An Authentication Based Scheme for Mobile Applications Using THJWT

Sarvesh Chopra [a], Amritpal Singh [a], Aman Singh [b]

[a] *Department of Computer Science, Lovely Professional University, Punjab, India*
[b] *Faculty of Engineering, Universidade, Cuito-Bie, Angola*

Abstract
Information and Communication Technology (ICT) integrates multiple devices and applications that are connected directly to the Internet communications. The development of customer server systems in technology, such as IoT based, cloud based, and smart homes systems is growing rapidly in the current era of technology. User authentication is an important concern for these applications. THJWT i.e., Tuned Hybrid JSON Web Token for JWT-based properties has been particularly evident in the recent growth of client server requests. JWTs (JSON Web Token) are used for authentication of subsequent customer requests without making regular calls to a resource server or database. In this paper, we have introduced the process of authenticating and verifying JWT on each client request based on the random stamp values to verify client authentication on the server data. The effectiveness of the proposed approach is enhanced by case studies that demonstrate time and space complexity.

Keywords
Authentication, Security, Client Authentication, JSON Web Token, THJWT call authentication

1. Introduction

Security and privacy of user data is a prime concern in today's digital world. User authentication, access to services and access time in terms of IoT, mobile, web and cloud-based applications [4] are considered important concerns regarding network security. While communicating with clients, servers maintain customer identity during customer-server interaction. If vital security checks are not performed to protect client identity, it can lead to serious misuse of personal data. Real data access becomes complicated when the same user access a resource server from a different device such as mobile phones, tabs, personal computers etc. using a different operating system such as such as IOS, Android, Windows, Linux, etc.

IOT devices such as actuators, gadgets, sensors, radio frequency identification (RFID) tags, and communication technologies can be integrated together and a variety of physical entities and devices can be connected to the Internet to permit those objects to communicate with each other.

From Data Analysis to healthcare and manufacturing, IOT consists mostly of little materials that are provided with unique identifiers (UIDs) with the ability to send data over a network without requiring user to user or user to computer interaction. That is why, IOT is the backbone of a grand technological

WCNC-2022: Workshop on Computer Networks and Communications, April 22 – 24, 2022, Chennai, India.
EMAIL: er.sarveshchopra@gmail.com (Sarvesh Chopra)
ORCID: 0000-0002-4061-0191 (Sarvesh Chopra)

future. IOT is getting major focus in almost every business sector and, it is a fact that it has already started having its influence on web app design & development. Further, due to COVID-19 our dependency on such systems has increased exponentially.

Although IOT devices have a lot of significance to our busy world but we need to the secure the data generating from these devices and must have a special check on the security of such devices. The binary data that is sent over these networks is always important and vulnerable to attack.

Thus, user data must be secured to protect the privacy and confidentiality of the registered user. If there is no check, then threats like data tempering, data breaching and theft of personal information can be there. Hence, some of the crucial terms associated with IOT security involve identification and authentication. The process of identifying a user as being the claimed/registered person is called user authentication. The server can provide different roles or privileges to various authorized users. But in IOT things are little different, there may be a case where a particular node may act as a client for one node and a server or Authentication Agent for some other, hence a better security system is required.

A web application (or "web app") is an application that runs on any web browser and performs various tasks with the help of Internet. The user-facing component of an IOT system is called the interface of the application. The app may run on any Operating System such as IOS or Android which can be installed on a laptop, smartphone or a tablet. Data stored at the sever side is used for access and verification. However, checking the database server for every request would cause a serious performance bottleneck.

2. User Authentication Schemes

2.1 Traditional Approach

Old way of providing security is through methods which store a session on each client's machine (browser) with client authentication on the server. As long as the session is active, the client can send the request to access the server resources and server responses to the request using the session key. Time keys are assigned to clients on initial server request after verification of ownership from the database server. The client keeps the key assigned to the cookie for ongoing communication with the server. The client sends the same session key to the server for verification for each request. Sever will only look for a valid session key from the client for client authentication instead of asking for details again. The weakness of this machine is to easily hijack session keys by attackers. Hackers may use various tools to identify a malicious power session and gain access to current user sessions. Various extensions, tools and plugins can be used to transfer cookie and key information to third-party applications.

2.2 SOAP based OAuth authentication in APIs

OAuth is a protocol which can grant one application the credentials it requires to access data and information in another application through a web service. Web services allows SOAP to provide a robust programming model that helps in the following manner:

The SOAP (Simple Object Access Protocol) is a messaging protocol specification which is based on XML to encode requests and responses in case of a web service. It can be used to handle the following scenarios:

1. Calling a method on a service.
2. To get a response from a service method, and handling its return value and out parameters.

3. Handling errors.

Limitations of SOAP: SOAP can only work on XML data, this is one of the biggest reasons why a SOAP API will be less effective, as XML is a verbose and heavy format compared to JavaScript Object Notations (JSON data). In case of SOAP, API calls made to your server will require more network resources such as bandwidth and heavy processing which will take more time to authenticate the client. In SOAP, tight coupling is implemented as the client-server communication depends on WSDL (Web Service Description Language) contracts. Therefore, SOAP is not recommended for applications which are loosely coupled. SOAP is harder to code, and can't be tested in the web browser (as opposed to REST where we have response codes). Further it is very difficult to make contracts in WSDL, create client stubs as it follows strict behavior.

2.3 JWT based authentication in APIs

The JSON Web Token (JWT) is a secure way of representing a collection of information between two parties. JWT has been used in various applications to maintain customer authentication while communicating with the server. Various development tools and frameworks such as Passport, OpenID Connect, Django REST framework, DNN, Arengu, etc have integrated support for JWT. JWT offers various methods through which we can authenticate a user and a token can be used to access the server resources. These methods can be misused by attackers to gain access to server resources.

