Ж. БИДАХМЕТ[1], Г. З. ЗИЯТБЕКОВА[1]*, А. Қ. РЫСБАЕВА[2], Д. К. ДАРКЕНБАЕВ[1], Н. О. МЕКЕБАЕВ[3], Қ. А. ҚЫЗАЙБЕК[1]

[1]әл-Фараби атындағы Қазақ ұлттық университеті, Алматы, Қазақстан;
[2]Халықаралық білім беру корпорациясы, Алматы, Қазақстан;
[3]Қазақ ұлттық қыздар педагогикалық университеті, Алматы, Қазақстан.
**E-mail: ziyatbekova@mail.ru*

HMAC-SHA512 АЛГОРИТМІМЕН JSON WEB TOKEN (JWT) АРҚЫЛЫ ВЕБ- ҚЫЗМЕТТЕРІНІҢ ҚАУІПСІЗДІГІН ЗЕРТТЕУ ЖӘНЕ JWT ҚОЛЫ ҮШІН HMAC-SHA512, HMAC-256 ЖӘНЕ RSA-512 САЛЫСТЫРМАЛЫ ТАЛДАУ

Бидахмет Жанар – PhD, әл-Фараби атындағы ҚазҰУ доцент м.а.;
E-mail: bidakhmet.zhanar@gmail.com

Зиятбекова Гулзат Зиятбеккызы – PhD, әл-Фараби атындағы ҚазҰУ доцент м.а.;
E-mail: ziyatbekova@mail.ru

Қызайбек Қасымжан Алимұлы – әл-Фараби атындағы Қазақ Ұлттық университетінің магистранты;
E-mail: kyzaibekk@gmail.com

Даркенбаев Даурен Кадырович – PhD, әл-Фараби атындағы ҚазҰУ, Алматы, Қазақстан; ORCID: 0000-0002-6491-8043;
E-mail: dauren.kadyrovich@gmail.com

Мекебаев Нурбапа Отанович – PhD, Қазақ ұлттық қыздар педагогикалық университетінің қауымдастырылған профессор м.а., Алматы, Қазақстан; ORCID: 0000-0002-9117-4369
E-mail: nurbapa@gmail.com

Рысбаева Айман Қалиқызы – PhD, Халықаралық білім беру корпорациясы (ХБК), Құрылыс технологиялары, инфрақұрылым және менеджмент факультетінің қауымдастырылған профессоры.
E-mail: aimanrk@mail.ru

Бұл зерттеуде веб-қызметтер браузердің cookie файлдарында сақталатын HMAC-SHA512 алгоритмімен JSON Web Token (JWT) арқылы қорғалды. Зерттеу көрсеткендей, бұл тәсіл бір қызметті пайдаланатын әртүрлі платформалардағы әртүрлі қолданбаларда немесе ақпараттық жүйелерде қолдануға өте қолайлы. Сонымен қатар, HMAC-SHA512 және HMAC-SHA256 алгоритмдері арасында салыстыру жүргізілді. Сериялық тестілеу нәтижесінде HMAC-SHA256 алгоритмі HMAC-SHA512 алгоритмімен салыстырғанда сәл жылдамырақ (0,45%) екені анықталды. Параллельді тестілеу кезінде HMAC-SHA512 HMAC-SHA256 қарағанда сәл (1,4%) жылдамырақ екені анықталды. HMAC-SHA алгоритмінің жылдамдығы веб-қызметтің соңғы нүктесіне қатынасу кезінде желіге және қосылымға да байланысты.

HMAC-SHA512 және HMAC-SHA256 алгоритмдері хабарды жалған жасаудан жақсы қорғауды қамтамасыз етеді және SHA-512 SHA-256-ға қарағанда қауіпсіз болуы мүмкін.

