An Authentication Algorithm for Sets of Spatial Data Objects

Wenhao Li[1], Chengliang Wang[1(✉)], Xiaobing Hu[2(✉)], Hongwen Zhou[3], Hang Zeng[3], and Yanai Wang[4]

[1] College of Computer Science, Chongqing University, Chongqing, China
Wangcl@cqu.edu.cn
[2] College of Mathematics and Statistics, Chongqing University, Chongqing, China
iamhxb@cqu.edu.cn
[3] Chongqing Planning and Natural Resources Information Center, Chongqing, China
[4] Chongqing BI Academy, Chongqing, China

Abstract. Data play a pivotal role in supporting both national economic development and scientific research, and ensuring their authenticity and integrity has become a key concern for scholars. To this end, various authentication algorithms, such as digital signatures, message authentication codes, and hash functions have been proposed. Many of these algorithms treat data as a unity, which can lead to different authentication results if the elements order of the data changes. However, spatial data object is a point set that lack a predetermined order, varying data acquisition and management systems used by operators may result in the same data appearing in different orders, and these approaches are no longer effective. Therefore, this study proposes an authentication algorithm for spatial data objects based on set nature. We consider spatial data objects as a data set, and each element in set will be transferred into a string data. Then, we use hash function SHA to compute each string data, and perform XOR operation to obtain message M. Finally, the final hash code is obtained by computing M using the SHA. Through tampering experiments, it has been shown this algorithm has the property of ignoring the data order, good sensitivity, diffusion, confusion, and security. This algorithm can overcome the limitations that rely on data order, contribute to improving the authentication process for spatial data objects, and enhance data integrity and reliability in various applications.

Keywords: Data Protection · Data Authentication · Spatial Data objects · Hash Function · Hash Authentication

1 Introduction

With the increasing volume and complexity of data, ensuring their authenticity and integrity has become a key concern for scholars. Spatial data objects are data that have spatial attributes and can represent geographic features or phenomena. They play a key role in supporting national economic development and scientific research, such as

H. Duan et al. (Eds.): SecureComm 2023, LNICST 567, pp. 461–475, 2025.
https://doi.org/10.1007/978-3-031-64948-6_23

urban planning, environmental monitoring, disaster management, and public health [4]. Authentication is a process of verifying the identity and validity of data sources and contents, and preventing unauthorized modification or tampering.

There are some approaches about data authentication, with the broad categories of hash functions [1–6], message authentication codes [7–12], digital signatures [13–15], certificate-based authentication [16–21], token-based authentication [22–26], and third-party access [27–31]. Many of these algorithms treat data as a unity, which can lead to different authentication results if the elements order of the data changes.

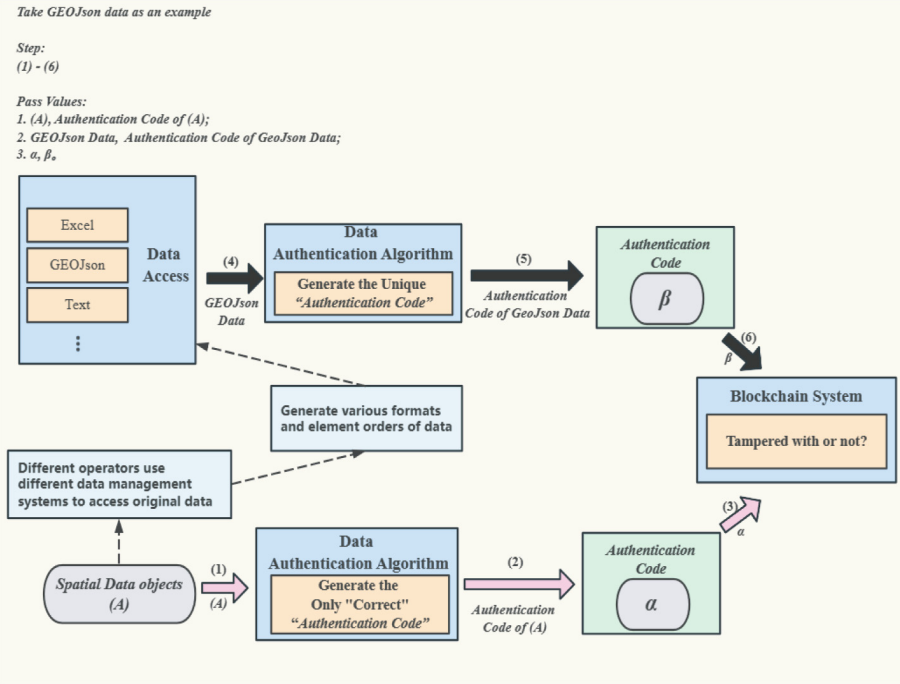

Fig. 1. The process of authenticating spatial data objects in blockchain (Take GEOJson data as an example)

In particular, when using blockchain to protect spatial data objects privacy, it is necessary to upload the algorithm result to the chain and later use this result as the basis for verification work. Blockchain is a distributed ledger technology with features of decentralization, security, trustworthiness, tamper-proofing and programmability. It can trace the source of tampering while detecting tampering during the verification process, thus assisting in precise crackdowns and punishment of interpolator. However, spatial data object is a point set that lack a predetermined order, different data collection and management systems used by operators may result in the same data appearing in a different order. This can lead to different results after algorithmic processing. As a result, it may not match the verification basis initially uploaded within the blockchain, which can result in a misjudgment of tampering behavior.