In this paper, we present a focused approach for detecting and accessing a server resource with a JWT predictor attack during client-server interaction and distinguishing between valid and invalid customer requests after a user's registered credentials are not matching. A case study is conducted in which a new JWT is developed based on each client request containing either a combination of token with an authorized unique key which can be a registered username or password. This proposed approach contributes to eliminating JWT predictors by attackers. JWT defines a secure way to transmit information between parties as a JSON object. This information can be verified and trusted because it is digitally signed. Here is how it works:

A JSON Web Token consists of a header section, payload section, and a signature section in base64url encoding, separated by dots (periods), given as follows:

HEADER.PAYLOAD. SIGNATURE

This is how a JWT actually looks like:

*eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9**.**eyJuYW1lIjoiSm9obiBEb2UiLCJ1c2VX25hbWUiOiJqb2huLm RvZSIsImlzX2FkbWluIjpmYWxzZX0**.**fSppjHFaqlNcpK1Q8VudRD84YIuhqFfA67XkLam0_aY*

1. The header contains metadata about the token, such as the algorithm used for the signature and the type of the token (which is simply JWT). For this example, the header before encoding is:

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

2. The payload contains information (claims) about the entity (user) that is going to be verified by the application. Our sample token includes the following claims:

{

"name": "Sarvesh",

"user_name": "Sarvesh_ch",

"is_admin": false

}

3. Finally, to generate the signature, we have to apply base64url encoding and sign the whole thing using a secret (for symmetric encryption) or a private key (for asymmetric encryption), depending on the algorithm specified in the header. We can use any algorithm like SHA256 in the header, which is a symmetric algorithm, so the encoding and signing operation would be:

HMACSHA256(base64UrlEncode(header) + "." +base64UrlEncode(payload), secret)

This gives us the following signature, which is then appended (after a dot) to the base64url-encoded headerand payload:

fSppjHFaqlNcpK1Q8VudRD84YIuhqFfA67XkLam0_aY

Generally, a server uses same JWT for the all requests until user logs out from system. There is a need to propose a generalized solution for all type of applications. The generalized solution should secure JWT to eliminate vulnerability in client authentication. Hence, we are proposing the same as follows:

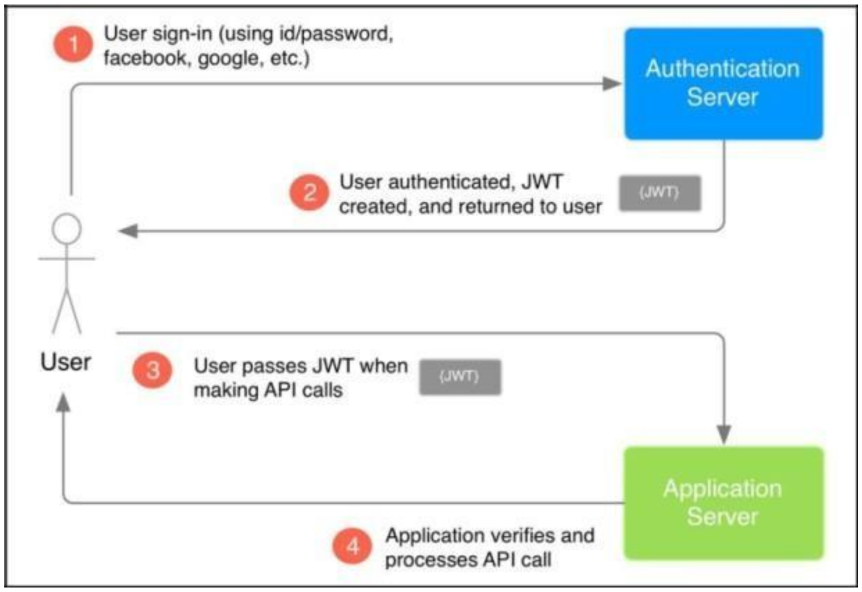

1 User sign-in (using id/password, facebook, google, etc.)
Authentication Server
2 User authenticated, JWT created, and returned to user
(JWT)
User
3 User passes JWT when making API calls
(JWT)
Application Server
4 Application verifies and processes API call

Figure 1: How JWT Works

Table 1: JWT Authentication Techniques

Algorithm Value	Digital Signature or MAC Algorithm

HS256	HMAC using SHA-256
HS384	HMAC using SHA-384
HS512	HMAC using SHA-512
RS256	RSASSA-PKCS1-v1_5 using SHA-256
RS384	RSASSA-PKCS1-v1_5 using SHA-384
RS512	RSASSA-PKCS1-v1_5 using SHA-512
ES256	ECDSA using P-256 and SHA-256
ES384	ECDSA using P-384 and SHA-384
ES512	ECDSA using P-521 and SHA-512
PS256	RSASSA-PSS using SHA-256 and MGF1 with SHA-256
PS384	RSASSA-PSS using SHA-384 and MGF1 with SHA-384
PS512	RSASSA-PSS using SHA-512 and MGF1 with SHA-512

3. Problem Statement

The two major problem statements are as follows:

1. Any change in the user role causes a significant value loss after the allocating the token to client. It can be hacked to exploit the permissions user was given in the first attempt. A resource on server should be secure and must not be accessible to the unauthorized users or the users whose permissions are revoked.
2. Access token or the JWT remains same in most of the cases (get/post/put request and response) happening between client and server interaction. Attacker can predict this vulnerability and can temper the JWT.

4. Proposed Solution

The following steps can be used to authenticate client on a server through JWT:

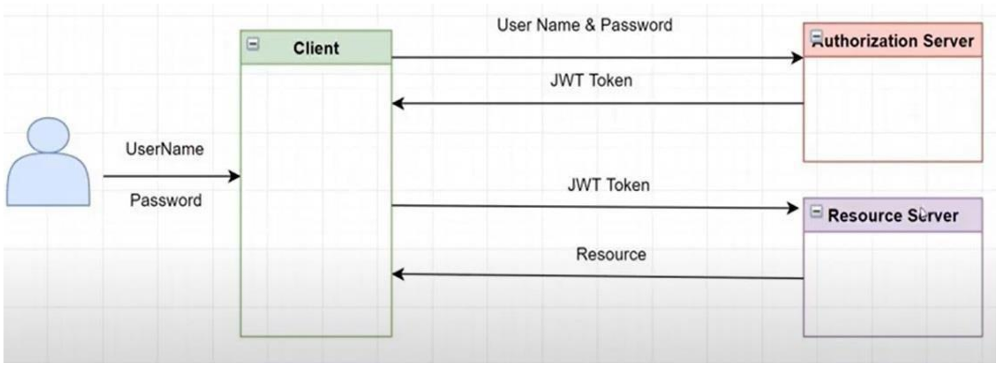
- The client will send an authentication request by credentials, such as a username– password combination for login.
- After a valid login, the authentication service will create a JWT with a secret key. This secret key can be a combination of username or any other registered attribute.
- The client uses this token with the request to a secured resource on server.
- When the server receives the request from client again, it checks the user authentication from JWT for reorganization of authentic request along with a check on the secret key. Server will send a proper response based on valid or invalid token.

The key feature of JWT is that it allows for flexibility in communication, so the client cannot change the details contained in the token. On the other hand, if the user role is updated it will be displayed or predictedby the server for any vulnerability in the next application. Under normal circumstances, clients cannot force you to forget an old JWT token and use a new token.

With every new THJWT (Tuned Hybrid JSON Web Token) for each resource request with an authenticatedidentity just to make things harder to predict for the attacker, So, let us assume T is the token and `T is the combination of Token and unique identity value. When the signature and secret key combination is calculated it will always generate a unique random-access token (JWT). This token will be returned to the client in response to each client's request.