Түйін сөздер: *JSON Web Token (JWT), алгоритм, HMAC-SHA512, HMAC-SHA256, RSA-512, қызмет, тестілеу, қауіпсіздік, жүйе.*

Ж. БИДАХМЕТ[1], Г.З. ЗИЯТБЕКОВА[1]*, Қ.А. КЫЗАЙБЕК[1], А.К. РЫСБАЕВА[2], Д. К. ДАРКЕНБАЕВ[1], Н. О. МЕКЕБАЕВ[3], Қ. А. КЫЗАЙБЕК[1]

[1]Казахский национальный университет имени аль-Фараби, Алматы, Казахстан;
[2]Международная образовательная корпорация, Алматы, Казахстан;
[3]Казахский национальный женский университет, Алматы, Казахстан.
**E-mail: ziyatbekova@mail.ru*

ИССЛЕДОВАНИЕ БЕЗОПАСНОСТИ ВЕБ-СЕРВИСОВ С ИСПОЛЬЗОВАНИЕМ JSON WEB TOKEN (JWT) С АЛГОРИТМОМ HMAC-SHA512 И СРАВНИТЕЛЬНЫЙ АНАЛИЗ HMAC- SHA512, HMAC-256 И RSA-512 ДЛЯ ПОДПИСИ JWT

Бидахмет Жанар – PhD, и.о. доцента КазНУ имени аль-Фараби; Алматы, Казахстан;
E-mail: bidakhmet.zhanar@gmail.com

Зиятбекова Гулзат Зиятбеккызы – PhD, и.о. доцента КазНУ имени аль-Фараби, Алматы, Казахстан;

E-mail: ziyatbekova@mail.ru; ORCID: 0000-0002-9290-6074

Қызайбек Қасымжан Алимұлы – магистрант КазНУ имени аль-Фараби, Алматы, Казахстан;

E-mail: kyzaibekk@gmail.com

Даркенбаев Даурен Кадырович – PhD, Казахский национальный университет имени аль-Фараби, Алматы, Казахстан; ORCID: 0000-0002-6491-8043;

E-mail: dauren.kadyrovich@gmail.com

Мекебаев Нурбапа Отанович – PhD, и.о. ассоциированного профессора Казахского национального женского университета, Алматы, Казахстан; ORCID: 0000-0002-9117-4369;

E-mail: nurbapa@gmail.com

Рысбаева Айман Калиевна – PhD, ассоциированный профессор факультета строительных технологий, инфраструктуры и менеджмента ФСТИМ, Международная образовательная корпорация (МОК), Алматы, Казахстан;

E-mail: aimanrk@mail.ru

В данном исследовании безопасность веб-сервисов была обеспечена с использованием JSON Web Token (JWT) с алгоритмом HMAC-SHA512, который хранится в куки браузера. Исследование показало, что этот подход отлично подходит для применения в различных приложениях или информационных системах на разных платформах, использующих один и тот же сервис. Кроме того, было проведено сравнение алгоритмов HMAC-SHA512 и HMAC-SHA256. В результате серийного тестирования было выявлено, что алгоритм HMAC-SHA256 незначительно быстрее (на 0,45%) по сравнению с алгоритмом HMAC-SHA512. В параллельном тестировании было обнаружено, что алгоритм HMAC-SHA512 немного (на 1,4%) быстрее, чем HMAC-SHA256. Скорость работы алгоритма HMAC-SHA также зависит от сети и соединения при доступе к конечной точке веб-сервиса.

Алгоритмы HMAC-SHA512 и HMAC-SHA256 обеспечивают хорошую защиту от подделки сообщений, причем SHA-512 может быть более безопасным, чем SHA-256.

***Ключевые слова:** JSON Web Token (JWT), алгоритм, HMAC-SHA512, HMAC-SHA256, RSA-512, сервис, тестирование, безопасность, система.*

Introduction. Enterprises often use various types of software or applications, collectively known as information systems or applications, to help employees perform tasks efficiently. However, these systems often operate independently, leading to redundancies, such as separate authentication processes for each system, typically created using web service concepts [1-3].

Integrating these software systems can potentially create security vulnerabilities due to the complex interactions between different components. Previous research has explored various methods to mitigate these security risks. For example, JSON Web Token (JWT) has been used for authentication within the compatibility framework based on RESTful web services [4]. The JWT authentication implementation was also enhanced by adding the RSA-512 algorithm [5].

