Performance comparison of signed algorithms on JSON Web Token

A Rahmatulloh, R Gunawan and F M S Nursuwars
Department of Informatics, Siliwangi University
Jln. Siliwangi No. 24, Tasikmalaya 46155, Indonesia

Email: alam@unsil.ac.id, rohmatgunawan@unsil.ac.id, firmansyah@unsil.ac.id

Abstract. The system survive in this digital era is a system that can work on multiple platforms, the use of web services is one of the solution. Exchange data using JSON format and for the security of authentication using JSON Web Token (JWT). The importance of token-based authentication using JWT on web services can solve interoperability problems. JWT is stateless and can include information in the token authorization. JWT has several options for using algorithms namely HMAC, RSA, and ECDSA. Unfortunately, it is unknown which algorithm has better performance. Here, we directly tested the signing algorithm in the three algorithms seen from several parameters. The experimental results showed the use of HMAC algorithm produces an average value of token-generating time is 21.3 s, token size 109 bytes and data transfer token speed 91.2 s. It was considered that the HMAC had an excellent performance.

1. Introduction
Current technology trends are stateless systems and can be used multi-platform so that interoperability problems can be resolved. Interoperability problems can be solved such as using web services. Web service (WS) uses a service-oriented architecture (SOA) concept that provides services on networks such as the web, and these services contain well-defined business functions which can then be consumed by users in various applications[1, 2]. SOA which has better performance is the Representational State Transfer (REST) model [3-4]. REST is a standard web-based service architecture for data communication using HTTP protocol standards [5, 6]. However, REST is still weak on the security side [7], securing REST including securing data and data communication lines to protect confidentiality and integrity data [8].

Various techniques for secure the REST architecture have been carried out by several researchers, Hussain (2014) developing an application programming interface (API) on the web using ASP.NET based on the view controller (MVC) model, using basic authentication, claim based authentication, and token based [9]. Likewise with research conducted by Huang (2015) and Sahoo (2017) using token-based authentication [10,11]. Authentication based on ID claims has been done by Xinhua (2013) [12] while the Security Assertion Markup Language (SAML) technology was conducted by Anand (2017) [13]. Authentication is a standard aspect of information, authenticating one of the most critical parts in each application. Users can be authenticated to get resources using passwords, biometrics, token-based or through certificates [14]. For decades, cookies and server-based authentication are the most comfortable solutions for authentication in web-based applications. However, it will be different when handling authentication for multi-platform, the use of server-based authentication will be very
complicated. Session-based authentication always makes the server overload, because the server must check the session all the time it reduces server performance. Unlike token-based authentication, there are no sessions created every time the user enters so this shows better performance than session-based techniques [15]. REST security using JSON Web Token (JWT) authentication has been conducted by Balaj (2017) [16].

Token-based authentication enables the credentials of the user's name and password to get tokens that allow them to retrieve certain resources. After a token is available, users can reuse the same token to access resources without using a username and password again [17]. The given token has been made by the server with a certain algorithm so that we can know the ownership of tokens and access rights owned by the user.

JWT is essentially part of the data signed in JSON format. JWT can be signed using an HMAC symmetric key algorithm or public and private key pair using RSA or ECDSA [18]. Research only compares the performance of the HMA SHA-256 algorithm with the SHA-512, there has been no performance testing of the algorithm options that JWT can be used [16]. Therefore, the focus of this research was to test the algorithm performance on JWT and compare the performance of the three algorithms in the parameters of the speed of generating tokens, the size of tokens and data transfer tokens.

2. Method

2.1. Representational state transfer (REST)

REST is not a programming language, but rather a hybrid architectural style derived from several network-based architectural styles that are often applied in web-based services [5]. REST architecture is generally run via HTTP (Hypertext Transfer Protocol), involving the reading of certain web pages that contain an XML or JSON file. REST is not stateless and resource-oriented, everything in the REST architecture is a resource. Each request is independent, and the server does not store any request conditions. The Application Programming Interface (API) that follows the REST style is called the RESTful API. The RESTful API uses Uniform Resource Identifiers (URIs) to represent resources. Each data source is identified using a URI link. Some of the methods used in the REST such as the GET method is used to obtain resources, and the POST method is used to create new resources, the PUT method is used to update resources based on resources, and the DELETE method is used to delete resources or collections of resources. Examples can be seen in table 1.

| Resource | Method |
|----------|--------|
| /api/student | Get a list of all student |
| /api/student/123 | Get a student by student’s ID |
| | Create a new list of student. Treat as a collection. Create a new student in it. |
| | Update a list of student If a student exists, update the student. If a student does not exist. Create a new student. |
| | Delete all student Delete the student. |

2.2. Token authentication

Token authentication or token-based authentication is a core element of scalable identity and authorization management. Authentication tokens are stateless, secure, scalable, and designed to ease users without overloading the server. Authentication is the process of identifying the user's identity. The traditional mechanism of an application will save the session ID on the server containing the user's identity from the session cookie. Unlike token authentication, it applies a modern approach designed to solve session ID problems, use tokens as a substitute for session IDs that can reduce server load, manage
permissions, and as a new, better mechanism to support distributed or multi-platform and cloud-based architectures.