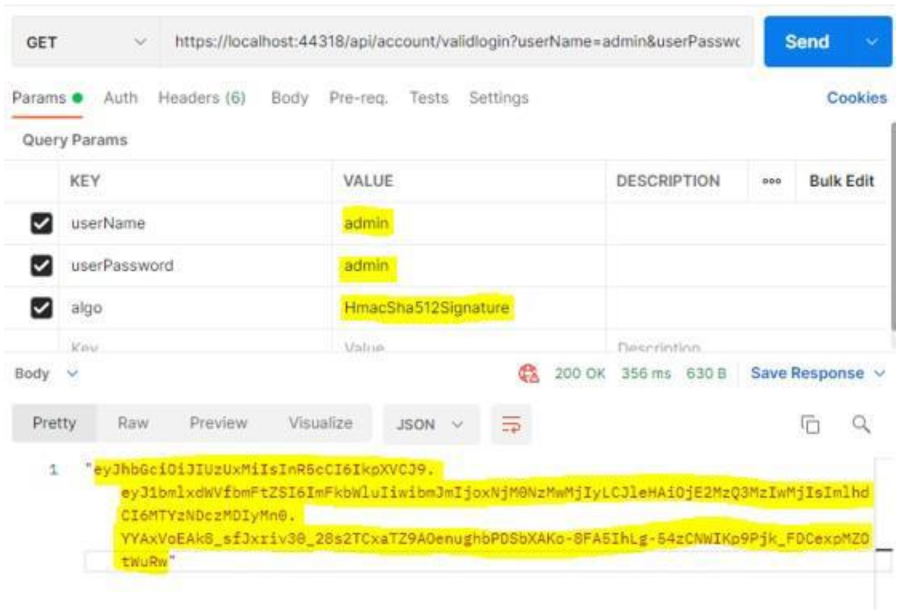
Here are the steps, that will enhance the authenticity of a client on sever:

Step 1: Client makes a request to server with login credentials.
Step 2: Server receives a request and verifies the credentials from server. If credentials are verified, then go to step 3 otherwise the control moves to step1.
Step 3: Server generates JWT by combining a value (it may be any registered user credential). T=token,T'=token+ (Unique registered credential, can be username)
Step 4: Server will receive a request from client with T' (Token + Unique registered username).
Step 5: Server will check the token information first if it is not valid, then return Response Code-401(Unauthorized Token) else move to 6.
Step 6: Server will check the unique registered username if it is not valid-return response code-401(Unauthorized user). If both 5 & 6 are valid it will move to Step-7 else to Step-1.
Step 7: Server will allocate or provide the resources as per request.



5. Mechanism

- Server will generate trigger for JWT token with the Help of login credentials (Username and password) and type of token method.

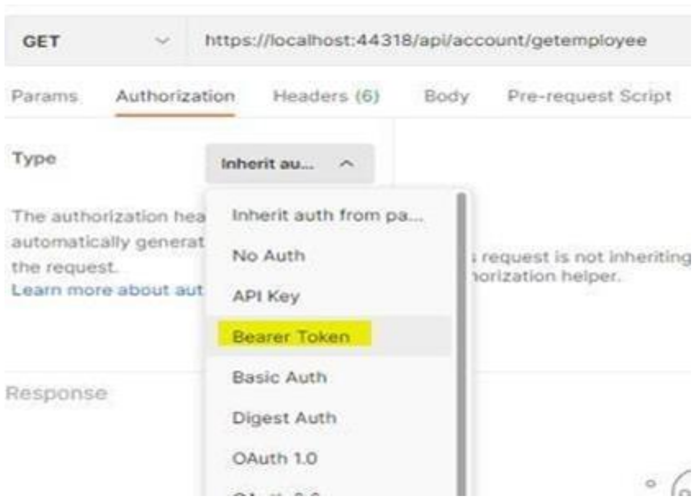


KEY	VALUE	DESCRIPTION
userName	admin	
userPassword	admin	
algo	HmacSha512Signature	
Key	Value	Description

```
"eyJhbGciOiJIUzUxMiIsInR5cCI6IkpXVCJ9.
  eyJ1bmlxdWVfbmFtZSI6ImFkbWluIiwibmJmIjoxNjM0NzMwMjIyLCJleHAiOjE2MzQ3MzIwMjIsImlhd
  CI6MTYzNDczMDIyMn0.
  YYAxVoEAk8_sfJxriv38_28s2TCxaTZ9A0enughbPDSbXAKo-8FA5IhLg-54zCNWIKp9Pjk_FDCexpMZ0
  tWuRw"
```

- Request to resources with the help of generated JWT token.

- We add the JWT token to the request with the help of the authentication tool.

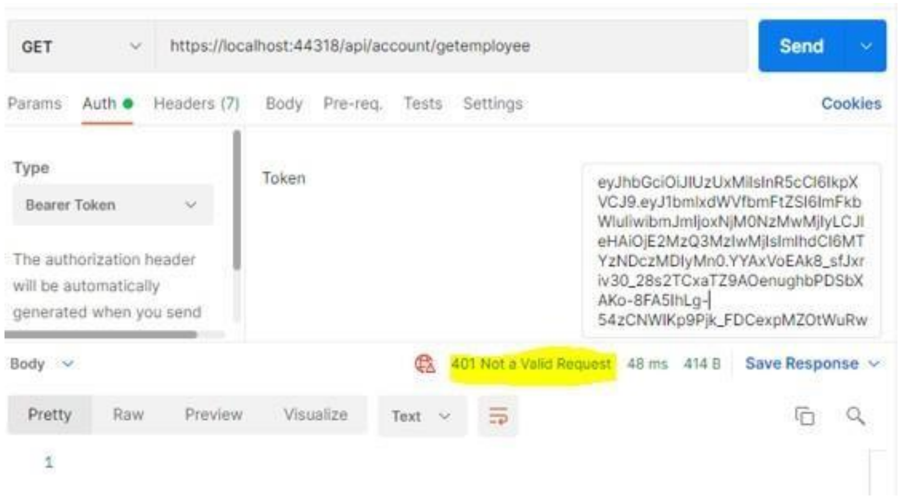


GET https://localhost:44318/api/account/getemployee

Type: Inherit au...

The authorization hea automatically generat the request.

- Request to resource with JWT token in the request and it will return 401 (Unauthorized).