Figure 1 takes GEOJson data as an example to show the process of authenticating spatial data objects in blockchain. First, a unique and correct authentication code "α" is generated using the authentication algorithm and uploaded to the blockchain as the basis for subsequent authentication. Second, different operators use different data management systems to access original data, which generates various formats and element orders of data (Here is GEOJson data). The authentication code "β" is generated using these data and uploaded to the blockchain. Finally, the blockchain system checks "α" and "β" to determine whether the data has been tampered with. However, existing algorithms consider data as a unified whole and do not allow modification of element order in data. As a result, "α" and "β" are not equal, and then the same space data object will be identified as having been tampered with.

Be more specific, in the planning work of the two-dimensional region shown in Fig. 2. (left), the same five points A-I are changed and the traversal order between them is changed, and the human authentication result is that the planning region is unchanged, but the authentication result is that the region is tampered when the five points with different order are given to an algorithm sensitive to the data input order, such as (a) A to I and (b) F to G. Accordingly, in the actual data operation and management process, factors such as different systems used by different managers, design differences in databases and coding differences in operating systems may lead to changes in order during data operation, and this pitfall limits the generality of authentication algorithms among different systems to a certain extent.

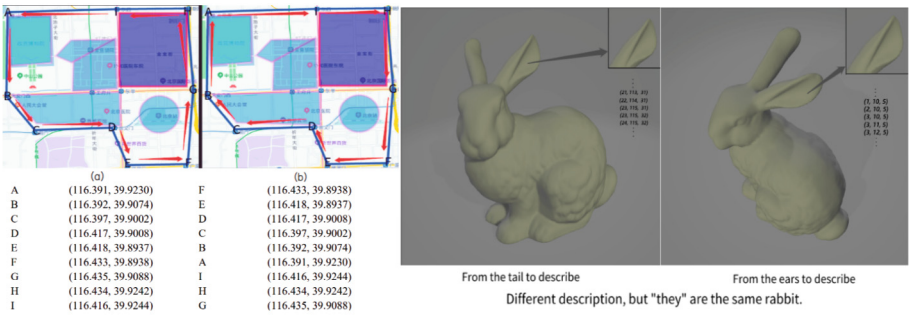

Fig. 2. Different access order for same 2D region (left) and same point cloud rabbit (right)

We can understand this better from the perspective of 3D objects. For example, when drawing a rabbit in the point cloud shown in Fig. 2 (right), we can start with the ears or the tail depending on the data set, but this does not affect the final generated rabbit.

Therefore, there is a need for an authentication algorithm that can overcome the limitations of input data order and improve the authentication process for spatial data objects. And in this paper, we propose an authentication algorithm for spatial data objects sets based on set nature. The main idea of this algorithm is to consider spatial data objects as a data set, and to use hash functions and XOR operations to generate a unique hash code for each set, regardless of the order of its elements. We describe the framework of this

algorithm in detail and conduct data tampering experiments to evaluate its performance in terms of ignoring input data order, sensitivity, diffusion, confusion, and security.

2 Contributions and Paper Structure

We propose an authentication algorithm, which has the potential to overcome the limitations of current algorithms that rely on input data order, and can contribute to improving the authentication process for spatial data objects, thus enhancing their integrity and reliability in various applications. And through tampering experiments, it has been shown this algorithm has the property of ignoring the data order, good sensitivity, diffusion, confusion, and security.

The structure of the paper is as follows: Sect. 3 introduces the authentication algorithm framework, including algorithm description, implementation of point cloud set authentication and parallel implementation; Sect. 4 provides the experiments of this algorithm, including uniqueness, sensitivity, confusion, diffusivity and security analysis; Sect. 5 explains some related works; Sect. 6 summarizes the work of this paper.

3 Algorithm Framework

3.1 Algorithm Description

The problem presented in the previous section is to authenticate for a set, and the order of the elements in the set should not have any effect on the authentication code. Then how to design a hash function (HF) for the set that is insensitive to the order of the elements is the key to this algorithm.

Definition 1: Let S be a set whose elements are a finite data set, i.e., $\forall A \in S, A = \{A_1, A_2, \cdots, A_m\}$, where $A_i (i = 1, 2, \cdots, m)$ is a data object, and call map $f : S \rightarrow GF(2)^N$ an hash function on the set S, where $GF(2)$ is a finite field and $GF(2)^N$ is an N-dimensional vector on field $GF(2)$.

Note 1: The input to the hash function is a set. Since the set has deterministic, unordered nature, it is required that changes in the order of arrangement of the elements in the set do not affect the value of the function.

Note 2: The data object A_i in the set A are not allowed to be repeated.

Note 3: N in $GF(2)^N$ can generally take *128, 196* and *256* as needed.

Definition 2: $\forall A \in S$, a collision is said to have occurred if it is possible to find another $B \in S$ such that $set\_hash(A) = set\_hash(B)$, where the *set_hash* is the hash function defined in Definition 1.