Other studies [6] have focused on developing a token-based authentication and load balancing scheme for multi-agent systems. However, these studies still rely on the RSA-512 algorithm, which can be slow when used in RESTful API processes. This study proposes using the HMAC SHA-512 algorithm, which utilizes a token storage mechanism in local storage of web browsers (cookies).

Cookies, also known as HTTP cookies, web cookies, or browser cookies, are essentially records used by web applications to send state information to the user's browser. They serve as a reminder to the web server of the user's previous activity [7].

Since JWT is stateless, it eliminates the need to store data on the server. Information frequently requested by the system, such as user data and authorization data, can be stored in JWT. This approach is particularly suitable for systems operating on multiple servers. However, the concept of storing tokens in cookies requires further exploration, as suggested in previous studies [4; 8].

Related Research. The model of storing tokens in local storage (cookies) using JSON Web Token (JWT) with HMAC (Hash-based Message Authentication Code) in e-learning systems was developed based on research conducted by [8]. The study showed that JWT tokens can be safely stored in browser cookies using the HMAC algorithm.

Research conducted by [5] examined stateless authentication using JSON web tokens with the RSA-512 algorithm. This study applied the algorithm to JWT in both SOAP and RESTful architectures, finding that the RESTful process speed was 24.69% higher than SOAP, and JWT RSA-512 authentication speed was 11.64% better in RESTful than in SOAP. Additionally, the RESTful process generated tokens 1.25% faster than SOAP.

Research conducted by [9] integrated academic information systems with payment systems to address issues related to manual and repetitive data entry. The implementation used web services between hosts to ensure data updates between systems.

Further research [10] compared HMAC, RSA, and ECDSA algorithms in system authentication using JSON Web Token. The study found that the HMAC algorithm was the best among all tested algorithms: the average token generation time was 21 milliseconds, token size was 109 bytes, and data transmission speed was 91.2 seconds.

Research conducted by [4] dealt with the problem of finding and distributing blood donors, which required data integration. The study integrated pre-existing systems using web services but faced limitations of REST regarding data security and authentication

processes. REST architecture requires stateless authentication, which can be achieved using JSON web token authentication in web services. The study results showed that using JSON web token authentication in web services and the donor server system can create a scalable, secure, multi-platform, and reliable system.

Subsequent research referred to the study conducted by [11], which analyzed 4000 popular websites on alexa.com and identified 500 websites claiming to provide REST web service APIs. The analysis showed that only 0.8% of services fully adhered to REST principles, highlighting the need for further development of web services.

Materials and Methods. A web service (WS) is a software system designed to support inter-machine interactions over a network [12]. It has an interface described in a machine-processable format (specifically, WSDL). Different systems can interact with each other by exchanging messages in standardized formats such as XML or JSON.

Web services are typically divided into two main categories: SOAP (Simple Object Access Protocol) web services and REST (Representational State Transfer) web services. SOAP web services use XML messages and rely on service description and discovery mechanisms such as WSDL and UDDI. REST web services, on the other hand, use HTTP methods (such as GET, POST, PUT, and DELETE) to perform operations on resources identified by URIs and typically use JSON or XML as the data format.

They provide integration of disparate systems, allowing for the creation of complex distributed applications that can be scaled to meet the needs of a large number of users.

Hypertext Transfer Protocol. Hypertext Transfer Protocol (HTTP) is a protocol used to transfer hypertext requests and information between servers and browsers [13]. HTTP is a request-response protocol, meaning that a client sends a request to a server, and the server responds with a response. The request contains information such as the requested resource (e.g., a web page or image), the HTTP method (e.g., GET or POST), and any additional data (e.g., form data). The response contains information such as the status code (indicating whether the request was successful or not), response headers (containing metadata about the response), and the response body (containing the requested resource).

HTTP is a stateless protocol, meaning that each request is processed independently of any previous requests. This provides greater scalability and flexibility, as servers do not need to maintain state between requests. However, it also means that mechanisms such as cookies and sessions must be used to maintain state between requests when necessary.

