily, securing this data is one of the priorities of organisations that collect this data. This is an obligation for the organisations concerned. For example, under GDPR rules, companies and organisations that collect, store and process personal data are obliged to provide safeguards for the protection of that data [20]. Thus, the organisations concerned must consider securing data in the 2 known forms: *data in motion* and *stored data*. To this end, organisations usually use data ***checksum algorithms***, ***hash algorithms*** and ***encryption algorithms*** [15] [3]. Each of these categories of algorithms plays a significant role in ensuring data integrity and confidentiality.

Checksum algorithms are used for ensuring data integrity. These algorithms generate fixed-size values, known as checksums, which uniquely represent the contents of a given data set. These algorithms aim to identify errors or changes in the data during transmission or storage. A checksum is calculated at the source and sent or saved with the data when it is transferred or stored. When receiving or retrieving data, the checksum is recalculated and if it differs from the original transmitted or stored checksum, this indicates a possible error or corruption of the data [15] [4]. In our research, we used three types of checksum algorithms**.** The first algorithm was ***CRC32*** (*Cyclic Redundancy Check 32*) is a checksum algorithm based on polynomial division, that hashes byte sequences to 32-bit values [2].

Another algorithm used is the Fletcher algorithm, which was devised by John G. Fletcher. There are three popular Fletcher checksum algorithms, of which I used Flecher32 which divides the data word into 16-bit blocks, computes two 16-bit checksums, and adds them together to form a 32-bit Fletcher checksum [13].

The last checksum algorithm used is Adler-32, which is an algorithm written by Mark Adler by modifying Fletcher's checksum. Compared to Flecher's algorithm, Adler32 is mainly known for its simplicity and speed, while Fletcher32 is chosen for applications where efficiency and simplicity are prioritized over cryptographic strength [9].

Hashing algorithms are one-way or irreversible, so the text cannot be deciphered and decoded by anyone else. Thus, hashing protects data at rest, so that even if someone gains access to the server where it is stored, the items remain unreadable [11][26]. In this study, we used SHA-1-HMAC, a keyed hash algorithm based on the SHA1 hash function and used as a hash-based message authentication code (HMAC). The HMAC process mixes a secret key with the message data, hashes the result using the hash function, mixes the hash value with the secret key again, and then applies the hash function a second time [25].

DOI: 10.2478/ijasitels-2023-0008

Data encryption is a security measure that involves the transformation of readable information, called plaintext, into an unreadable format, called ciphertext, using algorithms and cryptographic keys [23] [3]. The process serves to secure sensitive data from unauthorised access while maintaining confidentiality and privacy. In encryption, a mathematical technique uses a cryptographic key to alter the original data, rendering it unreadable without the appropriate decryption key. This cryptographic approach is critical for protecting information during transmission or storage, preventing cybercriminals, unauthorised users, and other harmful entities from posing a threat.

Encryption plays an essential role in securing various digital communications and transactions, such as online banking, e-commerce transactions and the transfer of sensitive information [21][7]. It serves as a fundamental component of cybersecurity strategies, preventing unauthorised parties from interpreting and exploiting confidential data [22].

There are two types of encryption algorithms, namely symmetric encryption algorithms, in which a single key is used for both encryption and decryption of data, and asymmetric encryption algorithms, which involve the use of two keys: the public key, used for encrypting data, and the private key, used for decrypting data [21] [7].

As encryption algorithms we used two asymmetric algorithms: *RSA* and *ECDSA*.

RSA, or ***Rivest-Shamir-Adleman***, is an encryption algorithm that provides a versatile approach for data securing through asymmetric cryptography, offering two distinct methods [14] [28]. One of the methods involves encrypting sensitive information with the recipient's public key, ensuring only the owner of the corresponding private key can decrypt the data. This method is ideal for secure transmission of data across networks, as the sender uses the recipient's public key for encryption. Other method involves encrypting a message with the sender's private key and sending both the encrypted data and the sender's public key to the recipient. Here, the recipient can decrypt the data using the sender's public key, verifying the sender's identity [18].