The hash code of a set is a fixed-length value that can be used as an authenticator generated when the set A is acted upon by function *set_hash*. The key to authenticating the set data is to design good *set_hash* functions and to require that for any $A \in S$, it is computationally infeasible to find a set $B \in S$ that collide with set A.

Algorithm Design Idea: Consider the set $A = \{A_1, A_2, \cdots, A_m\}$, where $A_i(i = 1, 2, \cdots, m)$ is the data object. First, each data object A_i in the data set is serialized to generate a sequence of characters closely related to the elements, and the hash function (SHA, SM3 etc.) is used to generate the hash code of each sequence to obtain the unique hash code D_i for each data object A_i. Then the hash code of all data objects is done as a bit-by-bit XOR operation, i.e., $M = D_1 \oplus D_2 \oplus \cdots \oplus D_m$.

And since the XOR operation satisfies the exchangeability, the value of M is independent of the order of the hash codes at each point. Finally, the final hash code HCode = *hash(M)* is obtained by computing M using the hash algorithm. The framework of creating hash code for set is shown in Fig. 3.

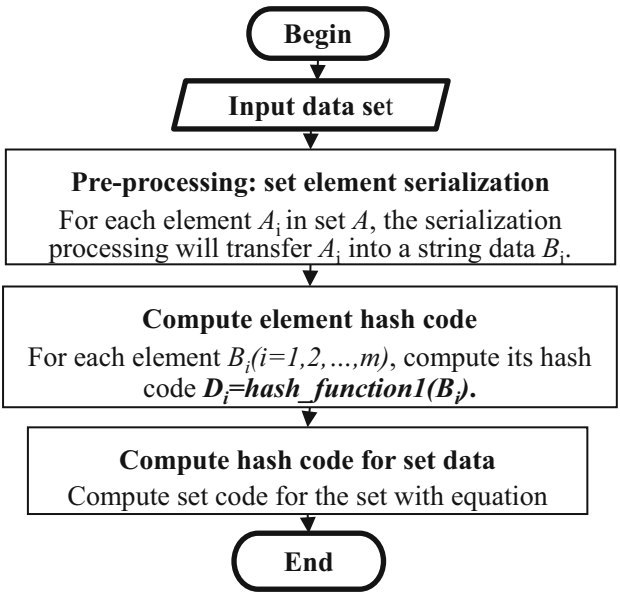

Fig. 3. The framework of creating hash code for set data

The flow of the framework is as follows:

1) Serialization of elements of data set

   The element $A_i(i = 1, 2, \cdots, m)$ in the set A may be a more complex data object, so in order to facilitate the next step to calculate the code of the element, it is necessary to do the serialization of the element A_i first, that is, to select a function $B_i = elem\_serialize(A_i)$ to convert the data object A_i to a string B_i. And function *elem_serialize* should be one-way, i.e., if $A_i \neq A_j$, then there must be $elem\_serialize(A_i) \neq elem\_serialize(A_j)$.

2) Compute hash code for the elements

   After serializing the elements $A_i(i = 1, 2, \cdots, m)$ in the set A, we get the set $B = \{B_1, B_2, \cdots, B_m\}$, whose elements $B_i(i = 1, 2, \cdots, m)$ are all strings. For each element in set B, an hash code $D_i(i = 1, 2, \cdots, m)$ is generated for the element, whose

length is N = 128, 192, 256 binary data. That is, $D_i = hash\_function1(B_i)$ $(i = 1, 2, \cdots, m)$. And the hash_*function*1 can be various types of hash functions such as SHA, SM3, etc., or other authentication code functions with keys.

3) Compute hash code for the data set

The hash code $D_i(i = 1, 2, \cdots, m)$ of all the elements will be done as an XOR operation, and then the authentication information of the data set A will be obtained after the hash run with the *hash_function*2, i.e., $\text{hcode}(A) = hash\_function2(D_1 \oplus D_2 \oplus \cdots \oplus D_m)$, where *hash_function*2 is optional for a certain hash function.

Note4: For the data set A, its elements should meet the basic property: the uniqueness of the set, that is, no duplicate elements are allowed in the set A. If there are duplicate elements present, such as $A_k = A_s$ in set $A = \{A_1, A_2, \cdots, A_m\}$, there are $hac(\{A_1, A_2, \cdots, A_m\}) = hac(\{A_1, \cdots, A_{k-1}, A_{k+1}, \cdots, A_m\})$. Therefore, if duplicate elements are allowed in data set A, attackers can add the same data objects in pairs to set A, and the hash authentication code will not be changed.

Note5: The complexity of the algorithm is closely related to the embedded hash algorithm. Let the spatial data set $A = \{A_1, A_2, \cdots, A_m\}$ and the element $A_i(i = 1, 2, \cdots, m)$ be serialized as B_i. General speaking, its length does not exceed the group length of the hash algorithm (e. g., SHA, SM3). Therefore, the $B_i(i = 1, 2, \cdots, m)$ produces two group lengths and a total of 2m group lengths will be produced. When calculating $hcode(A) = hash\_function2(D_1 \oplus D_2 \oplus \cdots \oplus D_m)$, the input $(D_1 \oplus D_2 \oplus \cdots \oplus D_m)$ also produces two group lengths, so the approximate time complexity of this hash algorithm is the hash algorithm that handles $(2m + 2)$ grouping length.