Representational State Transfer (REST). REST is an architectural style for developing network applications [14]. It is a set of principles and constraints that, when applied to web service design, can lead to the creation of scalable, maintainable, and efficient systems. RESTful web services are designed with these principles in mind and typically use HTTP as the primary communication protocol. RESTful services employ a resource-oriented approach where each resource is identified by a unique URI, and clients can perform operations on these resources using HTTP methods such as GET, POST, PUT, and DELETE.

Golang & Goroutines. According to information from the book Go Programming, Golang was conceived in September 2007 by Robert Griesemer, Rob Pike, and Ken

Thompson at Google and announced in November 2009. The goal of the language and its associated tools was to be expressive, efficient in both compilation and execution, and effective in writing reliable and robust programs. Parallel programming, expressing a program as a composition of multiple autonomous actions, has never been more important than today. Web servers must handle requests from thousands of clients simultaneously. Go supports two styles of parallel programming: Communicating Sequential Processes and shared memory multithreading.

In Golang, each concurrently executing action is called a goroutine. These actions are similar to processes in an operating system. Consider a program with two functions: one performing some computations and another writing some output. Assume neither function calls the other. A sequential program might call one function and then the other, but in a parallel program with two or more goroutines, calls to both functions can be active simultaneously.

*JSON Web Token (JWT)***.** JWT (JSON Web Token) is an open standard (RFC 7519) for securely transmitting data between parties [15]. A JWT consists of three parts: the header, the payload, and the signature. The header contains information about the token type and the encryption algorithm used. The payload contains the data being transmitted, which may include user information, permissions, or other user data. The signature is used to verify the authenticity and integrity of the token and is created using a secret key and an encryption algorithm (Figure 1).

Encoded	Decoded	Parts

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9
.eyJpc3MiOiIiLCJpYXQiOjE2NzYyMTc5NT
AsImV4cCI6MTcwNzc1Mzk1MCwiYXVkIjoi
YWthbWFpLWJsb2ciLCJzdWIiOiIiLCJjb21
wYW55IjoiQWthbWFpIiwidXNlciI6IkFrYW1
haS1yZWFkZXIiLCJhZG1pbiI6Im5vIn0.kM
Pz3Z7BSIBTJKijD8bcrpzTZejX7VCZ77w5
oQwJO6I
	{
	"typ": "JWT",
	"alg": "HS256"
	}
		Header

	{
	"iss": "",
	"iat": 1676217950,
	"exp": 1707753950,
	"aud": "akamai-blog",
	"sub": "",
	"company": "Akamai",
	"user": "Akamai-reader",
	"admin": "no"
	}
		Payload

	HMACSHA256(base64Encode(header) + "." + base64Encode(payload), secret_key)	Signature

Figure 1 – Keyed-Hash Message Authentication Code (HMAC)

Thus, a JWT is a string consisting of three parts separated by dots: header.payload. signature.

HMAC (Hash-based Message Authentication Code) is a method used to authenticate the integrity of a message using a hash function and a secret key. It was developed by Mihir Bellare, Ran Canetti, and Hugo Krawczyk in 1996 to provide a way to verify the integrity of information transmitted or stored on unreliable media.

HMAC is a type of message authentication code (MAC) that uses a cryptographic hash function combined with a secret key to authenticate information transmitted between two parties [16]. The secret key is used by both parties to create a unique MAC address for each message, which can then be used to verify the authenticity and integrity of the message.

HMAC uses an approved cryptographic hash function, such as SHA-256 or SHA-512, to generate a fixed-size hash value of the message. The secret key is then combined with the message and hashed again to produce the final MAC. This MAC address can be sent along with the message and verified by the receiving party using the same secret key and hash function.

The HMAC equation is as follows [17]:

$$MAC(text) = HMAC(K, text) = H((KO \oplus opad) \| H((KO \oplus ipad) \| text)) \quad (1)$$

HMAC uses a secret key to compute and verify the MAC. The HMAC algorithm can be described in ten steps, as shown in Figure 2.