GET https://localhost:44318/api/account/getemployee

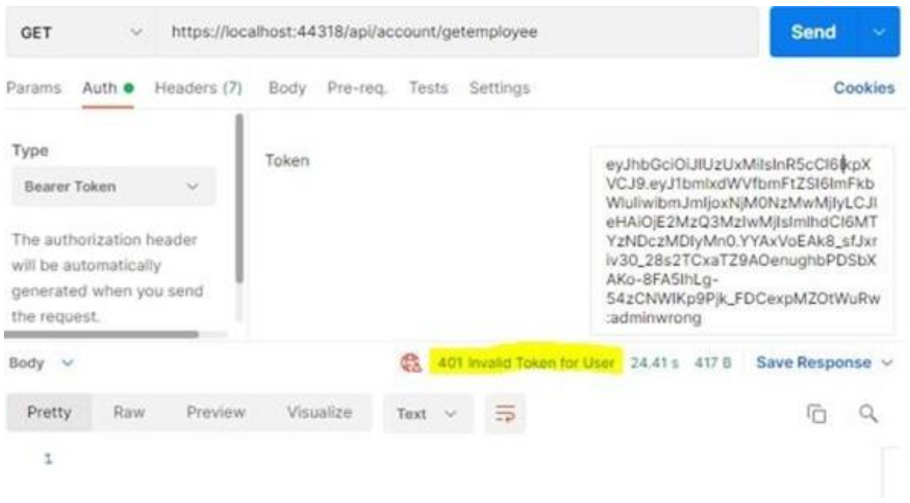
Type: Bearer Token

The authorization header will be automatically generated when you send

Token: eyJhbGciOiJIUzUxMiIsInR5cCI6IkpXVCJ9.eyJ1bmlxdWVfbmFtZSI6ImFkbWluIiwibmJmIjoxNjM0NzMwMjIyLCJleHAiOjE2MzQ3MzIwMjIsImlhdCI6MTYzNDczMDIyMn0.YYAxVoEAk8_sfJxriv30_28s2TCxaTZ9AOenughbPDSbXAKo-8FA5IhLg-54zCNWIKp9Pjk_FDCexpMZOtWuRw

401 Not a Valid Request 48 ms 414 B

- Request to resources with the help of JWT token and invalid user name, this will result in 401 (Unauthorized)



https://localhost:44318/api/account/getemployee

The authorization header will be automatically generated when you send the request.

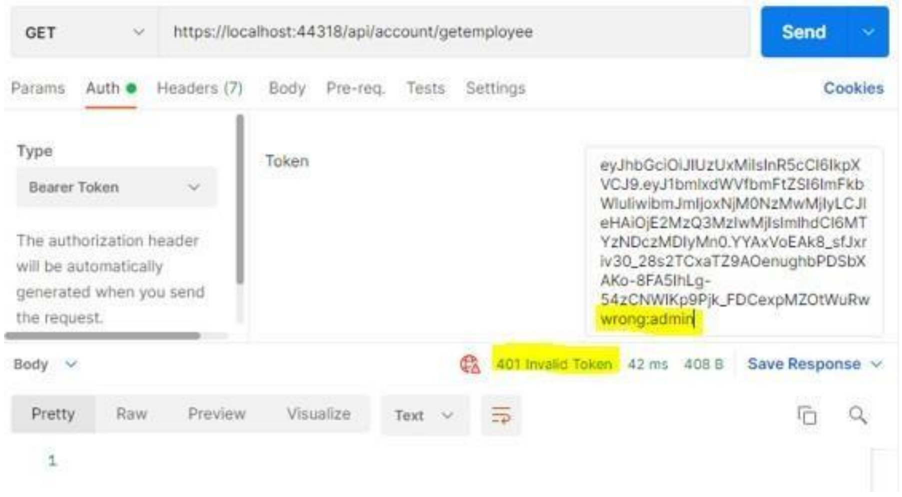
Token

```
eyJhbGciOiJIUzUxMiIsInR5cCI6IkpX
VCJ9.eyJ1bmlxdWVfbmFtZSI6ImFkb
WluIiwibmJmIjoxNjM0NzMwMjIyLCJl
eHAiOjE2MzQ3MzIwMjIsImlhdCI6MT
YzNDczMDIyMn0.YYAxVoEAk8_sfJxr
iv30_28s2TCxaTZ9AOenughbPDSbX
AKo-8FA5IhLg-
54zCNWIKp9Pjk_FDCexpMZOtWuRw
:adminwrong
```

401 Invalid Token for User 24.41 s 417 B

```
1
```

- Request resources with wrong JWT and correct unique this will result in name, 401 (Unauthorized)



https://localhost:44318/api/account/getemployee

The authorization header will be automatically generated when you send the request.

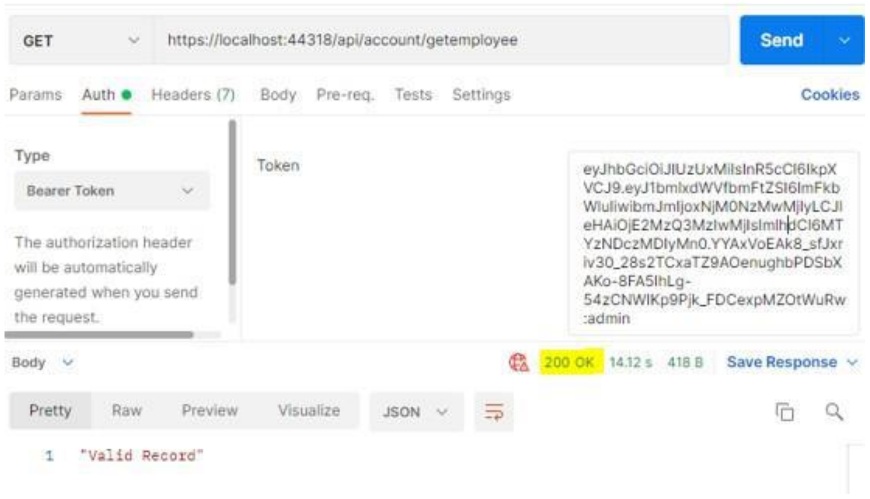
Token

```
eyJhbGciOiJIUzUxMiIsInR5cCI6IkpX
VCJ9.eyJ1bmlxdWVfbmFtZSI6ImFkb
WluIiwibmJmIjoxNjM0NzMwMjIyLCJl
eHAiOjE2MzQ3MzIwMjIsImlhdCI6MT
YzNDczMDIyMn0.YYAxVoEAk8_sfJxr
iv30_28s2TCxaTZ9AOenughbPDSbX
AKo-8FA5IhLg-
54zCNWIKp9Pjk_FDCexpMZOtWuRw
wrong:admin
```

401 Invalid Token 42 ms 408 B

```
1
```

- Request to resource with correct JWT and correct unique username. This will result in 200 (Ok, Authorized) and you will get the requested resources.