Figure 1. Difference between traditional and modern authentication tokens.

Figure 1 shows the authentication process between client and server, in modern authentication based on the user's credential token exchanged with a token that then the token can be used every subsequent request. Tokens are attached to users via cookies or stored locally or browser cookies.

2.3. JSON Web Token (JWT)

JWT or read 'jot' is a token in the form of a string consisting of three parts, namely header, payload, and signatures used for authentication and information exchange processes [18]. Each part is separated by a dot symbol (.) as depicted in figure 2.

Figure 2. Three main elements.

JWT encodes claims to be sent as JSON objects that are used as load structures from JSON Web Signature (JWS) or as a plaintext structure of JSON Web Encryption (JWE), allowing claims to be digitally signed or protected with Message Authentication Code (MAC). According to how tokens are presented there are two types of tokens, Carrier Tokens, and Key Holder Tokens and according to the token purpose, there are two token schemes, Identity Tokens, and Access Tokens [19]. JWT works the same way as a password when the user successfully logs in, and the server will give a token stored in local storage or browser cookies. The token is used to access certain pages, the user will send the token back as proof that the user has successfully logged in. The JWT structure can be seen in figure 3.

In the Header section consists of two parts: the algorithm used and the type of token, then JSON is encoded using Base64. The second part is the payload that contains the claim, and the claim is a statement about the entity which is usually additional user data and data. The payload is the same as the encoded header with Base64. The third part is the signature containing the encoded header, the encoded payload, the secret, the algorithm that has been defined in the header and the sign. Cryptographic algorithms for digital signatures and MACs at JWT can be seen in table 2.
**Header**: Algorithms and Token Types

```
{  
  "alg": "HS256",  
  "typ": "JWT"  
}
```

**Payload**: Data

```
{  
  "iss": "#appbackend",  
  "user": "alam",  
  "pass": "rahasia"  
}
```

**Signature**: results from Hash

```
{  
  "Base64-encoded(Header.Payload)" + "key" + "Algorithm"  
}
```

**Figure 3.** JWT structure.

In the Header section consists of two parts: the algorithm used and the type of token, then JSON is encoded Base64. The second part is the payload that contains the claim, and the claim is a statement about the entity which is usually additional user data and data. The payload is the same as the encoded header with Base64. The third part is the signature containing the encoded header, the encoded payload, the secret, the algorithm that has been defined in the header and the sign. Cryptographic algorithms for digital signatures and MACs at JWT can be seen in table 2.

**Table 2.** Signing algorithm [20].

| "alg" parameter value | Digital Signature or MAC Algorithm | Implementation Requirements |
|------------------------|------------------------------------|-----------------------------|
| HS256                  | HMAC using SHA-256                 | Required                    |
| HS384                  | HMAC using SHA-384                 | Optional                    |
| HS512                  | HMAC using SHA-512                 | Optional                    |
| RS256                  | RSASSA-PKCS1-v1_5 using SHA-256    | Recommended                 |
| RS384                  | RSASSA-PKCS1-v1_5 using SHA-384    | Optional                    |
| RS512                  | RSASSA-PKCS1-v1_5 using SHA-512    | Optional                    |
| ES256                  | ECDSA using P-256 and SHA-256      | Recommended+                |
| ES384                  | ECDSA using P-384 and SHA-384      | Optional                    |
| ES512                  | ECDSA using P-521 and SHA-512      | Optional                    |
| PS256                  | RSASSA-PSS using SHA-256 and MGF1 with | Optional                |
| PS384                  | SHA-256                            | Optional                    |
| PS512                  | RSASSA-PSS using SHA-384 and MGF1 with | Optional                |
| none                   | SHA-384                            | Optional                    |

The last stage of the three sections in figure 3 merged to produce JWT, with pseudocode (1).

\[ \text{Token} = f(Base64Encode) \sum_{n=\alpha,\beta}^\infty (\text{header.p} \text{ayload.s} \text{ignature}) \quad \text{(1)} \]