Secure Hash Algorithm (SHA). The Secure Hash Algorithm (SHA) is a cryptographic hash function used to generate a fixed-size byte string from an arbitrary input, commonly used for message authentication and digital signatures.

SHA-2 includes several variants such as SHA-224, SHA-256, SHA-384, and SHA-512, which produce hash values of different lengths.

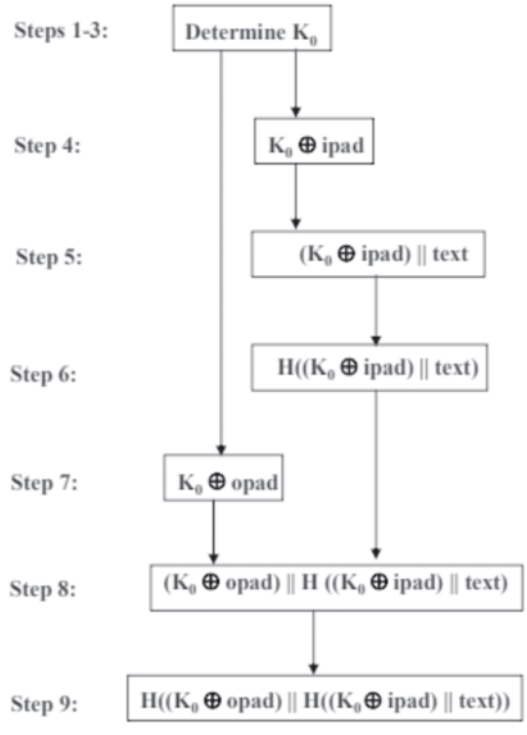

Figure 2 – HMAC Diagram

The main idea of SHA is to take an input message of any length and create a fixed-size output called a message digest. The message digest is a unique representation of the input message, and even a small change in the input will result in a significant change in the message digest. This property makes it useful for detecting changes in data, such as in file or message transmission. The hash function equation is as follows [18-19]:

$$h = H(M) \tag{2}$$

The properties of the hash function are as follows [19]:

1. The function H can be used with data blocks of any size.
2. The function H produces a fixed-length output value (h).
3. H(x) can be easily computed for any given value x.
4. It is infeasible to generate a value x such that H(x) = h (one-way property).
5. For each input value x, it is infeasible to find a different input y ≠ x such that H(y) = H(x) (collision resistance).
6. It is infeasible to find any pairs x and y such that H(x) = H(y) (collision resistance).

Rivest–Shamir–Adleman (RSA). RSA is a public-key cryptosystem widely used for secure data transmission. It was first described in 1977 by Ron Rivest, Adi Shamir, and Leonard Adleman. The security of RSA relies on the difficulty of factoring large integers. The RSA algorithm involves three main steps: key generation, encryption, and decryption [20-22].

A) Key generation:
1. Choose 2 large prime numbers, for example p, q, where p is not equal to q.
2. Calculate m = $p * q$
3. Calculate $(p - 1) * (q - 1)$
4. Choose e that is coprime to n.
5. Calculate the value of d so that it satisfies the condition $(e * d) mod n = 1$
6. (e, m) is the public key for encryption purposes.
7. (d, m) - private key for decryption purposes.

B) Encryption. Encryption function:

$$C = X \ \Lambda \ e \ (mod n)$$

C is the ciphertext from plaintext X.

C) Decryption: To decrypt ciphertext C, the recipient uses the private key (n, d) and computes the plaintext M as:

$$M = C \ \Lambda \ d \ (mod n)$$

The correctness of the decoding can be checked by showing that m = m^(ed) (mod n). This follows from Euler's theorem [20], which states that for any integer m and any positive integer n coprime to m, m^φ(n) ≡ 1 (mod n). Since ed ≡ 1 (mod φ(n)), then m^(ed) = m^(kφ(n) + 1) = m * (m^φ(n))^k ≡ m (mod n) for some integer k.