Type: Bearer Token

The authorization header will be automatically generated when you send the request.

Token

```
eyJhbGciOiJIUzUxMiIsInR5cCI6IkpX
VCJ9.eyJ1bmlxdWVfbmFtZSI6ImFkb
WluIiwibmJmIjoxNjM0NzMwMjIyLCJl
eHAiOjE2MzQ3MzIwMjIsImlhdCI6MT
YzNDczMDIyMn0.YYAxVoEAk8_sfJxr
iv30_28s2TCxaTZ9AOenughbPDSbX
AKo-8FA5IhLg-
54zCNWIKp9Pjk_FDCexpMZOtWuRw
:admin
```

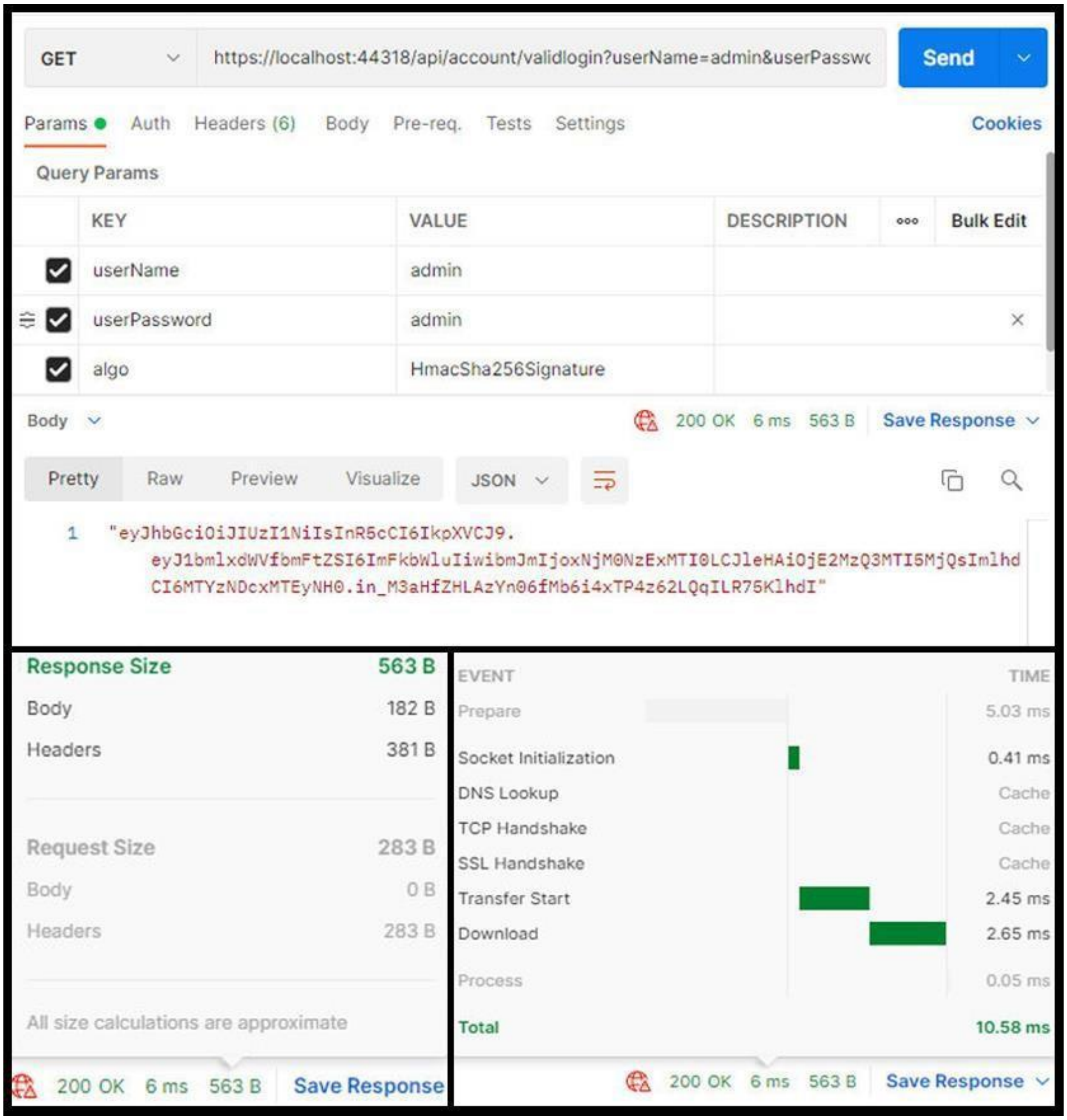
Body 200 OK 14.12 s 418 B

```
"Valid Record"
```

6. Results and Discussion

A mobile application is developed to find out the algorithm performance on HTJWT. We have tested the scenario with 50 times the process, starting from testing the time to generate the token, the size of the token generated, encrypting it 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.

If we compare the performance of all the algorithms used, we can see that the HMACSHA-256 algorithm is superior to an average response time of 8.6ms and size 563bytes when compared to HMACSHA-384 while with HMACSHA-512.



```
GET https://localhost:44318/api/account/validlogin?userName=admin&userPasswc
```

Query Params

	KEY	VALUE	DESCRIPTION
✓	userName	admin	
✓	userPassword	admin	
✓	algo	HmacSha256Signature	

Body — 200 OK 6 ms 563 B

```
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
    eyJ1bmlxdWVfbmFtZSI6ImFkbWluIiwibmJmIjoxNjM0NzExMTI0LCJleHAiOjE2MzQ3MTI5MjQsImlhd
    CI6MTYzNDcxMTEyNH0.in_M3aHfZHLAzYn06fMb6i4xTP4z62LQqILR75KlhdI"
```

Response Size	563 B
Body	182 B
Headers	381 B
Request Size	283 B
Body	0 B
Headers	283 B

