A Security Enhanced Verification Framework Based on Device Fingerprint in Internet of Things

Shichen Fu[1], Liuping Huang[1], and Guangyan Zhang[2(✉)]

[1] State Grid Zhangzhou Electric Power Supply Company, Zhangzhou, China
[2] Northeastern University, Shenyang, China

Abstract. IoT device fingerprint is a critical technology for device identification and access control. It is also the first step to test the protection ability of IoT devices. Existing schemes tend to deploy the verification server in the cloud, which leads to problems such as high bandwidth pressure, failure to update the device fingerprint normally, slow verification speed and others leading to security risks. To solve the problems, this paper presents a security enhanced verification framework based on device fingerprint in IoT to reduce the bandwidth pressure, computational overhead. A device fingerprint updating mechanism is proposed to distinguish normal devices from abnormal devices. Meanwhile, this paper proposes an efficient device fingerprint matching algorithm, which realizes the compressed storage and fast matching of massive device fingerprints.

Keywords: IoT · Device fingerprint · Security · Identity authentication

1 Introduction

With the development of industrial intelligence, the number and types of Internet of Things (IoT) devices are increasing rapidly. These IoT devices with weak security protection provide a huge and extensive network attack entrance for attackers, leading to many problems and challenges facing the Internet of Things [1–6]. In July 2019, several employees of a nuclear power plant near the city of Yuzh-Noukrainsk in southern Ukraine experienced a serious security breach when they connected the plant's internal network to a public network for cryptocurrency mining. The incident has been classified as a breach of state secrets, and investigators have begun looking into whether internet-connected devices could be used by hackers to break into the plant's internal network and steal confidential information. How to improve the security of IoT devices is an urgent problem to be solved.

Authentication of devices is a critical step in protecting the Internet of Things [7–9]. The existing authentication methods can be divided into password-based

W. Shi et al. (Eds.): SPNCE 2021, LNICST 423, pp. 140–158, 2022.
https://doi.org/10.1007/978-3-030-96791-8_11

authentication [10], cryptographic protocol-based authentication [11,12], and device fingerprint-based authentication [13–16]. Since the verification technology based on device fingerprint has a small demand on the computing power and storage capacity of the device, it is more suitable for resource-constrained devices in the Internet of Things environment. In this paper, the source address verification technology based on device fingerprint is adopted. However, through the study of the existing methods, it is found that most of the device fingerprints extracted in the existing work are based on the device hardware attributes, such as the transient characteristics of the device switch and machine, the modulation signal, etc. In practical applications, these attributes are affected by noise and interference in the environment, which makes the fingerprint easy to change [17]. By observing the IoT devices, it is clear that the response headers returned by the devices often carry device information to distinguish services. Therefore, this paper establishes a numerical device fingerprint based on the response headers of the IoT devices, which has the advantages of uniqueness and stability.

Meanwhile, the existing authentication methods need to do an identity match on the authentication request sent by the device [18,19]. If the match is successful, then the authentication is passed, and after that, the device identity is no longer verified. As more and more devices are connected to the Internet of Things, there are more and more types of devices and more and more complex types. Therefore, for the network with numerous devices, complex structure and difficult management, the attacker is often easier to find the place with weak defense, and can easily forge the device identity and enter the network by pretending to be a legitimate device. Therefore, even if the device identity passes the authentication, it is not necessarily credible. As a supplement to the existing authentication methods, this paper argues that the authentication terminal should not trust the device except the authentication request proposed by the device, and the authentication server should verify the identity of the current networked device again, so as to enhance the protection of the identity security of the device.

This paper presents a security enhanced authentication method based on device fingerprint, through the fingerprints are consistent alignment device, which can validate device identity, but as a result of actual IoT environment, authentication server deployment in the cloud environment, so this method is still facing the following challenges: the bandwidth pressure, fingerprint cannot normal slow speed matching, validation.

To solve the problem of high bandwidth pressure and device fingerprint can not match properly, this chapter proposes a security enhancement verification framework based on device fingerprint. Aiming at the problem of slow verification speed, this paper designs and implements an efficient fingerprint matching algorithm for devices. Finally, the experiment proves that the scheme is efficient. The main contributions of this paper are as follows:

(1) To solve the problem of untrusted devices, a security enhanced authentication framework based on device fingerprint is proposed, which takes the

authentication server as the requester, and the gateway provides the authentication service to enhance the security.
(2) Aiming at the storage and matching problem of device fingerprint, an efficient device fingerprint matching algorithm is proposed. This algorithm is oriented to device fingerprint and improves the inherent defects in the basic bloom filter. It has the advantages of extensibility, low false positive rate and fast verification speed.

Organization. The remainder of this paper is organized as follows. Section 2 defines the problems of identity authentication for IoT devices under cloud services. Section 3 presents the proposed scheme in detail. Section 4 introduces an efficient device fingerprint matching algorithm. Section 5 presents the performance analysis of the scheme. Finally, the conclusions are drawn in Sect. 6.