ECDSA, or ***Elliptic Curve Digital Signature Algorithm***, is an asymmetric cryptographic primitive which stands out for its digital signature efficiency compared to other algorithms, achieved by using smaller keys while maintaining a high level of security. *ECDSA* generates certificates, electronic documents for authenticating the certificate owner, containing information about the key, owner details, and the issuer's signature. The elliptic curve analysis forms the basis of *ECDSA*'s security, making it resistant to current encryption cracking methods. Despite being standardized in 2005, newer protocols favour *ECDSA* due to its relative novelty, reducing the time hackers have had to crack it (What is ECDSA Encryption? 2020). However, *RSA* remains widely used, primarily due to its longer presence since standardization in 1995, even though attackers have had more time to attempt to break it. While *ECDSA* offers enhanced security and is preferred in certain contexts, its drawback lies in complexity during implementation compared to the more straightforward setup of *RSA*, making proper implementation crucial for avoiding vulnerabilities [12].

DOI: 10.2478/ijasitels-2023-0008

2. Methodology

In this section, various QR code-based data validation schemes, each based on a subset of the algorithms described in section 1, will be presented in detail. Along with the specific data encoding scheme, for each case, an emphasis will be placed on certain theoretic remarks regarding the time complexity of the algorithms involved in the process, the physical space occupied by the code used for the data transfer, or the way each of these schemes would integrate into the modern data handling paradigm.

To analyse the behaviour of different data validation strategies, first and foremost, it is required to establish some sample data on which to apply each encoding. As such, a set of sample data (claims) representing a student's current enrolment status, based on the widely used JSON format [17] in data serialization, is presented below:

```
{
 "data": {
  "no": 309,
  "scope": "Internal",
  "universityYear": "2023-2024",
  "date": "12/12/2023",
  "holder": {
   "givenName": "Vlad-Andrei",
   "familyName": "Oleksik",
   "CNP": "1230405065000",
   "studyYear": "I",
   "programme": "Calculatoare"
  },
  "issuer": {
   "university": "ULBS",
   "faculty": "FING"
  }
 }
}
```

In this example, the JSON data contains information about the personal details of a student, as well as a context describing the issuer and intended receiver of the claims. This structure is in accordance with the most used data or authentication flows in information systems [27].

However, this format would apply for serializing any "object" modelled under the OOP paradigm. As it can be noticed, the format does not inherently optimize the space occupied by all the data, nor does the format describe each field efficiently. Thus, the total space required to handle the entire sample would amount to approximately 260 B.

For the data validation, several uses of QR codes will be detailed below. Depending on how one intends to use the QR-encoded data in the validation process, there are two main categories of validation schemes. Thus, a first concept would be the encoding of the data in its entirety, for it to then be validated and processed as-is, without the need for further communication to obtain relevant data. Another approach is only encoding an identifier in the data to be validated, while invariably keeping a copy of the complete claim's server-side. Thus, trusting a remote connection would still be a prerequisite for using data validated using QR codes. This latter approach will be first presented.

sciendo
DOI: 10.2478/ijasitels-2023-0008

2.1. Basic data validation

This strategy relies on the use of an identifier to locate the resources that need to be validated. The integrity of the identifier itself is ensured by using either a checksum or a hash-based message authentication code.

2.1.1. Checksum-enabled basic validation

This first approach is centred on the use of the basic data required for identification, along with a simple checksum based on the Cyclic Redundancy Check algorithm, as described above. Thus, the payload will have the structure presented below:

1230405065000-309-12/12/2023-46B6D55B

The size of the checksum used in this case is 4 bytes. Since both the identification data and the hexadecimal encoded checksum only contain alphanumeric data, the size of the QR code can be easily reduced to a small size as compared to a QR code encoding the same data with the binary encoding.

As mentioned in the previous chapter, were the checksum computation speed of particular concern, the Fletcher32 or Adler32 algorithms could be used instead. However, as this is typically not the case, only the CRC32 will be considered for the purpose of this paper.