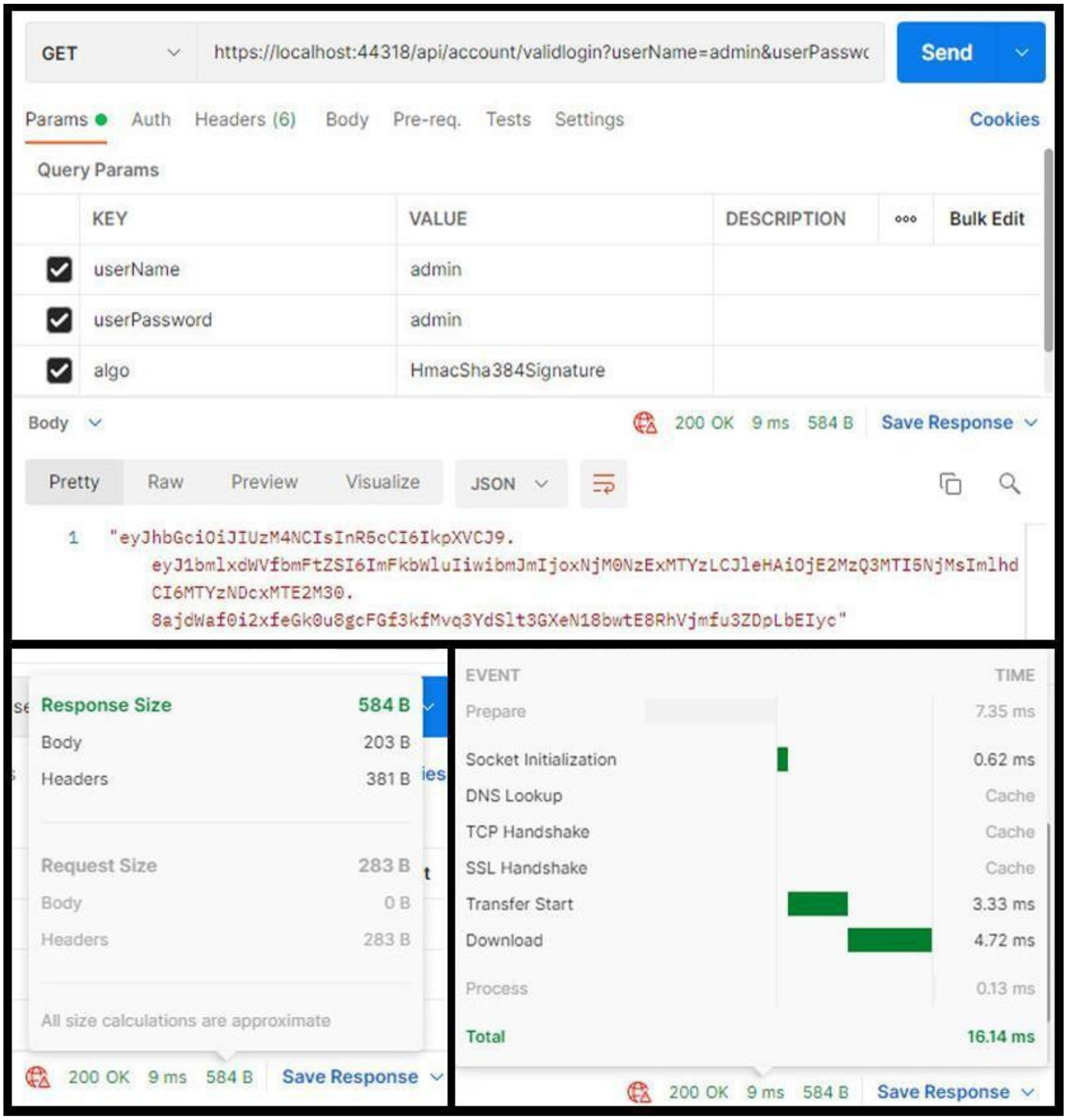
All size calculations are approximate

200 OK 6 ms 563 B

EVENT	TIME
Prepare	5.03 ms
Socket Initialization	0.41 ms
DNS Lookup	Cache
TCP Handshake	Cache
SSL Handshake	Cache
Transfer Start	2.45 ms
Download	2.65 ms
Process	0.05 ms
Total	10.58 ms

200 OK 6 ms 563 B

Figure 2: Results of HMACSHA-256


GET https://localhost:44318/api/account/validlogin?userName=admin&userPasswc

Query Params

KEY	VALUE	DESCRIPTION
✓ userName	admin	
✓ userPassword	admin	
✓ algo	HmacSha384Signature	

Body — 200 OK 9 ms 584 B

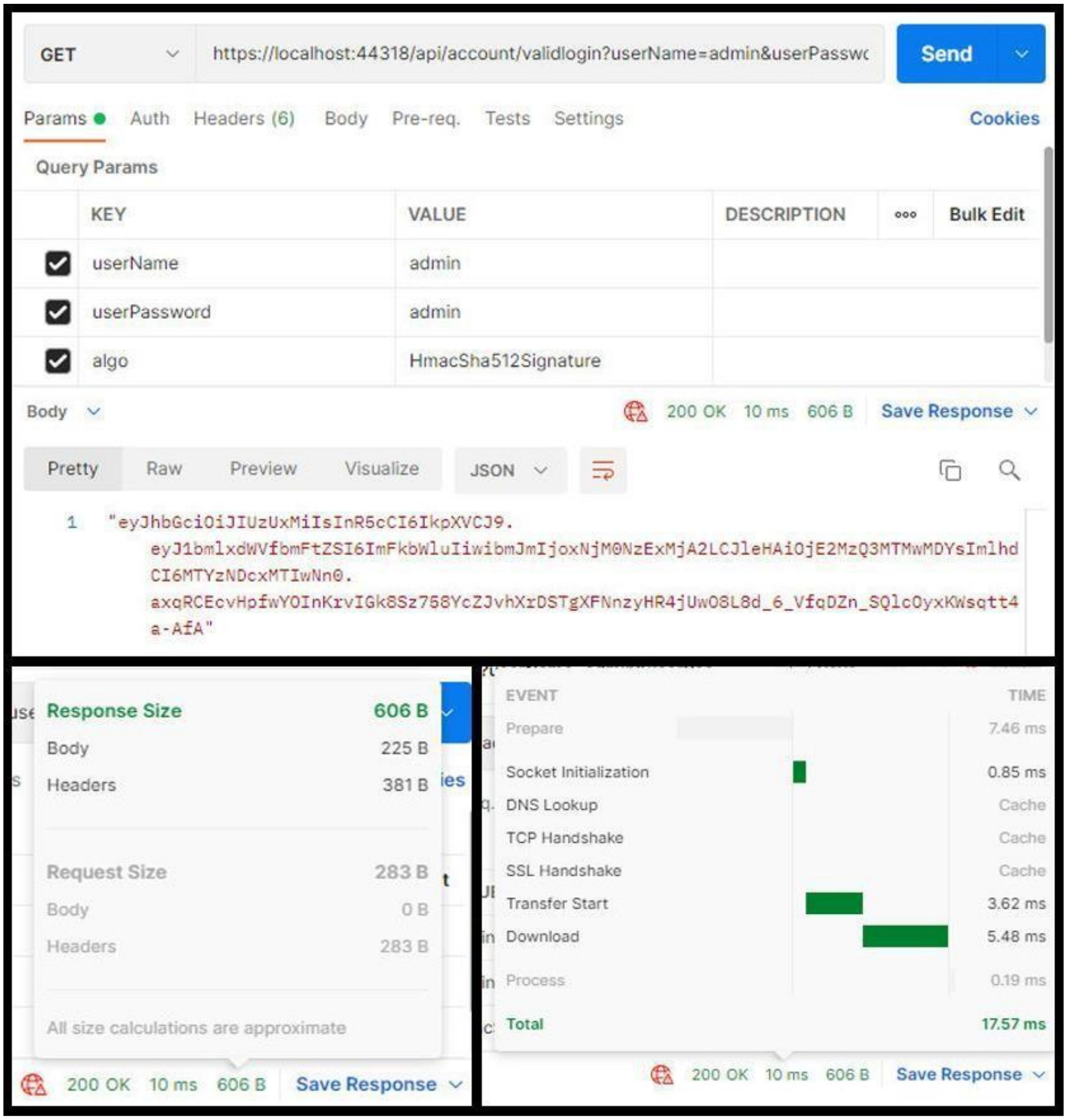
```
"eyJhbGciOiJIUzM4NCIsInR5cCI6IkpXVCJ9.
    eyJ1bmlxdWVfbmFtZSI6ImFkbWluIiwibmJmIjoxNjM0NzExMTYzLCJleHAiOjE2MzQ3MTI5NjMsImlhd
    CI6MTYzNDcxMTE2M30.
    8ajdWaf0i2xfeGk0u8gcFGf3kfMvq3YdSlt3GXeN18bwtE8RhVjmfu3ZDpLbEIyc"
```