2 Problem Definition

The existing Internet of Things structure uses a cloud server as the management center of the entire Internet of Things environment, and the authentication server is also included to provide identity authentication services for devices. However, if this method is deployed in a cloud environment, the following challenges are still faced:

(1) High bandwidth pressure: The cloud authentication server scans the terminal devices of the entire network, and the returned scan data is too concentrated. If a denial of service attack occurs at this time, the connection between the cloud and the underlying device will be cut off;
(2) Device fingerprints do not match properly: If there is a device in the network that requires a software or hardware upgrade, the IP address of the normal device will change. Because the device fingerprint does not match, the inherently legitimate device will be considered malicious by default and will eventually be ejected from the network. When this happens, the common solution is to bulk upgrade the underlying devices or to change the network configuration, which will result in re-initialization of the registry due to a large area of network failure. On the one hand, it causes communication burden, on the other hand, it provides opportunities for attackers.
(3) Slow verification speed: When the target network is initialized, the cloud obtains and establishes the device fingerprint of each device. Meanwhile, when providing verification service, the cloud environment needs to recalculate the fingerprint of each device and carry out extraction, comparison, retrieval and other operations with the previously stored fingerprint. This imposes a significant computational overhead.

To solve the first and second problems, this paper proposes a security-enhanced authentication framework based on device fingerprints. The authentication service is migrated to the gateway for execution. The gateway provides authentication services to store and update device fingerprints, which can

improve retrieval efficiency. It reduces the bandwidth pressure of the cloud server, reduces the computational overhead, and proposes a device fingerprint update mechanism to distinguish between normal devices and abnormal devices, and solves the problem of normal devices matching device fingerprints. Aiming at the problem of slow verification speed, this paper designs and implements an efficient device fingerprint matching algorithm to realize fast matching while keeping accuracy.

3 Security Enhanced Verification Framework Based on Device Fingerprint in Internet of Things

This section proposes the design goals of a security-enhanced authentication framework based on device fingerprints under the Internet of Things, and on this basis, proposes an identity verification framework, and then introduces the basic execution process.

3.1 Design Goals

The framework proposed in this paper will be deployed in the Internet of Things environment, oriented to Internet of Things terminal device, so considering the particularity of the network environment and terminal device, in addition to the basic functions of enhancing the identity security of the device, the authentication technology should also have the following design goals:

(1) Low energy consumption: Most of IoT devices are resource-constrained, so the verification technology ought not to bring additional computing and storage overhead.
(2) Universality: One of the reasons for the poor deployment of existing identity verification methods is that the devices are highly heterogeneous, so the method proposed in this paper should be suitable for heterogeneous devices.
(3) High efficiency: Identity verification technology should start from the perspective of actual deployment, comprehensively consider the existing Internet of Things network structure, and propose a more reasonable and deployable identity verification technology.

3.2 System model

The core of the design of this model is to deploy identity verification services on smart gateways with application deployment capabilities. Such devices mostly appear in new network architectures proposed in recent years, such as edge computing and fog computing. Based on the smart gateway structure with application deployment capabilities, we proposes a security-enhanced authentication architecture based on device fingerprints, as shown in Fig. 1.

Three different entities are involved in this architecture: cloud, smart gateway, and IoT devices. Each entity is introduced as follows:

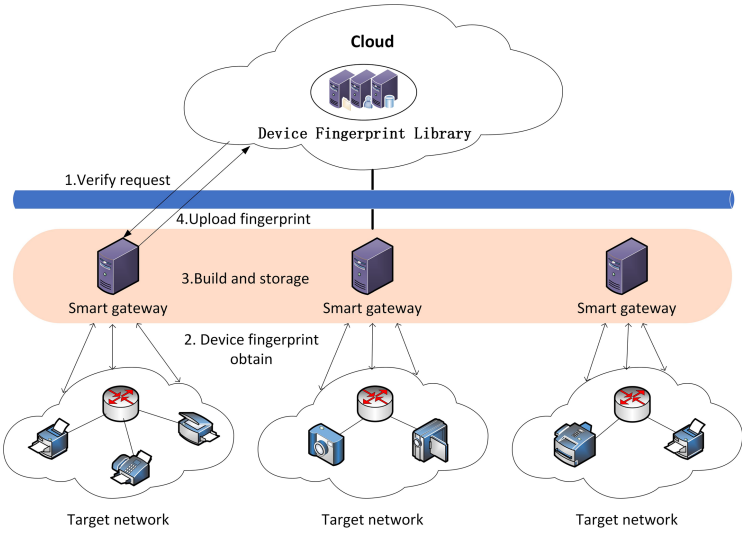

Fig. 1. Architecture diagram of identity verification system based on device fingerprint

(1) **Cloud.** Cloud is flexibility and on-demand charging. It has powerful storage and computing capabilities, but its bandwidth is limited. It cannot communicate with all sensor devices in the network environment at the same time. The cloud needs to store device fingerprints in the entire network environment and update the device fingerprint library in time.
(2) **Smart gateway.** The smart gateway is connected to the underlying sensor device and is deployed on the side of the network near the device. A smart gateway is connected to some sensor devices, and each sensor device is connected to only one smart gateway. That is, the smart gateway divides the entire terminal device set into multiple unrelated sub-sets. The gateway has certain computing and storage capabilities for device scanning, and completes the establishment and storage of device fingerprints, and provides identity verification services for its corresponding underlying devices.
(3) **IoT device.** The IoT device is an IP whose identity may be tampered with under attack. Upon receipt of an identity verification service request, data will be sent as required. Meanwhile, IoT device may be faced with software updates, hardware updates, and IP changes. At this time, the change information needs to be uploaded.