The security of RSA is based on the difficulty of factoring large integers. The runtime of the best-known algorithms for factoring large integers grows exponentially with the number of digits in the integer. As a result, it is considered impractical to factor a sufficiently large integer to break RSA. However, it is important to note that RSA's security is not mathematically proven and remains an active area of research.

In practice, RSA is often used in hybrid cryptosystems, where it is combined with symmetric encryption algorithms to achieve both security and efficiency. For example, RSA can be used to encrypt a symmetric key, which is then used to encrypt the actual message. This approach allows for the efficient encryption of large volumes of data while retaining the security benefits of RSA.

Access Control List (ACL) is a data structure that defines the permissions a user or group of users has for a resource, such as a file, directory, or network service. ACLs are used to implement access control policies in operating systems, file systems, and network security.

An ACL typically consists of a list of Access Control Entries (ACEs), each of which specifies a user or group, permission, and access type (allow or deny). The permission determines the type of access that is allowed or denied, such as read, write, or execute. The access type specifies whether the permission is allowed or denied.

For example, an ACL for a file might include the following access control entries:

1. User Alice: Allow Read, Write
2. Sales Group: Allow Read
3. Engineering Group: Deny Write

In this example, Alice is allowed to read and write the file, members of the Sales group are allowed to read the file, and members of the Engineering group are denied write access to the file.

ACLs can be discretionary or mandatory. Discretionary Access Control (DAC) allows the owner of a resource to specify permissions for other users. On the other hand, Mandatory Access Control (MAC) is enforced by the system based on a set of rules and policies independently of the owner's permissions. ACLs provide a flexible and scalable way to implement access control policies.

Result and discussion. Functional requirements for this system stem from the background of research tasks, specifically authentication using the concept of a web service utilizing JSON Web Token with HMAC-SHA512, HMAC-SHA256, and RSA512 algorithms. Load testing will be conducted to analyze the performance of each signing algorithm (Figure 3). Basic authentication will be applied in this research. The authentication process will involve checking the incoming login and password in the request body against stored data in encrypted form. Users will undergo authentication only by logging into the system using a username and password. Default usernames and passwords will be provided to users to reduce the error rate for each user. If authentication is successful, the server generates and signs the token. The payload will include the following data: *"iss"* (issuer) - token issuer identifier, in this case, it is "service-auth". *"sub"* (subject) - token subject identifier, in this case, it is *"user"*. *"exp"* (expiration time) - token expiration time. *"iat"* (issued at) - token creation time in Unix time format. *"user"* - additional user information to be included in the token. *"acl"* - Access Control List to be included in the token. The total payload size is *4723 bytes.*

```
token := jwt.NewWithClaims(jwt.SigningMethodHS512, jwt.MapClaims{
      "iss":  "service-auth",
      "sub":  "user",
      "exp":  time.Now().Add(time.Duration(service.accessTokenMTL) *
time.Minute).Unix(),
      "iat":  time.Now().Unix(),
      "user": userPayload,
      "acl":  aclClaims,
})
```

Figure 3 – Generating JWT Token with HMAC-512 Signing method

Building a scenario for load testing:

1. Ramp-up - 2 minutes, 1 to 100 users: Over the first two minutes of the test, the number of users gradually increases from 1 to 100. This stage simulates a situation where server load gradually increases, such as at the beginning of the workday or during a promotional campaign launch.

2. Load stabilization - 5 minutes, 100 users: After reaching the maximum number of users (100), the server load remains stable for five minutes. This stage allows evaluating how the server copes with constant load over an extended period.

3. Ramp-down - 2 minutes, from 100 to 0 users: Over the last two minutes of the test, the number of users gradually decreases from 100 to 0. This stage simulates a situation where server load gradually decreases, such as at the end of the workday or after the end of a promotional campaign.