3.2 Implementation of Point Cloud Set Authentication

Using the previously mentioned algorithm ideas, the hash code is generated for 2D point cloud data, thus realizing the authentication work for such aggregate data.

Let the set $A = \{A_1, A_2, \cdots, A_m\}$, where $A_i = (a_{i1}, a_{i2}) \in R^2$, $(i = 1, 2, \cdots, m)$. Using the framework diagram of the hash algorithm in Fig. 3, the steps are as follows:

Step1: Serialize the set elements. Serialize the elements $A_i = (a_{i1}, a_{i2})$, $i = 1, 2, \cdots, m$ as $B_i = str(a_{i1}) + str(a_{i2})$, where the function str converts real numbers into strings, noting $B = \{B_1, B_2, \cdots, B_m\}$.

Step2: Calculate the element authentication code. Use the hash function SHA to operate on the elements in set B, and get the authentication code of all elements $D_i = SHA(B_i)$, where D_i is the binary data of N = 256bits.

Step3: Calculate the hash code for the set. The hash code $D_i(i = 1, 2, \cdots, m)$ of all the elements will be done as an XOR operation, and then the authentication information of the data set A will be obtained after a hash run using SHA algorithm, i.e., $SHA(D_1 \oplus D_2 \oplus \cdots \oplus D_m)$.

3.3 Parallel Implementation

This algorithm has a natural advantage for parallel implementations. If t machines are available, the parallel implementation can be performed as follows:

The point set $A = \{A_1, A_2, \cdots, A_m\}$ is partitioned according to the computing power of different machines to obtain the set $\{Q_1, Q_2, \cdots, Q_t\}$, where $Q_i (i = 1, 2, \cdots, t)$ is a non-empty subset of the set A with t $\leq m$ and satisfies $A = Q_1 \cup Q_2 \cup \cdots \cup Q_t$, $Q_i \cap Q_j = \varnothing, (i \neq j, 1 \leq i, j \leq t)$.

Put the subset $Q_i (i = 1, 2, \cdots, s)$ into the i-th computer for computation: first compute the hash codes of all elements in subset Q_i, and then do the XOR operation on all hash codes to get the authentication information hcode_i in subset Q_i.

Compute the hash code for the point set $A = \{A_1, A_2, \cdots, A_m\}$ according to Eq. (3.1).

$$hcode(A) = SHA(hcode_1 \oplus hcode_2 \oplus \cdots \oplus hcode_t) \tag{3.1}$$

Since the computation on each machine is completely independent and the final computation is just the result of (3.1), the algorithm is well suited for parallelism implementation, which is extremely beneficial for the authentication of large-scale flat data sets.

4 Algorithm Framework

The hash function is a one-way function that can compress any finite-length message plaintext into a fixed-length Hash value, and it is computationally difficult to find its inverse mapping [9]. In this paper, firstly, it also needs to satisfy the insensitivity to the element order of the set, so it needs to test whether the encryption result of the algorithm is unique under the random order. Secondly, reasonable and secure hash algorithms usually have good sensitivity, confusion and diffusion, so it is necessary to evaluate the reasonableness and security of the algorithm by testing these properties.

In these experiments, we use two-dimensional data from a real region planning exercise as the experimental data for testing the algorithm, which consists of a total of 4464 points.

4.1 Uniqueness Analysis of Authentication Results

The experiment was performed using a dataset consisting of 4464 points with 44 rounds of data selection, with 100 points added in each round and 10 random order encryptions in each round, where 164 points were added in the 44th round because the number of follow-ups was less than 100. And it recorded the number of data point in each round, the number of randomizations, and whether the results were unique (true if they met the requirements, false if they did not).

The results show that the algorithm still has uniqueness under the random order, so it satisfies the requirement of "insensitive to the order of the elements of the set", and the specific results can be referred to Table 1.

4.2 Sensitivity Analysis of Algorithm

Sensitivity refers to the degree of sensitive dependence of the hash value on the plaintext of the message. If every slight change in the message plaintext will bring a significant

Table 1. Table for uniqueness analysis results (N is number of turn)

Turns	Num of Points	Uniqueness
1	100	TRUE
…	…	…
N	N * 100	TRUE
44	4464	TRUE

change in the encryption result, i.e., the algorithm has a high initial value sensitivity, which in turn indicates that the algorithm has a good one-way hashing performance.

In this experiment, a small change was made to the coordinate value of a dimension of a point in the original data, and the change range was 1E-05, 1E-04 or 1E-03 increase in the coordinate value, and each change operation was repeated 256 times, and finally the average number of bits changed and its sample standard deviation were obtained by comparing the changed results with the original results, and the specific results are shown in Table 2.