It is worth mentioning that, as all the algorithms involved are publicly known and reversible, this method is generally not recommended when the code is used for sensitive identification purposes. Thus, this encoding scheme only ensures resistance against very basic tampering, while being very effective space-wise.

2.1.2. HMAC-enabled basic validation

If a higher security level is expected for validating some basic identification information, while the encoded size represents a concern, then the use of a Hash-based Message Authentication Code primitive instead of the checksum is desirable.

Briefly, this algorithm uses a one-way function which takes in both the payload and a secret key, outputting a value unique (considering the collision probability to be negligible) to each payload and secret key. While the SHA-1 algorithm is not recommended anymore for the purpose of concealing secrets, it is still considered safe for the purpose of HMAC. This HMAC algorithm provides an output of size 20 bytes. Thus, the encoded data will resemble the following format:

1230405065000-309-12/12/2023-C62600020DC4A288A4A1EC411C494AA5B275C1F9

When validating the data, a validator knowing the secret key can compute the same HMAC value for the payload, the data being considered valid if it matches the one provided within the QR code.

The use of this algorithm enables for reasonable security, while maintaining a relatively small size of the encoding data.

DOI: 10.2478/ijasitels-2023-0008

The two validation schemes described above cover the case of validating basic identification data, referencing claims stored server-side. As such, the verification process typically continues online, certifying the accuracy of the data. The next subchapter will present schemes that could be used for validating the entire data based solely on the QR-encoded data.

2.2. Complete data validation

This strategy is focused on encoding the entire data in the QR code. As this scheme would not necessarily require an online validation unless claims dynamically change and the space used will inevitably increase, the use of stronger algorithms for ensuring the integrity of the data is recommended.

Another area of concern when including the whole sample data in the QR code is the format to be used for handling it. Since the data to be ultimately encoded is structured as a payload and a code or signature used for validation, the JSON Web Token standard is a desirable choice. It is widely used in sensitive authentication data handling [27] and encompasses a header describing the validation algorithm to be used, as well as payload and validation fields.

2.2.1. Complete symmetric (HMAC-based) data validation

This first approach keeps relying on the HMAC algorithm, although, for this case, where the offline validation should suffice, this article will treat the HS256 (SHA-256-HMAC) algorithm. As opposed to the SHA-1-HMAC algorithm, the size of the output is 64 bytes. The final JWT is as presented below:

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJkYXRhIjp7Im5vIjozMDksInNjb3BlIjoiSW50ZXJuYWwiLCJ1bml2ZXJzaXR5WWVhciI6IjIwMjMtMjAyNCIsImRhdGUiOiIxMi8xMy8yMDIzIiwiaG9sZGVyIjp7ImdpdmVuTmFtZSI6IlZsYWQtQW5kcmVpIiwiZmFtaWx5TmFtZSI6Ik9sZWtzaWsiLCJDTlAiOiIxMjMwNDA1MDY1MDAwIiwic3R1ZHlZZWFyIjoiSSIsInByb2dyYW1tZSI6IkNhbGN1bGF0b2FyZSJ9LCJpc3N1ZXIiOnsidW5pdmVyc2l0eSI6IlVMQlMiLCJmYWN1bHR5IjoiRklORyJ9fX0.qWeK7SYOZidksAt8js0UAQyVRYw_1lzoHPd5WjZNYHw

Since the payload, like the validation data, is of considerably larger size, the JSON Web Token will have them encoded in base 64. As such, the alphanumeric encoding mode of the QR code can no longer be used and the binary encoding will be used instead. As only a subset of the characters in the binary character space are used, the efficiency of the space usage in the QR code will decrease unless the resulting token is compressed and converted to another base.

As for the scheme presented in subsection 2.1.2., a validator knowing the secret key convened upon would have to compute the HMAC output for the header-payload combination and the secret, an output corresponding to the last field of the JWT certifying the integrity of the data.