The main steps of identity verification in this framework are: identity verification request sending, device fingerprint acquisition, device fingerprint establishment, device fingerprint storage, and device fingerprint update.

(1) Identity verification request sending: the cloud server establishes the identity verification service and deploys it on the smart gateway, establishes a connection path between the cloud server and the smart gateway, and divides

the network where the underlying terminal device is located, so that the connected smart gateway can collect all The response message of the terminal device. Sending an identity verification request means that the verification server needs to verify the identity of all running devices in the network at this time, and the device responds accordingly after receiving the request.

(2) Device fingerprint acquisition: The smart gateway scans the device to the target network and receives the response information from the device. Through the incremental device scanning start-up time prediction algorithm, by sending verification requests to devices from time to time, new devices and frequently changing devices in the network can be discovered in time.

(3) Device fingerprint establishment: When scanning for the first time, the gateway uses the high-precision device fingerprint extraction algorithm to process the scanned data and establish the device fingerprint corresponding to the device, and set the device IP and device fingerprints according to a certain frequency Upload to the cloud to ensure that bandwidth fluctuations are as small as possible.

(4) Device fingerprint storage: To provide verification services faster, the gateway is responsible for storing device fingerprints of scanning devices and sending the device fingerprints to the cloud. The cloud builds a device fingerprint library for a large-scale IoT environment. When the device fingerprint database of the gateway is updated, the updated information is synchronized and uploaded to the cloud. Section 4 proposes an efficient device fingerprint matching algorithm to achieve compressed storage and fast matching of device fingerprints.

(5) Device fingerprint update: When the device is updated normally, it needs to submit a request to the gateway to update the device fingerprint. After receiving the update request, the smart gateway first retrieves the original device fingerprint in the device fingerprint database, and then calculates according to the uploaded original characteristics The fingerprint is compared with the stored fingerprint. If the match is successful, the device fingerprint will be recalculated, and the original fingerprint will be replaced in the device fingerprint library.

The process of identity verification is shown in Fig. 2. When the smart gateway reaches the scan start time and needs to verify the identity of the device, it sends a scan request to the terminal device. After receiving the scan request, the terminal device sends back corresponding response data. After the gateway receives the response data, it first uses similar devices based on hierarchical aggregation. The device fingerprint classification algorithm is used for clustering, and then the SVD-based device fingerprint dimensionality reduction algorithm is used to obtain the numerical representation of the device, and the two-tuple information of the device is obtained. Finally, an efficient device fingerprint matching algorithm is used to perform the two-tuple compressed storage and fast matching, and judge whether it is a new device or a device that is already running. If it is a new device, update the device fingerprint library and upload it to the cloud environment. If it is a running device and the device's fingerprint

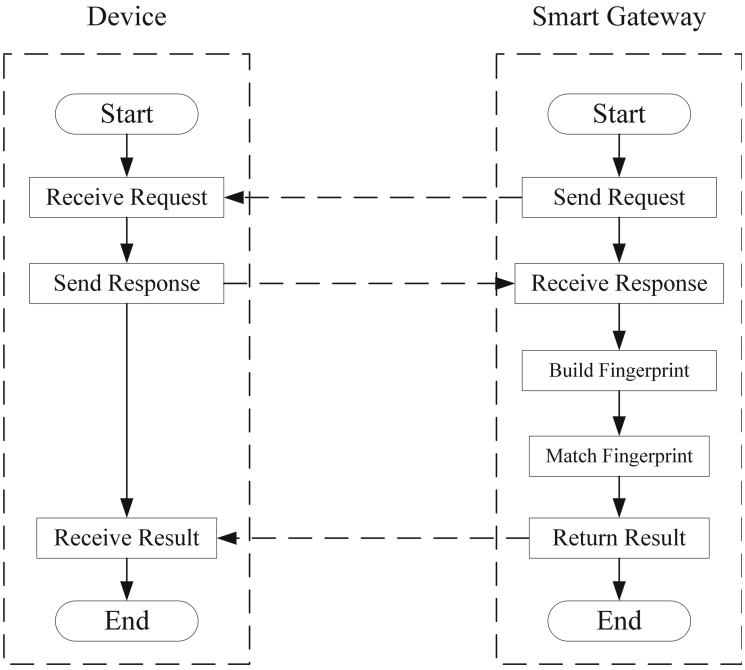

Fig. 2. Authentication flow chart

information has not changed, then the device identity is considered normal. If the device fingerprint match is unsuccessful and no fingerprint update request has been sent, the device identity is considered abnormal and its network connection is disconnected.