Table 2. Table for effect of different levels of minor changes on the results (number of bits changed)

Small Change	1E-05	1E-04	1E-03
Number of Bits to Change (Average)	64.42	63.71	63.69
STD (Sample Standard Deviation)	5.72	5.51	5.66

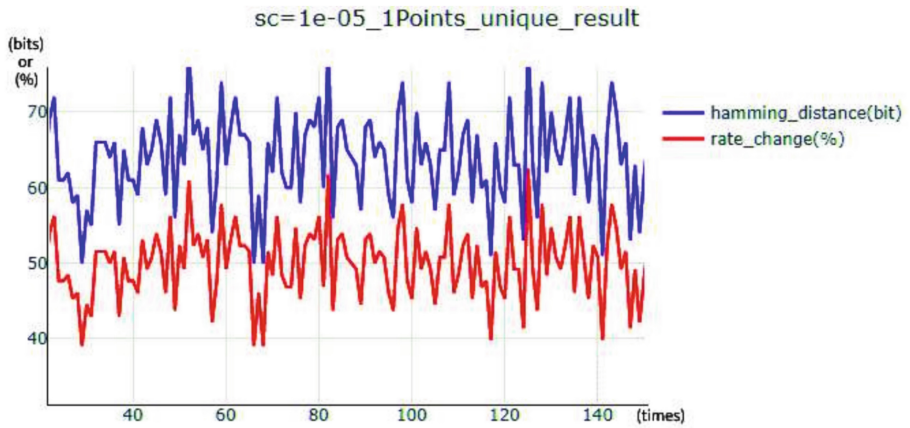

Fig. 4. 256 times 1E-05 minor change bit distribution chart. (Color figure online)

The experimental results show that a small change like 1E-05 of the plaintext data leads to a huge change in the result, so the algorithm has a good initial sensitivity. The

specific sensitivity experimental record is shown in Fig. 4, which shows the bit change distribution under 256 times 1E-5 changes, and the two rows record the number of changed bits (blue line) and the bit change rate under total 128 bits (red line), respectively.

4.3 Confusion and Diffusivity Analysis of Algorithm

Diffusivity means that a change in each bit of the plaintext of the designed function affects a change in many bits of the ciphertext. For the binary result, there are only two possibilities of 1 or 0 per bit, an excellent diffusion effect manifests when a small change in the initial value causes a 50% probability of change in each bit of result. In addition, confusion means that the designed algorithm should make the dependencies between the plaintext and the ciphertext quite complex.

The specific quantitative indicators are as follows:

- Mean changed bit number

$$\overline{B} = \frac{1}{E}\sum_{i=1}^{|E|} B_i \tag{4.1}$$

- Mean changed probability

$$\overline{P} = \frac{\overline{B}}{L} \times 100\% \tag{4.2}$$

- Sample standard deviation of the two indicators

$$\Delta B = \sqrt{\frac{1}{E-1}\sum_{i=1}^{|E|}\left(B_i - \overline{B}\right)^2} \tag{4.3}$$

$$\Delta P = \sqrt{\frac{1}{L-1}\sum_{i=1}^{|E|}\left(P_i - \overline{P}\right)^2 \times 100\%} \tag{4.4}$$

In these equations, B is the number of change bits in the authentication message of 128 bits initially obtained after each tampering, E is the number of times each tampering operation is performed, P is the rate of change in the authentication message of 128 bits initially obtained after each tampering, and L is the length of the message, which is 128 bits here in the experiment. ΔB, ΔP marks the stability of the hash confusion and diffusion properties, the smaller is the more stable, if the calculated Δ are small, it means that the algorithm is strong and stable to the confusion and diffusion of plaintexts.

Therefore, we test the algorithm's confusion and diffusion by tampering with two-dimensional data in different ways, such as adding, deleting, and replacing, and recording the corresponding metrics. For each way, we randomly select a point and repeat 2048 times.

1) Add 1 arbitrary point
The experimental results are shown in Fig. 5, which plots the distribution of B and P for increasing this tampering method. The blue line represents B and the red line represents P. Excellent diffusion effect should change with 50% probability for each bit, so P should be close to 50% as well as B should be close to 64.

The specific indicators are given in Table 3. ΔB, ΔP marks the stability of the hash confusion and diffusion properties, and if the calculated Δ are small, it means that the algorithm is strong and stable against the confusion and diffusion of plaintexts.

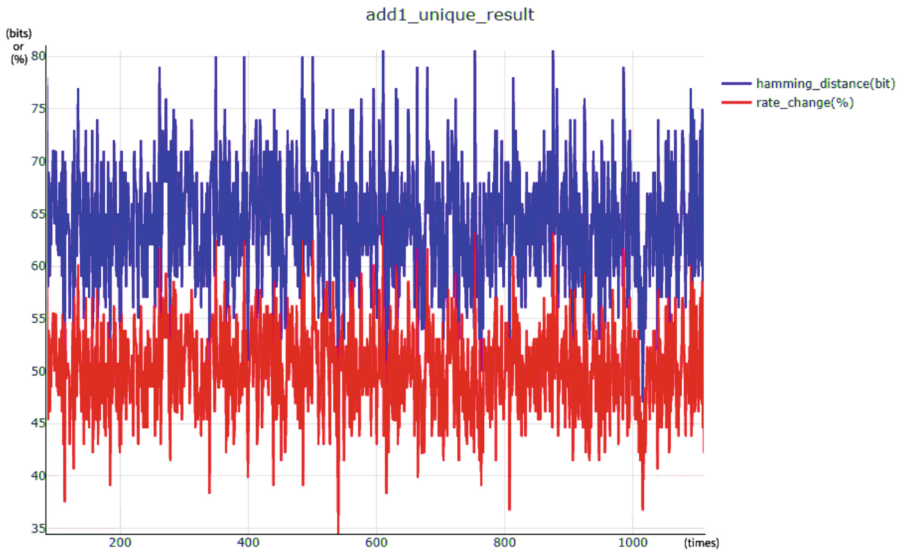

Fig. 5. Distribution of B and P (Add) (Color figure online)