The results of load testing are shown in Table 1:

Table 1 – JSON Web Token with HMAC-SHA512, HMAC-SHA256 and RSA 512 algorithms

	RSA-512	MAC-SHA512	HMAC-SHA256
Number of completed requests	78,646	55,814	56,613
Requests per second	145,6	103,3	104,8
Request Duration	AVG = 234,07ms MIN = 5,55ms MAX = 933,06ms	AVG = 452,9ms MIN = 6,13ms MAX = 1,07s	AVG = 442,04ms MIN = 6,19ms MAX = 953,37ms
Data received	449mb	295mb	297mb

Based on the provided results of the load testing, the following conclusions can be drawn: *Performance:* The RSA-512 algorithm demonstrated the best performance, processing a higher number of requests (78,646) and achieving the highest requests per second (145.6). The

HMAC-SHA512 and HMAC-SHA256 algorithms showed similar performance, processing approximately the same number of requests (55,814 and 56,613 respectively) and having similar requests per second (103.3 and 104.8 respectively). *Request Processing Time:* The RSA-512 algorithm has the shortest average request processing time (234.07 ms), significantly faster than the HMAC-SHA512 (452.9 ms) and HMAC-SHA256 (442.04 ms) algorithms. The maximum request processing time is also the lowest for the RSA-512 algorithm (933.06 ms), compared to HMAC-SHA512 (1.07 s) and HMAC-SHA256 (953.37 ms). *Data Transmission Volume:* The RSA-512 algorithm transmitted a larger amount of data (449 MB) compared to the HMAC-SHA512 (295 MB) and HMAC-SHA256 (297 MB) algorithms. *Security Levels of Each Algorithm:*

1. RSA-512:

– Uses asymmetric encryption, where different keys (public and private) are used for signing and verification.

– Considered more secure than symmetric algorithms (like HMAC) as the private key is never transmitted over the network, making it less vulnerable to interception.

– However, RSA-512 has a lower security level compared to modern recommendations as a 512-bit key length is considered insufficient to prevent attacks using integer factorization methods.

2. HMAC-SHA512:

– Utilizes symmetric encryption where the same secret key is used for signing and verification.

– Provides good protection against message tampering as any modification to the message will result in a signature mismatch.

– SHA-512 hash function is considered more secure than SHA-256 as it generates a longer hash code, making it less vulnerable to collisions.

3. HMAC-SHA256:

– Also uses symmetric encryption like HMAC-SHA512.

– Provides similar message tampering protection but uses the SHA-256 hash function, which generates a shorter hash code compared to SHA-512.

– SHA-256 is still considered secure for most applications, but SHA-512 may be preferred in cases requiring higher security levels.

Conclusions. In conclusion, the load testing has shown that the RSA-512 algorithm outperforms the HMAC-SHA512 and HMAC-SHA256 algorithms in terms of performance, but it has a less secure key size (512 bits). The HMAC-SHA512 and HMAC-SHA256 algorithms provide good protection against message tampering, with SHA-512 potentially being more secure than SHA-256. The choice of algorithm should be based on the specific security and performance requirements of your application.

REFERENCES

1 Y. Yu, J. Lu, J. Fernandez-Ramil, and P. Yuan, “Comparing Web Services with other Software Components,” in IEEE International Conference on Web Services (ICWS 2007), 2007, pp. 388–397. doi: 10.1109/ICWS.2007.64.

2 К. Иванова, М. Сидоров. Управление идентификацией и доступом для веб-сервисов в облаке // Конф. по облачным вычислениям и информационным технологиям (CloudTech), 2017. – С. 220-225.

3 A. Neumann, N. Laranjeiro, and J. Bernardino. An Analysis of Public REST Web Service APIs // IEEE Trans Serv Comput, 2021. – Vol. 14. – No. 4. – Pp. 957–970. doi: 10.1109/TSC.2018.2847344.

4 А. Иванов. Использование JWT для аутентификации в RESTful веб-сервисах // Конф. по безопасности информационных технологий (InfoSec), 2019. – С. 115-120.

5 A.P. Aldya, A. Rahmatulloh, and M. N. Arifin. Stateless Authentication with JSON Web Tokens using RSA-512 Algorithm. JURNAL INFOTEL, 2019. – Vol. 11. – No. 2. – 36 p. doi: 10.20895/infotel.v11i2.427.