DOI: 10.2478/ijasitels-2023-0008

2.2.2. Complete asymmetric (RSA-based) data validation

This approach switches to the asymmetric RSA algorithm as described in the first chapter. The issuer passes the protected data in the JWT through the private function of the RS256 primitive, the output, of size 256 B, being appended to the encoded data, which would resemble the following structure:

eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJkYXRhIjp7Im5vIjozMDksInNjb3BlIjoiSW50ZXJuYWwiLCJ1bml2ZXJzaXR5WWVhciI6IjIwMjMtMjAyNCIsImRhdGUiOiIxMi8xMy8yMDIzIiwiaG9sZGVyIjp7ImdpdmVuTmFtZSI6IlZsYWQtQW5kcmVpIiwiZmFtaWx5TmFtZSI6Ik9sZWtzaWsiLCJDTlAiOiIxMjMwNDA1MDY1MDAwIiwic3R1ZHlZZWFyIjoiSSIsInByb2dyYW1tZSI6IkNhbGN1bGF0b2FyZSJ9LCJpc3N1ZXIiOnsidW5pdmVyc2l0eSI6IlVMQlMiLCJmYWN1bHR5IjoiRklORyJ9fX0.auauRueb4p2kbVxLcEyCldoGQD6AIU9YbL-RpDqEk60ffcqYGO919rjphbeh0ZJybovKVRTpWL8wKXFNFuWFfZEyzR3X5d6_6y9fxZoBIC46dxikuHkTCXauY0XE1Ytnh5qxkQdQSA5w_A-xD-hlzlR-Ah4U5qc4Hm8A6uTG1RAvlb3-R1ckkv4USl7kz2WC1MCQzP-bySsw6JaXNPI0mBGkbpMnt9ms2V4uBAi8CMPzTUgbUHYpcRvvcTYOIiIchTxmns-hRPbKCjA6N9pnkGT8RogSyjzxyOL1WL8Ea2R_KxXf-ULYhxgVnNG6cUj0JByFq3AhUJ5Vdxx8N43D7w

For processing the data, the validator must use the public key of the issuer to reverse the signature process back to the data to be protected. If it matches the claims provided in the QR code, they are assumed to be valid, with a very high degree of certainty.

The security is greatly enhanced for this scheme in that the validator is no longer required to know the private key used to generate other valid tokens, but only its public component that can be used exclusively for checking the integrity of the data. Thus, security threats of the system are reduced.

2.2.3. Complete asymmetric (ECDSA-based) data validation

This approach uses the asymmetric ECDSA algorithm as described in the first chapter instead of RSA. There are few high-level functional differences between the two algorithms, but ECDSA is generally preferred in that the size of the signature is lower, for a security equivalent to that provided by RSA.

The issuer passes the protected data in the JWT through the private function of the ES256 primitive, the output, of size just 128 B, being appended to the data to be included in the QR code:

eyJ0eXAiOiJKV1QiLCJhbGciOiJFUzI1NiJ9.eyJkYXRhIjp7Im5vIjozMDksInNjb3BlIjoiSW50ZXJuYWwiLCJ1bml2ZXJzaXR5WWVhciI6IjIwMjMtMjAyNCIsImRhdGUiOiIxMi8xMi8yMDIzIiwiaG9sZGVyIjp7ImdpdmVuTmFtZSI6IlZsYWQtQW5kcmVpIiwiZmFtaWx5TmFtZSI6Ik9sZWtzaWsiLCJDTlAiOiIxMjMwNDA1MDY1MDAwIiwic3R1ZHlZZWFyIjoiSSIsInByb2dyYW1tZSI6IkNhbGN1bGF0b2FyZSJ9LCJpc3N1ZXIiOnsidW5pdmVyc2l0eSI6IlVMQlMiLCJmYWN1bHR5IjoiRklORyJ9fX0.AICvWh5P1bDrOXtdyY2FcicjoR5CVHN5xtHM62-9dStgAy1RO1mnEw9ODFKdjG6QlKMbS4zxSfzbGG9LJh9Iel04AFDhen6uLZLvQFQ79iSev3EE7HFYpcRP1DxYLNf1YpaxdQiA1QRZtsEBeCn0Ha2H6rgd8L_bUWY1lUnQuOo3hcBw

The other considerations stay the same as for RSA.