Table 3. Statistics of the number of bit changed B of the E times test (Add)

Indicators/E	256	512	1024	2048	Total
$\overline{B}$	63.71	64.36	64.07	64.09	64.06
ΔB	5.25	5.58	5.72	5.73	5.57
$P/\%$	49.77	50.28	50.05	50.07	50.04
$\Delta P/\%$	4.1	4.36	4.47	4.47	4.35

According to Fig. 5 and Table 3, it can be concluded that the algorithm change rate P is stable around 50% with an average of 50.04% and the number of change bits

B is stable around 64 with an average of 64.06. And Δ is also smaller with an average of 5.57 and 4.35, respectively, indicating that the algorithm has a better ability of confusion and diffusion to increase this kind of tampering.

2) Remove 1 arbitrary point

The experimental results are shown in Fig. 6, and again the distribution of B and P for removing this tampering method is plotted. Again, the specific indicators for the removal experiments are given in Table 4 in this paper.

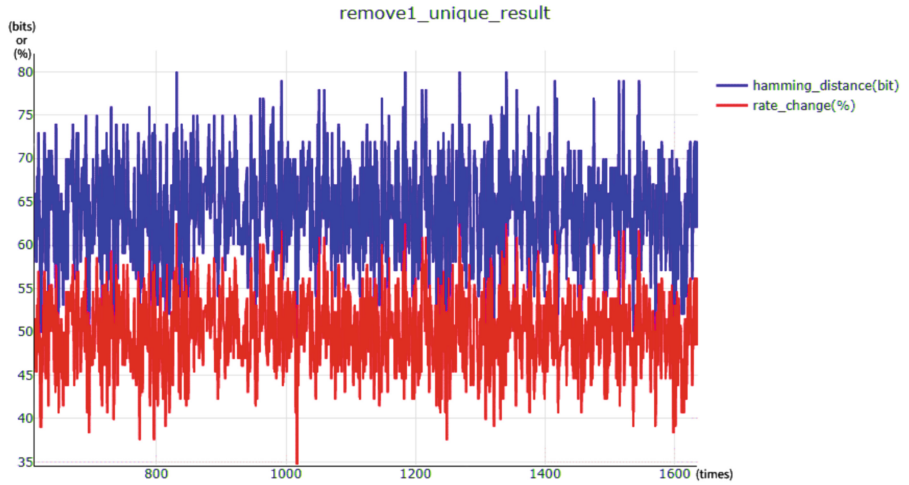

Fig. 6. Distribution of B and P (Remove)

Table 4. Statistics of the number of bit changed B of the E times test (Remove)

Indicators/E	256	512	1024	2048	Total
$\overline{B}$	64.74	64.39	64.04	64.23	64.39
ΔB	5.93	5.75	5.73	5.70	5.80
P/%	50.60	50.31	50.03	50.18	50.31
ΔP/%	4.63	4.49	4.48	4.45	4.53

According to Fig. 6 and Table 4, the algorithm change rate P averaged 50.31% and the number of change bits B averaged 64.39. While Δ averaged 5.80 and 4.53, respectively, indicating that the algorithm also has good confusion and diffusion ability for removing this type of tampering.

3) Replace 1 arbitrary point

The experimental results are shown in Fig. 7, and again the distribution of B and P for replacing this tampering method is plotted. And the specific indicators for the replacement experiments are given in Table 5 in this paper.

According to Fig. 7 and Table 5, the algorithm change rate P averages 50.1% and the number of change bits B averages 64.13. While Δ averages 5.71 and 4.47,

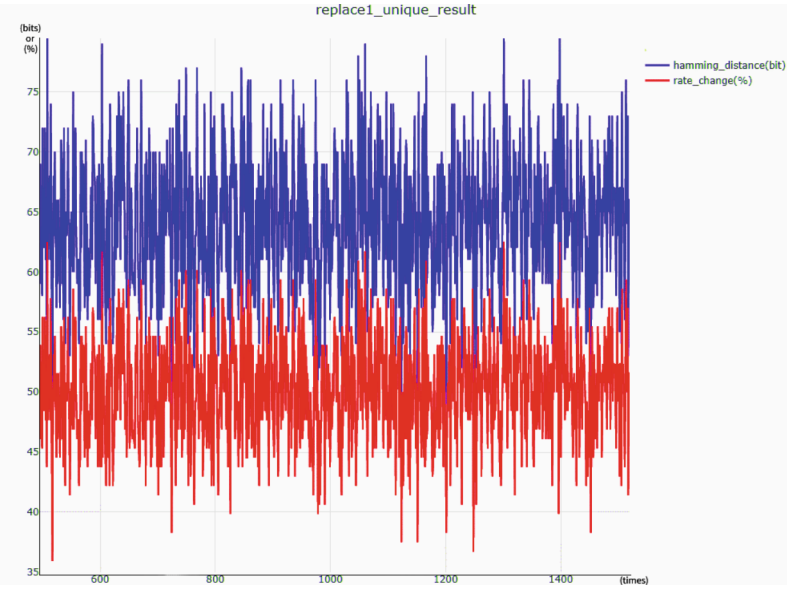

Fig. 7. Distribution of B and P (Replace)