With tokens obtained from the server, users can access web service resources by entering tokens in the HTTP header in the authorization method. The JWT tokens example can be seen in figure 4.
2.4. Related work
The use of token-based authentication using JWT has already been investigated by Ethelbert (2017) which results in that token-based authentication using JWT is superior to the others seen from the Authentication mechanism, Access control mechanism, Compact & stateless token structure, Dual authentication / SSO support, and access control scalability [21]. Then in the study of Mestre (2017) that compares JWT performance with single and multiple tokens on one or several servers. The results show that the proposed system has a worse performance than those using a single token [22]. Research produces a comparison of the performance of SHA-512 better than SHA-256 in the HMAC hash algorithm alone [16]. While the general algorithm choice that is often used on JWT is HMAC + SHA256, RSASSA-PKCS1-v1_5 + SHA256 or ECDSA + P-256 + SHA256 but no one has yet tested its performance [18].

3. Results and Discussion
A simple simulation application has been created to find out the algorithm performance on JWT. Then the test scenario with experiment 50 times the process, starting from testing the time to generate the token, the size of the token generated and finally the data transfer speed of the token from the client request to the server until the token response is received by the client.

3.1. Generate token
The first test is to generate a token process 50 times and in table 3 only ten experiments are presented, but the results of the test are all poured into a graph in figure 5.

![Figure 4](image.png)

**Figure 4.** The example of JWT tokens.

![Figure 5](image.png)

**Figure 5.** Comparison performance of time process generating for tokens using RSA, ECDSA, and HMAC algorithms.

### Table 3. Generate token processing time.

| Testing | Signing Algorithm | Testing | Signing Algorithm |
|---------|-------------------|---------|-------------------|
| 1       | 277762            | 114581  | 21171             |
| 2       | 271418            | 81652   | 21414             |
| 3       | 278841            | 86963   | 15092             |
| 4       | 252384            | 90251   | 14371             |
| 5       | 262182            | 126767  | 26075             |
| 6       | 263163            | 100349  | 17970             |
| 7       | 251739            | 86033   | 18839             |
| 8       | 243438            | 90439   | 51525             |
| 9       | 271581            | 101983  | 34792             |
| 10      | 262023            | 96329   | 22439             |
| Avg     | 268290            | 95845   | 21322             |

3.2. Token size
The second test is testing the JWT token size, the results of the experiment 50 times can be seen in table 4 and graphical form in figure 6.
Table 4. Token size.

| Testing | Signing Algorithm | RSA (bytes) | ECDSA (bytes) | HMAC (bytes) |
|---------|-------------------|-------------|---------------|--------------|
| 1       |                   | 164         | 164           | 109          |
| 2       |                   | 164         | 164           | 109          |
| 3       |                   | 164         | 164           | 109          |
| 4       |                   | 164         | 164           | 109          |
| 5       |                   | 164         | 164           | 109          |
| 6       |                   | 164         | 164           | 109          |
| 7       |                   | 164         | 164           | 109          |
| 8       |                   | 164         | 164           | 109          |
| 9       |                   | 164         | 164           | 109          |
| 10      |                   | 164         | 164           | 109          |

Figure 6. Comparison of tokens size using RSA, ECDSA, and HMAC algorithms.

3.3. Data transfer
Likewise, testing all three trials of transferring token data 50 times from client to server can be seen in table 5 and figure 7.

Table 5. Data transfer.

| Testing | Signing Algorithm | RSA (ms) | ECDSA (ms) | HMAC (ms) |
|---------|-------------------|----------|------------|-----------|
| 1       |                   | 368412   | 190990     | 83660     |
| 2       |                   | 339241   | 144385     | 91155     |
| 3       |                   | 359306   | 151147     | 74678     |
| 4       |                   | 311298   | 152766     | 77904     |
| 5       |                   | 322892   | 218130     | 100363    |
| 6       |                   | 319571   | 163614     | 78579     |
| 7       |                   | 315724   | 147288     | 79581     |
| 8       |                   | 301652   | 163899     | 138004    |
| 9       |                   | 379141   | 202094     | 110295    |
| 10      |                   | 320035   | 162519     | 93199     |

Avg 339676 173532 91180

Figure 7. Comparison performance time of token data transfer using RSA, ECDSA, and HMAC algorithms.

If we compare the performance of the token generating time (table 3 and figure 5) we can see that the HMAC algorithm is superior to an average score of 21.3s, there is an increase of 349.5% when compared to ECDSA while with RSA an increase of 1158.3%. Viewed from token size (table 4 and figure 6) HMAC produces 109 bytes token size when compared to ECDSA and RSA increased by 50.5%. The final test results (table 5 and figure 7) in HMAC token transfers received an average score of 91.2s, while an increase of 90.3% in ECDSA and 272.5% in RSA.