The framework proposed in this papermigrates the authentication service to the gateway to execute and obtains device fingerprints through network scanning. Network scanning is based on the principle of network communication. The device only needs to send corresponding response packets and does not require operations such as built-in identification or fingerprint calculation. Therefore, it does not bring additional computing overhead to the terminal, which is in line with the low energy consumption in the design goal; a unified numerical device fingerprint is established, which is in line with the versatility in the design goal. This paper proposes an incremental scanning method to meet the real-time requirement of identity verification services, and proposes a device fingerprint compression storage algorithm to improve the efficiency, so the method proposed in this paper meets the design goals.

4 Efficient Device Fingerprint Matching Algorithm

This section introduces an efficient device fingerprint matching algorithm, including a description of the research problem, an introduction to the basic principles of the algorithm, and a detailed description of the algorithm.

4.1 Problem Description

There are a large number of terminal devices in the IoT environment. This paper establishes device fingerprints for these devices on the gateway. How to store and match large amounts of data is the problem to be solved in this section.

To reduce the storage overhead and improve the verification speed, it is necessary to realize the compressed storage and fast matching of the device fingerprint library, so this paper chooses the bloom filter algorithm. Bloom filter is a kind of data structure, the realization principle of its compressed storage is: by selecting a set of hash mapping function and a bit space, using Hash function mapping, the longer elements are mapped to the bit space, which can achieve compression The purpose of the storage space.

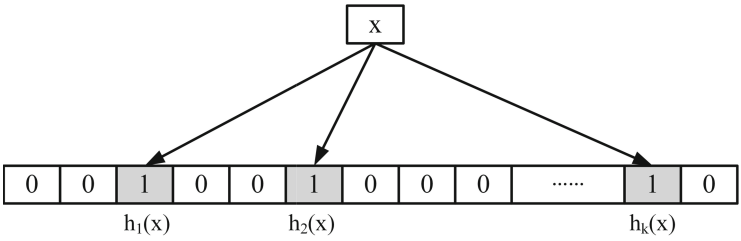

Fig. 3. Bloom filter element insertion process

The element insertion process of bloom filter is divided into two steps, as shown in Fig. 3:

(1) Calculate the k hash function values of the elements x, which are $h_1(x), h_2(x), \ldots, h_k(x)$, respectively.
(2) Search the bit space V and set the corresponding k bits of V to 1, namely

$$V[h_1(x)] = V[h_2(x)] = \ldots = V[h_k(x)] = 1$$

.

The element query process of bloom filter is:

(1) Calculate the k hash function value of the element y, $h_1(y), h_2(y), \ldots, h_k(y)$ respectively;
(2) Search the bit space V, if $V[h1(y)] = V[h2(y)] = \ldots = V[hk(y)] = 1$, it means that the element y to be queried already exists, if at least one of the hash function value is 0, it means that the element y does not exist.

Bloom filters have great advantages in data storage and query, but because the data is compressed and stored, it is difficult to avoid the problem of reduced accuracy. At the same time, basic bloom filters have inherent defects: (1) Weak of dynamic. This is because the number of hash functions and the length of the

bit space cannot be changed after it is determined; (2) Lack of scalability. The bit space can be preset according to the size of the data, but with storage With the continuous increase of data, the size of the stored data cannot be estimated. Therefore, if the preset value of the bit space is too small, it may cause conflicts, and if it is too large, it will cause a waste of space.

Extensible Counting CBF. (1) The scalable counting CBF is designed to solve the problem that the basic bloom filter cannot dynamically add or delete elements. It draws on the idea of counting bloom filters. In counting bloom filters, it is no longer a simple setting. Operation is to set a counter in the bit space. When a conflict occurs in the bit space, it is no longer a set operation, but to count the number of times the position is set to 1. When the element is deleted, the counter corresponding to the bit space is decremented by 1.

The size setting of the bit space can be preset according to the size of the data, but the number of IoT devices in the target network cannot be predicted in advance, and the storage capacity of the extracted device fingerprints is also unknown. In this case, if the bit space is set too large, it will cause a waste of space. If the bit space is set too small, the data storage capacity is insufficient, resulting in the lack of scalability of the basic bloom filter. We design a scalable counting, and its structure diagram is shown in Fig. 4, where CBF is counting bloom filter and $Counter$ denotes counter.

The implementation method of expandable counting CBF is shown as follow. As the element stored in the bloom filter, the two-tuple $<IP, device\ fingerprint>$ is added with a mark ID, which is a monotonously increasing sequence, so that the storage element becomes a triple-tuple $<ID, IP, device\ fingerprint>$, and the stored element is sequentially stored in the bloom filter. As the stored device fingerprints continue to increase, after the current bloom filter is full, create a new CBF. The sequence ID of the new filter is the maximum ID of the current filter plus 1. When the device fingerprint is inquired, the current filter group is inquired one by one. When the device fingerprint is deleted, the filter is determined according to the filter, and then the filter is reliably deleted. An example will be given below in conjunction with Fig. 4.