Table 5. Statistics of the number of bit changed B of the E times test (Replace)

Indicators/E	256	512	1024	2048	Total
$\overline{B}$	64.11	64.12	64.04	64.23	64.13
ΔB	5.74	5.68	5.73	5.7	5.71
$P/\%$	50.09	50.09	50.03	50.18	50.10
$\Delta P/\%$	4.49	4.44	4.48	4.45	4.47

respectively, indicating that the algorithm also has good confusion and diffusion ability for replacement as a tampering method.

4) Comprehensive conclusion
Finally, combining the three sets of experiments, although the tampering methods are different, the algorithm change rate P is stable at around 50% and the number of change bits B is also stable at around 64. And Δ is also smaller, i.e., the algorithm has better confusion and diffusion capability for data plaintexts.

4.4 Security Analysis of Algorithm

The algorithm uses the fundamentals of the SHA hash function to produce two distinct hash codes. The security of SHA has been verified by the current attacks against it, as reported by Sahu, A., et al. [32] and Gilbert, H., et al. [33], etc. Therefore, Therefore, the hash function proposed in this paper can defend against hash collisions for birthday attacks [34] in cryptography, i.e., the security of the algorithm is reliable.

5 Related Work

There are some approaches about data authentication, with the broad categories of hash functions [1–6], message authentication codes [7–12], digital signatures [13–15], certificate-based authentication [16–21], token-based authentication [22–26], and third-party access [27–31]. However, many of these algorithms treat data as a unity, which can lead to different authentication results if the elements order of the data changes.

Therefore, we propose this algorithm that does not depend on input data order, and can contribute to improving the authentication process for spatial data objects, thus enhancing their integrity and reliability in various applications.

6 Conclusion

In this paper, in order to reduce and avoid the impact of cumbersome and erroneous data input order on the authentication process and results, we proposed a novel authentication algorithm for spatial data object sets that is independent of the order of the input data. The algorithm transforms spatial data objects into string data and then applies the hash function SHA to generate a unique hash code for each string data. The hash codes are then combined using XOR operation and hashed again to produce the final hash code. We also tested the algorithm and verified its sensitivity, diffusion, confusion, and security. The proposed algorithm can enhance the integrity and reliability of spatial data objects in various applications.

Acknowledgement. This work is supported by the Chongqing Technology Innovation & Application Development Key Project (No. cstc2020jscx-dxwtBX0055; cstb2022tiad-kpx0148) and the Fundamental Research Funds for the Central Universities (No. 2022CDJYGRH-001).

References

1. Butin, D., Wälde, J.: Post-quantum authentication in OpenSSL with hash-based signatures. In: 2017 Tenth International Conference on Mobile Computing and Ubiquitous Network (ICMU), pp. 3–5. IEEE (2017)
2. Hedam, N., Mollerup, J., Tözün, P.: Hash-based authentication revisited in the age of high-performance computers. In: International Workshop on Accelerating Analytics and Data Management Systems: ADMS 2020, Tokyo, Japan (2020)
3. Han, S., Xu, K., Zhu, Z., Guo, S., Liu, H., Li, Z.: Hash-based signature for flexibility authentication of IoT devices. Wuhan Univ. J. Nat. Sci. **27**(1), 1–10 (2022)
4. Lei, H., Xin-mei, L., Song-he, J., Zeng-yu, C.: A one-way Hash based low-cost authentication protocol with forward security in RFID system. In: 2010 2nd International Asia Conference on Informatics in Control, Automation and Robotics (CAR 2010), pp. 269–272. IEEE (2010)
5. Zhang, Y., Wang, Y.: A hash-based authentication protocol for IoT devices. In: 2019 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC), vol. 1, pp. 1–6. IEEE (2019)
6. Zhang, Y., Wang, Y., Liang, X.: Hash-based signature for flexibility authentication of IoT devices. Wuhan Univ. J. Nat. Sci. **26**(1), 1–10 (2021)