Response Size	584 B
Body	203 B
Headers	381 B
Request Size	283 B
Body	0 B
Headers	283 B

All size calculations are approximate

EVENT	TIME
Prepare	7.35 ms
Socket Initialization	0.62 ms
DNS Lookup	Cache
TCP Handshake	Cache
SSL Handshake	Cache
Transfer Start	3.33 ms
Download	4.72 ms
Process	0.13 ms
Total	16.14 ms

Figure 3: Results of HMACSHA-384



```
GET https://localhost:44318/api/account/validlogin?userName=admin&userPasswc
```

Query Params

KEY	VALUE	DESCRIPTION
✓ userName	admin	
✓ userPassword	admin	
✓ algo	HmacSha512Signature	

Body 200 OK 10 ms 606 B

```
"eyJhbGciOiJIUzUxMiIsInR5cCI6IkpXVCJ9.
    eyJ1bmlxdWVfbmFtZSI6ImFkbWluIiwibmJmIjoxNjM0NzExMjA2LCJleHAiOjE2MzQ3MTMwMDYsImlhd
    CI6MTYzNDcxMTIwNn0.
    axqRCEcvHpfwYOInKrvIGk8Sz758YcZJvhXrDSTgXFNnzyHR4jUwO8L8d_6_VfqDZn_SQlcOyxKWsqtt4
    a-AfA"
```

Response Size	606 B
Body	225 B
Headers	381 B
Request Size	283 B
Body	0 B
Headers	283 B

All size calculations are approximate

200 OK 10 ms 606 B

EVENT	TIME
Prepare	7.46 ms
Socket Initialization	0.85 ms
DNS Lookup	Cache
TCP Handshake	Cache
SSL Handshake	Cache
Transfer Start	3.62 ms
Download	5.48 ms
Process	0.19 ms
Total	17.57 ms

200 OK 10 ms 606 B

Figure 4: Results of HMACSHA-512

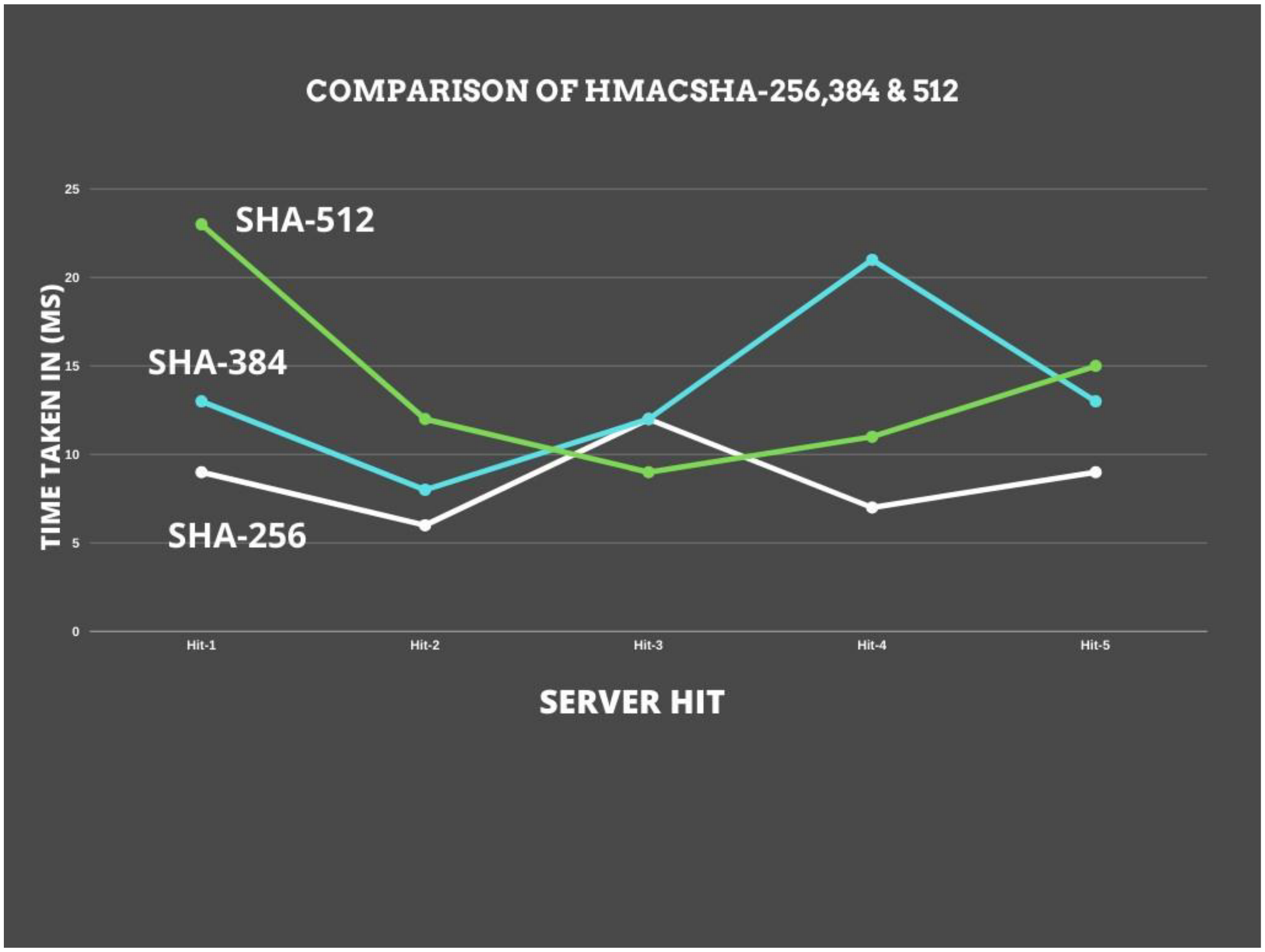


Figure 5: Graph of Comparison of HMACSHA-256, 384 & 512

7. Conclusion

In this study, we have presented an enhancement in the JWT access control solutions for different applications developed on the platforms, such as mobile apps and cloud systems. Creation of new tokens on every client request on server can resist attacker to identify the signature of a client. We have implemented a technique which can be used by application developers for securing the server resources from revoked permissions and unauthenticated requests.