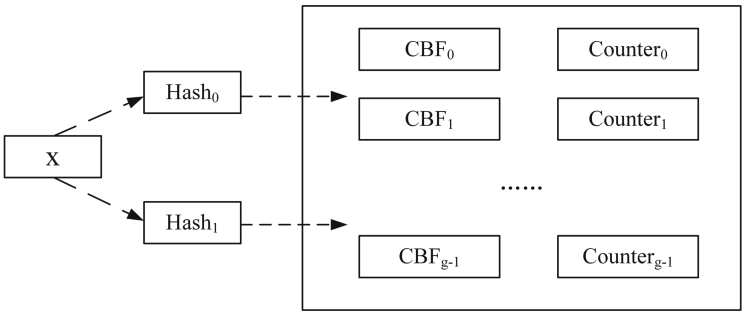

Fig. 4. Schematic diagram of bloom filter structure

Suppose the triples $<ID, IP, F>$ used for storage, where ID is a sequence of integers increasing from 0, IP denotes the IP address of the device, and F notes the device fingerprint corresponding to the address. There are 320 device fingerprints to be stored, and each CBF has a capacity of 100, which is obviously not enough.

Bloom filter initialization: only one CBF_0 is created during initialization, at this time the triplet is $<0, IP_0, F_0>$, and the ID is 0.

Device fingerprint insertion: There are 320 device fingerprints to be stored. When $ID \in [0, 99]$, the device fingerprint was inserted in the same way. When $ID = 99$, the maximum value of ID was updated to 99. Then the device fingerprints whose $ID = 100$ will continue to be inserted. At this time, the capacity of CBF_0 is full. Initialize $CBF_1 =<100, IP_0, F_0>$ and then continue to store device fingerprints in the device until all device fingerprints are stored in the bloom filter, the last established bloom filter is CBF_3, and the maximum value ID is updated to 319.

Device fingerprint deletion: Assuming that the device fingerprint is requested to be updated, the original device fingerprint needs to be deleted. First, you need to find its ID, and check in the order of CBF_3, CBF_2, CBF_1, CBF_0 in reverse order, and compare the CBF initialization ID and the device fingerprint ID. When the first initialization ID is less than or equal to the device fingerprint ID, it indicates that the device fingerprint is there, and then delete the corresponding device fingerprint according to the deletion method.

Device fingerprint query: The device fingerprint query process does not need ID. it just need to traverse the CBF. If the device fingerprint is found in any CBF, it is considered to exist; if not, it is determined that the device fingerprint does not exist.

Hash Function Split. Basic filters often use a set of hash functions and a bit space. As the number of fingerprints of storage devices increases, the conflict rate will also increase greatly, which affects the accuracy of the overall query. Therefore, this paper adopts the hash function split method. The hash functions are grouped, and the same group of hash functions are mapped to a bit space, so that the original mapping to one bit space can be changed to multiple bit spaces under the premise that the number of hash functions remains unchanged.

Assuming that the number of hash functions is h, dividing them into I groups requires I bit space. Each group has h/I hash values that is mapped to the same bit space. As shown in Fig. 5, four hash functions were selected to map the device fingerprint, which was split into two groups, and a total of two bit spaces were needed. The mapping values of $Hash_1$ and $Hash_2$ are storaged in the first bit space. The mapping values of $Hash_3$ and $Hash_4$ are storaged in the second bit space. By splitting the Hash function and adding bit space, the false positive rate will be greatly reduced. However, the increase of bit space will also increase the storage space of bloom filter. This paper analyzes the influence of the number of hash functions and the size of bit space on the false positive rate based on experiments in Sect. 5.

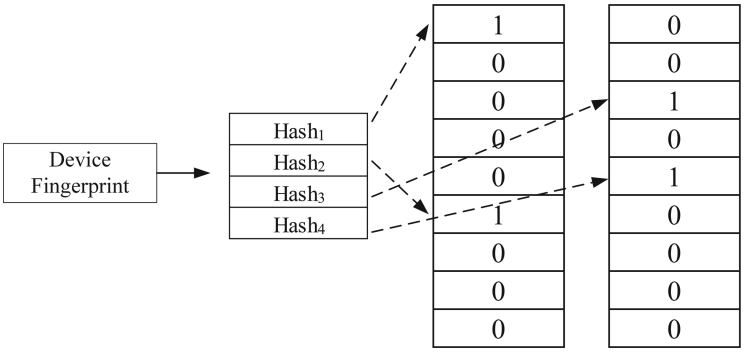

Fig. 5. Hash split mapping

Dual Feature Storage. This paper establishes the one-to-one correspondence between scanning IP and its corresponding device fingerprint. To use the device fingerprint for authentication, both the device's IP address and the device fingerprint need to be stored. If the IP is not found in the existing device fingerprint database, it is proved that the device corresponding to the IP is the new device to join the network. If the IP is found to exist and the device fingerprint is found at the same time, it is proved that the device fingerprint of the IP has not changed and the identity is credible. If the IP is searched but the corresponding fingerprint is not found, it proves that the IP's device fingerprint has changed and the identity is in doubt. To realize the simultaneous query of IP and corresponding device fingerprint, this paper chooses IP address and device fingerprint as two features to be stored simultaneously.