DOI: 10.2478/ijasitels-2023-0008

3. Results

To assess the behaviour and performance of each of the approaches described above, sample codes have been generated using the sample data and the schemes presented in section 2. For all cases, the medium level of error correction was used.

For each code, several parameters have been recorded, including the physical space occupied by each code (determined by the QR version needed to accommodate for each of the strings to be encoded) and the bits of security provided by each of the validation algorithms and other specifications.

Using an implementation of the checksum algorithm and the cryptographic primitives previously described, codes were generated in accordance with each of the validation schemes presented above. The results can be found in Figures 1-5.

Figure 1 Minimal data encoded for checksum validation

Figure 2 Minimal data encoded for HMAC validation

DOI: 10.2478/ijasitels-2023-0008

Figure 3 Complete data encoded for HMAC validation

Figure 4 Complete data encoded for RSA validation

Figure 5 Complete data encoded for ECDSA validation

DOI: 10.2478/ijasitels-2023-0008

As it can be seen, the size of the QR codes varies widely depending on the validation scheme to be used. In accordance with this and given the increasing trend towards using QR codes on printable documents, the approach to be preferably used is largely determined by the physical space available. As such, a comparison of the space required by each approach for the data to be encoded is presented in Table 1. For the minimum printed size, the standard ratio of 12 modules/cm [16]

Table 1 Size characteristics of the resulting codes

Approach	Data size (B)	Signature relative size (%)	QR code version	Module number	Minimum printed size (cm^2)
1.	25	21.6	2	25 x 25	2.08 x 2.08
2.	48	60.0	4	33 x 33	2.75 x 2.75
3.	428	10.0	16	81 x 81	6.75 x 6.75
4.	727	47.2	22	105 x 105	8.75 x 8.75
5.	561	31.4	19	93 x 93	7.75 x 7.75

As it can be noticed, for encoding the entire data to be transmitted, the size required increases boldly, a compromise thus existing between the reliability and the physical space occupied to ensure readability of data.

The other relevant factor for such a data validation scheme is the security that it provides. The most widespread way to quantify this security would be represented by the number of bits of entropy in the key that would be required to obtain the correct key. The security inherent to each approach, along with the other defining factors of each of them, are presented in Table 2. For the checksum algorithm, since it is not designed for use with a key, a value of 0 bits of security was considered.

Table 2 Security of the resulting codes

Approach	Data included	Data size (B)	Security (b)	Algorithm type
1.	minimal	25	0	-
2.	minimal	48	~160*	symmetric
3.	complete	428	~256*	symmetric
4.	complete	727	128	asymmetric
5.	complete	561	128	asymmetric

**The level of security of HMAC is influenced by an implementation detail, making it difficult to estimate the exact information needed to break it [1] The level of security for SHA-1-HMAC can often be as low as 160 bits.*

As it can be seen in the table above, apart from the unsecure first approach, which is based on checksums, the levels of security are sufficient for all the other validation schemes, above the current, agreed upon, security threshold of 100 bits of security.

While the theoretical entropy of the keys used in symmetric algorithms is higher, their security is otherwise hindered by the need for disclosing the key used for issuing new data to be validated.

DOI: 10.2478/ijasitels-2023-0008

An additional factor to consider when evaluating these approaches is the time elapsed for each process (generation and validation, respectively). As such, the generation and the validation rounds were each timed with a microsecond-precision timer, the average results for each algorithm being presented in Table 3.

Table 3 Time elapsed for processing the resulting codes

Approach	Algorithm	Generation time (µs)	Verification time (µs)
1.	CRC32	<1	<1
2.	SHA-1-HMAC	<1	<1
3.	SHA-256-HMAC	1	1
4.	RSA	42823	644
5.	ECDSA	3098	2184

These results are represented in the logarithmic-linear chart in Figure 6. As it can be seen from the table above and the chart, the only significant delays are produced by the asymmetric signature algorithms.