The results are explained with the comparison of token- based authentication performance using JWT with several algorithms. The overall results show that the use of THJWT with HMACSHA-256 signature is superior in case of time taken to generate token, token size and token transfer speed.

8. Future Work

We can implement various user roles and authorize them to access specific resources according to the need of the overall application. Further we can test our system (mobile application) for various types of cross scripting attacks such as None Algorithm attack, signature stripping, manipulating Kid (Key Identifier), SQL injection, etc.

9. References

[1] Kumar P, Gurtov A, Iinatti J, Ylianttila M and Sain M, "Lightweight and Secure Session-Key Establishment Scheme in Smart Home Environments," in IEEE Sensors Journal, vol. 16, no.1, pp. 254-264, Jan.1, 2016. doi: 10.1109/JSEN.2015.2475298.

[2] Janardanan, Ajil Paul C, Anju P, Eldiva Thomas V and Davis D, "Android Application for Car Wash Services," 2018 International Conference on Emerging Trends and Innovations In Engineering And Technological Research (ICETIETR), Ernakulam, 2018, pp. 1-3. doi: 10.1109/ICETIETR.2018.8529025.

[3] Liu Z and Gupta B"Study of Secured Full-Stack Web Development," Proceedings of 34th International Conference on Computers and Their Applications, vol. 58, pp. 317- 324, 2019 doi: 10.29007/jpj6.

[4] Ethelbert O, Moghaddam F.F, Wieder P and Yahyapour R, "A JSON Token-Based Authentication and Access Management Schema for Cloud SaaS Applications," 2017 IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud), Prague, 2017, pp. 47-53. doi: 10.1109/FiCloud.2017.

[5] Hong N, Kim M, Jun M, Kang J, "A Study on a JWT-Based User Authentication and API Assessment Scheme Using IMEI in a Smart Home Environment," in jornal of sustainability, vol. 9, no. 7, June 2017.

[6] Jones M, Bradley J, and Sakimura N, "JSON Web Token (JWT) RFC 7519", http://www.rfc-editor.org/rfc/rfc7519.txt, RFC Editor 2015.

[7] Chifor B, Arseni S, Matei I and Bica I, "Security-Oriented Framework for Internet of Things Smart-Home Applications," 2019 22nd International Conference on Control Systems and Computer Science (CSCS), Bucharest, Romania, 2019, pp. 146-153. doi: 10.1109/CSCS.2019.00033.

[8] Gutzmann K, "Access control and session management in the HTTP environment," in IEEE Internet Computing, vol. 5, no. 1, pp. 26-35, Jan.-Feb. 2001. doi: 10.1109/4236.895139.

[9] Yuan X, Borkor E, Beal S, Yu. H "Retrieving relevant CAPEC attack patterns for secure software development," In Proceedings of the 9th Annual Cyber and Information Security Research Conference (CISR '14), ACM, pp. 33-36, 2014 doi: https://doi.org/10.1145/2602087.2602092.

[10] Viktor Jánoky L, Levendovszky J, Ekler P, "An analysis on the revoking mechanisms for JSON Web Tokens," International Journal of Distributed Sensor Networks, vol. 14, September 2018, doi: https://doi.org/10.1177/1550147718801535.

[11] Gomez C and Paradells J, "Wireless home automation networks: A survey of architectures and technologies," IEEE Commun. Mag., vol. 48, no. 6, pp. 92–101, Jun. 2010.

[12] Kim J E, Boulos G, Yackovich J, Barth T, Beckel C, and Mosse D, "Seamless integration of heterogeneous devices and access control in smart homes," in Proc. 8th Int. Conf. Intell. Environ. (IE), Jun. 2012, pp. 206–213.

[13] Mantas G, Lymberopoulos D, and Komninos N, "Security in smart home environment," in Wireless Technologies for Ambient Assisted Living and Healthcare: Systems and Applications. Hershey, PA, USA: IGI Global, 2006.

[14] Pishva D and Takeda K, "A product based security model for smart home appliances," in Proc. 40th Annu. IEEE Int. Carnahan Conf. Secur. Technol., Oct. 2006, pp. 234–242.

[15] Suryadevara N K, Mukhopadhyay S C, Wang R, and Rayudu R K, "Forecasting the behavior of an elderly using wireless sensors data in a smart home," Eng. Appl. Artif. Intell., vol. 26, no. 10, pp. 2641–2652, Nov. 2013.

[16] Bhardwaj S, Ozcelebi T, Lukkien J, and Uysal C, "Resource and service management architecture of a low capacity network for smart spaces," IEEE Trans. Consum. Electron.,vol. 58, no. 2, pp. 389–396, May 2012.

[17] Tschofenig H, Arkko J, and McPherson D, "Architectural considerations in smart object networking," Internet Engineering Task Force, Fremont, CA, USA, Tech. Rep. RFC-7452, Jul. 2014.

[18] Korzun D G, Balandin S I, and Gurtov A V, Deployment of Smart Spaces in Internet of Things: Overview of the Design Challenges (Lecture Notes in Computer Science), vol.8121. New York, NY, USA: Springer, 2013.
[19] Chen Y and Luo B, "S2A: Secure smart household appliances," in Proc. 2nd ACM Conf. Data Appl. Secur. Privacy (CODASPY), 2012, pp. 217–228.
[20] Li, "Design Y of a key establishment protocol for smart home energy management system," in Proc. 5th Int. Conf. Comput. Intell., Commun. Syst. Netw. (CICSyN), Jun. 2013, pp. 88–93.
[21] Vaidya B, Makrakis D, and Mouftah H T, "Device authentication mechanism for smart energy home area networks," in Proc. IEEE Int. Conf. Consum. Electron. (ICCE), Jan. 2011, pp. 7