The efficient device fingerprint matching algorithm proposed in this paper is designed according to the above content. By designing a counting bloom filter, the scalability of the algorithm is increased to realize dynamic adding and deleting. Hash function splitting and dual feature storage are used to reduce the false positive rate. In the improved algorithm, each feature corresponds to a set of bitspaces, so the number of bitspaces is the number of hash groupings multiplied by the number of features. Of course, the device fingerprint database in this paper has two characteristics. The mapping process of the improved bloom filter algorithm is shown in Fig. 6.

4.2 Algorithm Description

This section introduces the basic operations of the device fingerprint matching algorithm, including the query, insertion, and deletion of device fingerprints, and the modification of device fingerprints. The method of deleting and inserting is used in this paper.

Device Fingerprint Query. During authentication, IP address and device fingerprint in duple $<IP, F>$ should be verified respectively. If each correspond-

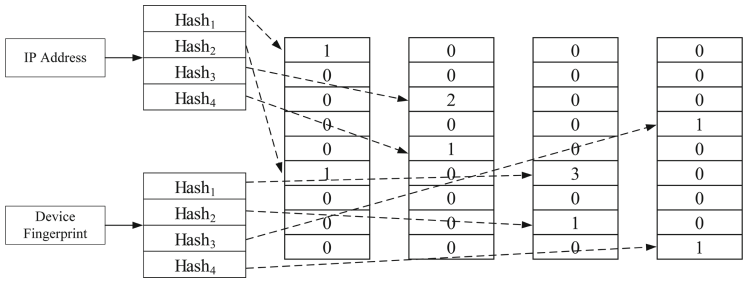

Fig. 6. Schematic diagram of the structure of the counting bloom filter based on multi-features

ing bit-space value of the IP address and device fingerprint is 1, the match is considered successful; otherwise, the match is considered unsuccessful. The main process of query algorithm is as follows: first query whether the IP address exists, if it does not, then return false, if it exists, then continue to verify its device fingerprint, if it exists, then return true, if it does not, then return false. The pseudo code of the query algorithm is shown in Algorithm 1.

Algorithm 1. Device fingerprint query algorithm

Input:
$< IP, F >$
Output:
Result

```
for j=0 to l − 1
  sum ← 0
  for i=0 to k − 1
    if CBF_j [hash_i (IP)] = 0x0000
      break
    else if CBF_j [hash_i (F)] = 0x0000
      break
    else
      sum++
  end for
  if sum == k
    return true
end for
return false
```

Device Fingerprint Insertion. The insertion process of device fingerprint is as follows: find the currently active filter CBF. If the identification ID is greater than the current maximum ID and the current CBF capacity is full, a new CBF will be created. At the same time, the total number of filters will be increased by 1, and the new CBF will take the maximum ID plus 1 as its own identifier

and set the number of saved device fingerprints to 0. If the identifier ID is less than the maximum value of the current ID, the identifier ID is updated, and the counter corresponding to the element after the hash is incremented by 1. If the counter overflows after incrementing by 1, it stays at $0x1111$ and returns False. The pseudo code of the insertion algorithm is shown in Algorithm 2.

Algorithm 2. Device fingerprint insertion algorithm

Input:
$< IP, F >$, $elementID$
Output:
$Result$
for $j = l - 1$ to 0
  $ActiveCBF \leftarrow CBF_j$
  for i=0 to $k - 1$
    if $elementID \geq ActiveCBFID$
      break
end for
if $elementID > maxIDandActiveCBF$ is full
  $ActiveCBF \leftarrow CreateCBF$
  $l++$
  $ActiveCBFID \leftarrow maxID + 1$
  $ActiveCBF.count \leftarrow 0$
if $elementID < maxID$
for j=0 to $k - 1$
  $ACtiveCBF\,[hash_i\,(IP, F)] \leftarrow ActiveCBF.array\,[hash_i\,(IP, F)] + 1$
  if 4bit counter is overflow
    $ACtiveCBF\,[hash_i\,(IP, F)] \leftarrow 0x1111$
    return $false$
end for
$ActiveCBF.count++$
return $false$

Device Fingerprint Deletion. With the dynamic changes of the scan, the IP address may apply for not using the IP address or change the IP address. Therefore, the bloom filter needs to be deleted. If the filter is not selected correctly when deleting, the counter will be decreased by 1 by mistake. Caused by accidental deletion, so reliable deletion of elements is described in the device fingerprint deletion algorithm.