7. Liang, K., Susilo, W., Liu, J.K., Xiang, Y. (eds.): Information Security and Privacy: 22nd Australasian Conference, ACISP 2017, Auckland, New Zealand, 3–5 July 2017, Proceedings, Part I. Springer, Cham (2017)
8. Wang, X., Yu, S., Zhang, Y., Liang, J.: A lightweight message authentication code algorithm for RFID systems based on quasigroups and chaotic maps. IEEE Access **8**, 11469–11478 (2020)
9. Van, D.H., Thuc, N.D.: A privacy preserving message authentication code. In: 2015 5th International Conference on IT Convergence and Security (ICITCS), pp. 1–6. IEEE (2015)
10. Ogawa, Y., Sato, S., Shikata, J., Imai, H.: Aggregate message authentication codes with detecting functionality from biorthogonal codes. In: 2020 IEEE International Symposium on Information Theory (ISIT), pp. 868–873. IEEE (2020)
11. Wagner, E., Serror, M., Wehrle, K., Henze, M.: BP-MAC: fast authentication for short messages. In: Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec 2022), pp. 201–206. ACM (2022)
12. Bellare, M., Canetti, R., Krawczyk, H.: Keying hash functions for message authentication. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 1–15. Springer, Cham (1996). https://doi.org/10.1007/3-540-68697-5_1
13. Kasodhan, R., Gupta, N.: A new approach of digital signature verification based on BioGamal algorithm. In: 2019 3rd International Conference on Computing Methodologies and Communication (ICCMC), pp. 1–6. IEEE (2019)
14. Shestopal, S.S., Lobach, D.V., Smirnova, E.A.: The international legal regulation of a digital signature: Asia-Pacific region experience. In: European Proceedings of Social and Behavioural Sciences EpSBS, vol. 126, pp. 1–10 (2021)
15. Alshahrani, A., Brinkworth, R.: Digital signature scheme over lattices. In: 2019 IEEE International Conference on Communications, Signal Processing, and Their Applications (ICCSPA), pp. 1–5. IEEE (2019)
16. Badhe, V., Nhavale, P., Todkar, S., Shinde, P., Kolhar, K.: Digital certificate system for verification of educational certificates using blockchain. Int. J. Sci. Res. Sci. Technol. **7**(5), 45–50 (2020)
17. Al-Mutairi, A., Al-Mutairi, M.: Certificate-based authentication for mobile devices. Int. J. Comput. Sci. Netw. Secur. **19**(1), 1–7 (2019)
18. Kaur, R., Singh, A.: Certificate based authentication in MANETs: a survey. In: 2016 International Conference on Computing, Communication and Automation (ICCCA), pp. 1222–1227. IEEE (2016)
19. Chen, L., Li, H.: A certificate-based authentication scheme for wireless sensor networks. In: 2010 IEEE International Conference on Wireless Communications, Networking and Information Security, pp. 1–5. IEEE (2010)
20. Liu, J., Xiao, Y., Chen, C.L., Xie, L., Li, S.: CHAP: a certificate-based host authentication protocol for VANETs. In: 2009 IEEE International Conference on Communications, pp. 1–5. IEEE (2009)
21. Zhang, Y., Fang, Y.: ARSA: an attack-resilient security architecture for multihop wireless mesh networks. IEEE J. Sel. Areas Commun. **24**(10), 1916–1928 (2005)
22. Alzahrani, A., Alshammari, A., Alshammari, T.: A survey on token-based authentication for web applications. Int. J. Comput. Sci. Netw. Secur. **19**(1), 1–10 (2019)
23. Bhatti, M.A., Khan, M.A.: Token based authentication for RESTful web services. In: 2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET), pp. 1–6. IEEE (2018)
24. Chaudhary, S., Kumar, R.: Token based authentication for securing RESTful web services. In: 2017 International Conference on Computing and Communication Technologies for Smart Nation (IC3TSN), pp. 1–5. IEEE (2017)

25. Kaur, G., Singh, S.: Token based authentication for securing RESTful web services using JWT. In: 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), pp. 1–6. IEEE (2020)
26. Liang, X., Shetty, S., Tosh, D., Kamhoua, C., Kwiat, K., Njilla, L.: ProvChain: a blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability. In: 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), pp. 468–477. IEEE (2017)
27. Almuhimedi, H., et al.: Your location has been shared 5,398 times!: A field study on mobile app privacy nudging. In: Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI 2015), pp. 787–796. ACM (2015)
28. Balebako, R., Leon, P.G., Shay, R., Ur, B., Wang, Y., Cranor, L.F.: Measuring the effectiveness of privacy tools for limiting behavioral advertising. In: Web 2.0 Security and Privacy Workshop (2014)
29. Chen, J., Reinecke, K., Gajos, K.Z.: Who are the turkers? Worker demographics in amazon mechanical turk. In: Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, pp. 1–14 (2019)
30. Datta, A., Tschantz, M.C., Datta, A.: Automated experiments on ad privacy settings: a tale of opacity, choice, and discrimination. In: Proceedings on Privacy Enhancing Technologies, pp. 92–112 (2015)
31. Egelman, S., Harang, R.E., Pearman, S., Peebles, D., Consolvo, S.: Constrained consent: moving beyond mutual agreement in research intermediaries. In: Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, pp. 653–664 (2016)
32. Sahu, A., Ghosh, S.M.: Review paper on secure hash algorithm with its variants. ResearchGate (2017)
33. Gilbert, H., Handschuh, H.: Security Analysis of SHA-256 and sisters. In: Matsui, M., Zuccherato, R.J. (eds.) SAC 2003. LNCS, vol. 3006, pp. 175–193. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24654-1_13
34. Soltanian, M.R.K., Amiri, I.S.: Problem solving, investigating ideas, and solutions. In: Soltanian, M.R.K., Amiri, I.S. (eds.) Theoretical and Experimental Methods for Defending Against DDOS Attacks, pp. 33–45. Syngress (2016)