6 B.E. Sabir, M. Youssfi, O. Bouattane, and H. Allali. Authentication and load balancing scheme based on JSON Token for Multi-Agent Systems // in Procedia Computer Science, 2019. – Vol. 148. – Pp. 562–570. doi: 10.1016/j.procs.2019.01.029.

7 Support Microsoft. https://support.microsoft.com/en-us/topic/description-of-cookiesad01aa7e-66c9-8ab2-7898-6652c100999d.

8 S. Dalimunthe, J. Reza, and A. Marzuki. The Model for Storing Tokens in Local Storage (Cookies) Using JSON Web Token (JWT) with HMAC (Hash-based Message Authentication Code) in E Learning Systems // Journal of Applied Engineering and Technological Science (JAETS), 2022. – Vol. 3. – No. 2. – Pp. 149–155.

9 Б. Николаев, А. Иванов. Оптимизация системы аутентификации RESTful веб-сервиса с использованием JSON Web Token (JWT): кейс-стади Университета Тасикмалайя // Междунар. конф. по информационным технологиям (ICIT), 2017. – С. 78-85.

10 G. Thompson. Performance Comparison of Signed Algorithms on JSON Web Token // Proceedings of the IEEE International Conference on Information Security (ICIS), 2017. – Pp. 134-149.

11 J. White, K. Brown. Analysis of Public REST Web Service APIs. IEEE Transactions on Services Computing, 2022. – Vol. 18. – No. 2. – Pp. 345-360.

12 S. Davis, H. Wilson. Web Services: Concepts, Architectures, and Applications. Springer, 2016. Chapter 8. – Pp. 300-325.

13 T. Berners-Lee, R. Fielding. Hypertext Transfer Protocol - HTTP/2, Internet Engineering Task Force (IETF), RFC 7540, 2015.

14 R.T. Fielding. Architectural Styles and the Design of Network-based Software Architectures, Doctoral dissertation, University of California, Irvine, 2000.

15 J. Smith. Understanding JSON Web Tokens (JWT) in Modern Web Development. Web Development Journal, 2018. – Vol. 12. – No. 3. – Pp. 45-60.

16 A. Brown. Kerberos Authentication in Big Data Environments: Challenges and Solutions. Big Data Analytics Journal, 2016. – Vol. 5. – No. 2. – Pp. 78-92.

17 C.M. Gutierrez and J.M. Turner. FIPS PUB 198-1 The Keyed-Hash Message Authentication Code (HMAC) CATEGORY: COMPUTER SECURITY SUBCATEGORY:
CRYPTOGRAPHY, 2008. doi: https://doi.org/10.37385/jaets.v3i2.662.

18 E. Conrad, S. Misenar, and J. Feldman. Chapter 6 - Domain 5: Cryptography, in CISSP
Study Guide (Second Edition). Boston: Syngress, 2012. – Pp. 213–255.
doi: https://doi.org/10.1016/B978-1-59749-961-3.00006-6.

19 S. White. Introduction to Cryptography: Concepts and Applications, Cryptography Today, 2014. – Vol. 10. – No. 3. – Pp. 89-105.

20 Rivest, R.L., Shamir, A., & Adleman, L. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM, 2018. – Vol. 21. – No. 2. – Pp. 120-126. DOI: 10.1145/359340.359342

21 N. Cheng, H. Li, S. Wu and K. Xu. Edge Computing-enabled Fine-grained Access Control Scheme for Resource Constrained IoT Devices // in IEEE Transactions on Industrial Informatics, 2022. – Vol. 18. – No. 2. – Pp. 961-971. DOI: 10.1109/TII.2021.3082762

22 Ж. Бидахмет, А. Уайда, А.Д. Майлыбаева, Д.К. Даркенбаев, С. Бекназаров, Д. Бағдаулет. Metasploit framework арқылы желі мен сервердегі осалдықтарды сканерлеу және операциялық жүйелерге қашықтан қол жеткізу // Вестник КазУТБ, 2024. – № 1(22). – 97-106 б.