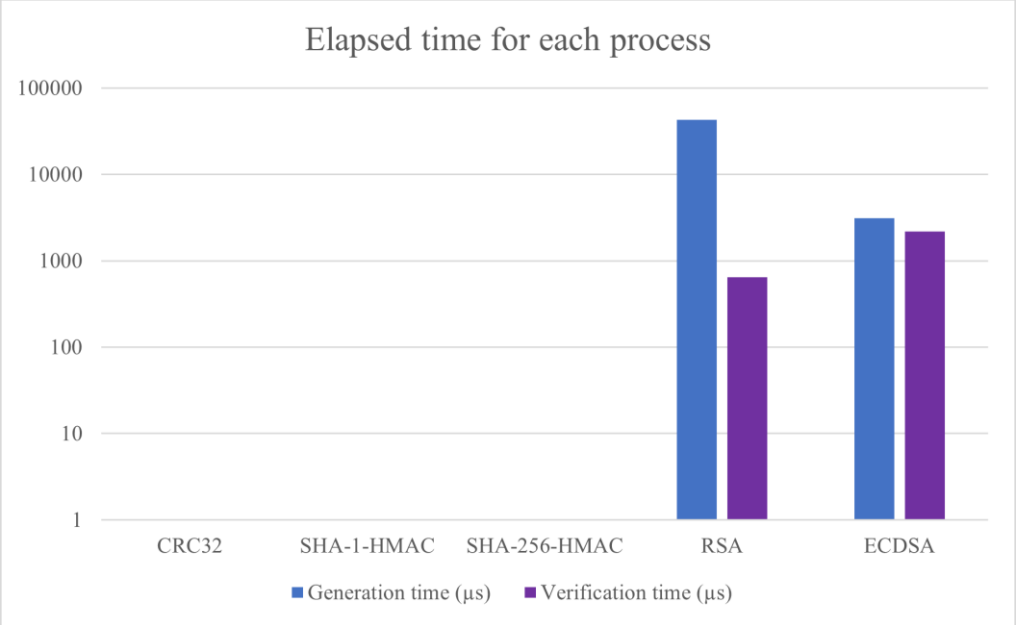

Figure 5 Complete data encoded for ECDSA validation

Notably, while the generation of the signature takes a very long time for RSA, ECDSA has a much higher signature speed, although the verification process is slightly slower than for RSA. However, as these two algorithms considerably enhance security, the time cost using asymmetric algorithms is justified.

4. Conclusions

The present paper has presented multiple alternatives for streamlined data handling using QR code-based data validation.

Although there are many factors whose influence on the optimum approach highly depends on the circumstances that describe the intended use of the validation flow, these approaches generally cover most practical use-cases.

DOI: 10.2478/ijasitels-2023-0008

For encoding only minimal data, the HMAC-based approach should not be used as the last verification layer, it is very effective in ensuring the integrity of data even against relatively qualified, but not large-scale tampering or attacks.

On the other hand, where the physical space available for the QR code is not a very important constraint, the ECDSA-signed JSON Web Token represents an asset in that it allows for both privacy (the data is stored with the holder) and transparency (anyone can verify the authenticity of the data). Thus, this scheme is currently being introduced in an increasing number of official documents in the worldwide process of digitalisation [20]

5. References

[1] Bynens, M, PBKDF2+HMAC hash collisions explained. https://mathiasbynens.be/notes/pbkdf2-hmac (March 25, 2014).

[2] Davies, J. Understanding CRC32. https://commandlinefanatic.com/cgi-bin/showarticle.cgi?article=art008 (December 10, 2023).

[3] De Groot, J., What Is Data Encryption? (Definition, Best Practices & More). https://www.digitalguardian.com/blog/what-data-encryption (December 10, 2023).

[4] Fisher T., What Is a Checksum? See a Definition, Examples, and More. Lifewire. https://www.lifewire.com/what-does-checksum-mean-2625825 (December 10, 2023).