4. Conclusions
The results of the research in this paper have been explained about the comparison of token-based authentication performance using JWT with several algorithms. The overall results show that the use of HMAC is superior to the parameters of time generate token, token size and token transfer speed.
However, there has been no discussion and testing when viewed from other parameters such as testing the security of tokens.

JWT is stateless so the server does not need to store data because data such as scope or authorization and a user can all be stored in JWT. This concept is very suitable if it is applied to a centralized authentication system that uses single sign-on. For future work, it needs to be tested regarding the mechanism for storing tokens on local storage (cookies), whether it is safe and how to save the correct token.

References
[1] Fielding R T and Taylor R N 2002 Principled design of the modern Web architecture ACM Transactions on Internet Technology (TOIT) 2(2) 115–150
[2] Shofa R N, Aradea and Kurnia B B 2013 Penerapan Service Oriented Architecture (SOA) Dalam Pembangunan Web Based Learning, Jurnal Penelitian SITROTIKA 9(2) 221–19
[3] Rathod D M 2017 Performance evaluation of restful web services and soap / wsdl web services, International Journal of Advanced Research in Computer Science 8(7) 415–420
[4] Memeti A, Imeri F and Cico B 2017 REST vs. SOAP: Choosing the best web service while developing in-house web applications, Journal of Natural Sciences and Mathematics of UT 2(3) 63–8
[5] Fielding R T 2000 Architectural Styles and the Design of Network-based Software Architectures Dissertation (Irvine: University of California) 1–180
[6] Fielding R and Reschke J 2014 Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content, Internet Engineering Task Force (IETF) (https://tools.ietf.org/html/rfc7231)
[7] Kumari V 2015 Web Services Protocol: SOAP vs REST IJARCET 4(5)
[8] Kanmani K and Smitha P S 2013 Survey on Restful Web Services Using Open Authorization, IOSR-JCE 15(4) 53–6
[9] Hussain M I and Dilber N 2014 Restful web services security by using ASP.NET web API MVC based JISRC 12(1) 4–10
[10] Sahoo P, Janghel N K and Samanta D 2017 Securing WEB API Based on Token Authentication, IJAECE 4(2) 1–4
[11] Huang X-W, Hsieh C-Y, Wu C H and Cheng Y C 2015 NBIS ’15 Proc. 2015 18th Int. Conf. Network-Based Information Systems (USA: IEEE Computer Society) 601–6
[12] Li X 2013 The Design of Digital Campus Unified Identity Authentication System Based on Web Services Applied Mechanics and Materials 427–429 2301–4
[13] Indu I, Rubesh-Anand M R and Bhaskar V 2017 Encrypted Token based Authentication with Adapted SAML Technology for Cloud Web Services Journal of Network and Computer Applications 99(C) 131–45
[14] Pavlovski C, Warwar C, Paskin B and Chan G 2015 Unified Framework for Multifactor Authentication, 22nd International Conference on Telecommunications (ICT 2015) 209–13
[15] Balaj Y 2017 Token-Based vs Session-Based Authentication: A survey 1–5
[16] Rahmatulloh H, Sulastri and Nugroho R 2018 Keamanan RESTful Web Service Menggunakan JSON Web Token (JWT) HMAC SHA-512, Jurnal Nasional Teknik Elektro dan Teknologi Informasi (JNTETI) 7(2) 131–7
[17] Bhawiyuga A, Data M and Warda A 2017 Architectural Design of Token based Authentication of MQTT Protocol in Constrained IoT Device, 2017 11th International Conference on Telecommunication Systems Services and Applications (TSSA), 1–4
[18] Jones M, Sakimura N and Bradley J 2015 RFC 7519: JSON Web Token (JWT) (https://www.heise.de/netze/rfc/rfc/rfc7519.shtml)
[19] Zheng K and Jiang W 2014 A Token Authentication Solution for Hadoop Based on Kerberos Pre-Authentication, 2014 International Conference on Data Science and Advanced Analytics (DSAA) 354–60
[20] Jones M 2015 JSON Web Algorithms (JWA), Internet Engineering Task Force (IETF)
[21] Ethelbert O, Moghaddam F F, Wieder P and Yahyapour R A 2017 JSON Token-Based Authentication and Access Management Schema for Cloud SaaS Applications, in 2017 IEEE 5th International Conference on Future Internet of Things and Cloud (Cornel University Library) p 6

[22] Mestre P, Madureira R, Melo-Pinto P and Serodio C 2017 Securing RESTful Web Services using Multiple JSON Web Tokens, in Proc. World Congress on Engineering 2017 418–23