When performing the delete operation, we need to correctly find the filter that holds the binary group $<IP, F>$, so we search forward from the latest CBF built, and when we find the first CBF whose ID is smaller than the identity ID, we can confirm that the element is stored in this CBF, and confirm that this CBF is an active filter. Delete the tuple $<IP, F>$. Decrease all counters at the corresponding hash position of the element by 1 to complete the deletion

operation. The pseudo code of the deletion algorithm is shown in Algorithm 3. The basic operation of the bloom filter is introduced. Compared with the

Algorithm 3. Device fingerprint deletion algorithm

Input:
$< IP, F >, elementID$
Output:
$Result$
for $j = l$
  $ActiveCBF \leftarrow CBF_j$
  if $elementID \geq ActiveCBFID$
    for i=0 to $k-1$
      $ACtiveCBF\left[hash_i\left(IP, F\right)\right] \leftarrow ActiveCBF.array\left[hash_i\left(IP, F\right)\right] - 1$
      if 4bit counter is decrementing zero
        $ACtiveCBF\left[hash_i\left(IP, F\right)\right] \leftarrow 0x0000$
        return $false$
    end for
    $ActiveCBF.count - -$
    return $true$
end for
return $false$

basic bloom filter, the improved bloom filter in this paper can realize dynamic additions and deletions, increase scalability, and reduce the rate of false positive.

4.3 False Positive Analysis

Suppose S is the total number of IP addresses in the test data set, R is the number of elements judged wrongly, F is the false positive rate, the number of Hash functions is h, and the bit space size is j, then the calculation formula of the false positive rate is as follows:

$$f = \frac{R}{S} \tag{1}$$

When the IP address of the first device is mapped through hash, the formula for calculating the probability that a bit in the bit space j of I is still 0 is as follows:

$$1 - \frac{1}{j} \tag{2}$$

In this paper, hash is used to split the bit space into group I. Therefore, the formula for calculating the probability that a bit in the bit space j of I is still 0 is as follows:

$$1 - \frac{1}{Ij} \tag{3}$$

Suppose there are n IP addresses to be stored, and each element needs to be hashed h times. When all n IP addresses are hashed, the probability that a bit is still 0 is

$$p = (1 - \frac{1}{Ij})^{hn} = (1 - \frac{1}{Ij})^{Ijhn/Ij} \tag{4}$$

According to the formula of Napierian e:

$$\lim_{x \to \infty} (1 - \frac{1}{x})^{-x} = e \tag{5}$$

get

$$p \approx e^{-hn/Ij} \tag{6}$$

When a new IP does not originally belong to the device fingerprint database, but the corresponding Hash value of the IP is no longer 0, it will be considered to be in the set, and false positive will occur. The calculation formula of false positive rate is as follows:

$$f = (1 - p)^k \approx (1 - e^{-hn/Ij})^k \tag{7}$$

For IP address and device fingerprint in this paper, the calculation formula of false positive rate is as follows:

$$f = (1 - p)^k \approx (1 - e^{-hn/Ij})^{2k} \tag{8}$$

As can be seen from the above formula, compared with the basic filter algorithm, the improved device fingerprint matching algorithm can effectively reduce the false positive rate by splitting and double features through the hash function.

5 Performance Evaluation

This section analyzes the false positive rate of the proposed efficient device fingerprint matching algorithm through real experiments. Fingerprint collection requires scanning Internet of things devices in cyberspace. For ethical considerations, we used the Censys dataset [20] as the experimental device fingerprint.

It can be seen from formula 8 that the three parameters that affect the false positive rate are: the number of hash functions, the size of the bit space, and the size of the device fingerprint library. First of all, this paper tests the influence of the number of hash functions and the size of the bit space on the false positive rate, and then comprehensively considers the accuracy of the query and the storage cost, selects the appropriate number of hash functions and the size of the bit space, and finally compares it with the basic layout. Long filter algorithm for experimental comparison.

This experiment randomly selected $1,117,241$ device fingerprints extracted and stored in the bloom filter to construct a device fingerprint library, and also selected $10,000$ data that were not in the device fingerprint library as the test data. Whether the statistics will cause false positive, and calculated the false positive rate.

To determine the influence of the number of hash functions and the size of the bit space on the false positive rate, the experiment tested the number of hash functions as 2, 4, 6, and 8, respectively. As the bit space continues to increase, the false positive rate changes as shown in Figs. 7 and Figs. 8.