[5] Hetler, A, Common Social Media Privacy Issues. WhatIs.com. https://www.techtarget.com/whatis/feature/6-common-social-media-privacy-issues (October 14, 2023).

[6] Kerner, S., M., What Is Data Validation? Data Management. https://www.techtarget.com/searchdatamanagement/definition/data-validation (December 9, 2023).

[7] Loshin, P., What Is Encryption and How Does It Work? Security. https://www.techtarget.com/searchsecurity/definition/encryption (December 10, 2023).

[8] Neri, A , 'We Should Treat Data as a Natural Resource. Here's Why'. World Economic Forum. https://www.weforum.org/agenda/2020/03/we-should-treat-data-as-a-natural-resource-heres-why/ (December 10, 2023).

[9] 'https://en.wikipedia.org/w/index.php?title=Adler-32&oldid=1146207638 (December 10, 2023).

[10] What Is Data Validation?' Astera. https://www.astera.com/knowledge-center/what-is-data-validation/ (December 9, 2023).

[11] Fundamental Difference Between Hashing and Encryption Algorithms | Baeldung on Computer Science. https://www.baeldung.com/cs/hashing-vs-encryption (December 10, 2023).

[12] Encryption: ECDSA vs. RSA Keys | Baeldung on Computer Science. https://www.baeldung.com/cs/encryption-asymmetric-algorithms (December 10, 2023).

[13] Fletcher's Checksum. https://www.tutorialspoint.com/fletcher-s-checksum (December 10, 2023).

DOI: 10.2478/ijasitels-2023-0008

[14] https://clarvision.ro/referinte/arsat/ (December 27, 2022).
[15] What Is a Checksum? An Easy-to-Understand Checksum Definition. Code Signing Store. https://codesigningstore.com/what-is-checksum-how-it-works (December 10, 2023).
[16] QR Code Model 1 and Model 2: Point for setting the module size. https://www.qrcode.com/en/howto/cell.html (May 4, 2013).
[17] The JSON Data Interchange Syntax. https://www.ecma-international.org/wp-content/uploads/ECMA-404_2nd_edition_december_2017.pdf (December, 2017)
[18] What Is RSA? How Does an RSA Work? | Encryption Consulting. https://www.encryptionconsulting.com/education-center/what-is-rsa/ (December 10, 2023).
[19] Data Protection under GDPR. https://europa.eu/youreurope/business/dealing-with-customers/data-protection/data-protection-gdpr/index_en.htm (December 10, 2023).
[20] eHealth Network Guidelines on Technical Specifications for EU Digital COVID Certificates | Volume 2'. https://health.ec.europa.eu/system/files/2022-07/digital-covid-certificates_v2_en.pdf (June 15, 2022).
[21] What Is Data Encryption? https://www.forcepoint.com/cyber-edu/data-encryption (December 10, 2023).
[22] What Is Encryption and How Does It Work?' Google Cloud. https://cloud.google.com/learn/what-is-encryption (December 10, 2023).
[23] What Is Encryption? Data Encryption Defined. https://www.ibm.com/topics/encryption (December 10, 2023).
[24] What Is Data Validation: Definition. https://www.informatica.com/services-and-training/glossary-of-terms/data-validation-definition.html (December 9, 2023).
[25] 'HMACSHA1 Class (System.Security.Cryptography)'. https://learn.microsoft.com/en-us/dotnet/api/system.security.cryptography.hmacsha1?view=net-8.0 (December 10, 2023).
[26] Hashing Algorithm Overview: Types, Methodologies & Usage. https://www.okta.com/identity-101/hashing-algorithms/ (December 10, 2023).
[27] OpenID Connect Basic Client Implementer's Guide 1.0 - draft 46'. https://openid.net/specs/openid-connect-basic-1_0.html (October 28, 2023).
[28] veritas.com. 'What Is RSA Encryption?' https://www.veritas.com/information-center/rsa-encryption (December 10, 2023).