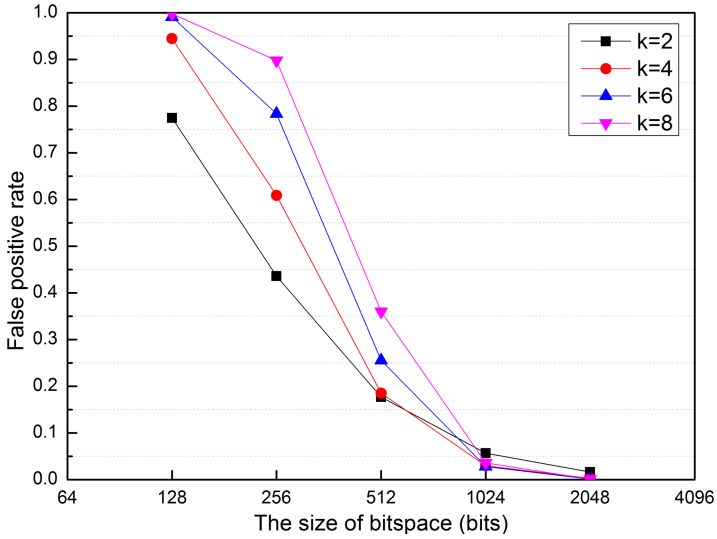

Fig. 7. The relationship between bit space and false positive rate

As shown in Fig. 7, when the number of Hash functions and the size of the device fingerprint library are determined, as the bit space increases, the false positive rate shows a downward trend. This shows that as the bit space increases, the Hash in the bit space. The probability of conflict will be greatly reduced, and the false positive rate will also decrease. As can be seen in Fig. 7, there are four Hash functions. When the bit space is 128 KB, the false positive rate is 0.775, and the bit space. When the storage space is doubled, the false positive rate becomes 0.4359. It can also be seen that the method proposed reduces the false positive rate by sacrificing a certain amount of storage space. It can be observed from Fig. 7 that when the bit space is within the interval of 1024 KB to 2048 KB, although the false positive rate is also reduced, it can also be seen from the figure that the false positive rate curve at this time has become flat, which shows the increase of the bit space. A large bit space can indeed effectively reduce the false positive rate, but too large bit space will also bring about the problem of space waste, so it is necessary to choose a suitable size bit space.

As shown in Fig. 8, when the bit space size and the device fingerprint library size are determined, as the number of Hash functions increases, the increase and decrease of the false positive rate does not show a fixed pattern. When the bit space is 256 KB, as the Hash function increases and decreases. As the number

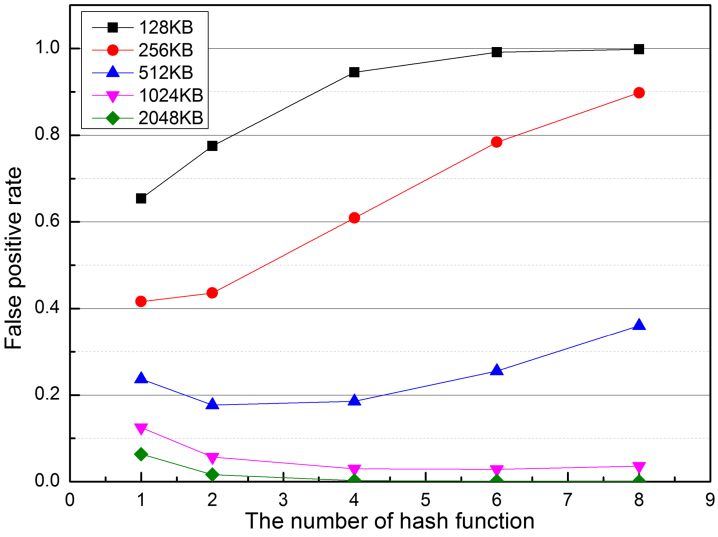

Fig. 8. The relationship between Hash function and false positive rate

of functions increases, the false positive rate increases. This is because the more Hash functions are mapped to the bit space, the false positive rate will increase. When the bit space is 1024 KB, with the number of Hash functions increase, the false positive rate decreases, so the choice of the number of Hash functions is not as much as possible, but the size of the bit space and the number of Hash functions should be considered comprehensively.

It is necessary to meet the accuracy of the device fingerprint query, but also to realize the compressed storage of the device fingerprint library to save storage overhead. This paper believes that the false positive rate of 0.03 has achieved a relatively satisfactory experimental effect, so this paper selects the number of Hash functions It is 4, and the bit space size is 1024 KB or 1 MB. After determining the number of Hash functions to be used and the size of the bit space, this paper compares the basic Bloom filter algorithm with the algorithm proposed in this paper through the size change of the device fingerprint library. The experimental results are shown in Fig. 9.

As shown in Fig. 9, when the number of Hash functions and the size of the bit space are determined, as the number of device fingerprints increases, the false positive rate is also increasing. After the number of device fingerprints exceeds 616287, the error of the basic filter algorithm The judgment rate has also exceeded 0.03, and the false positive rate of the improved algorithm proposed in this paper has been lower than that of the basic Bloom filter, which is consistent with the theoretical false positive rate analysis, and after the number of device fingerprints stored reaches 1117241, the improved algorithm The false positive rate still does not exceed 0.03. In terms of storage capacity, the original 1117241 device fingerprints occupies 25.23 MB, and the improved bit space size is 1024

KB or 1 MB. Compared with the bit space, the storage space is reduced by 96%, and the false positive rate is not higher than 0.03. When the storage capacity is greatly reduced, the experimental analysis from the two aspects of false positive rate and storage capacity proves that the device fingerprint matching algorithm proposed in this paper is efficient in both matching and storage.

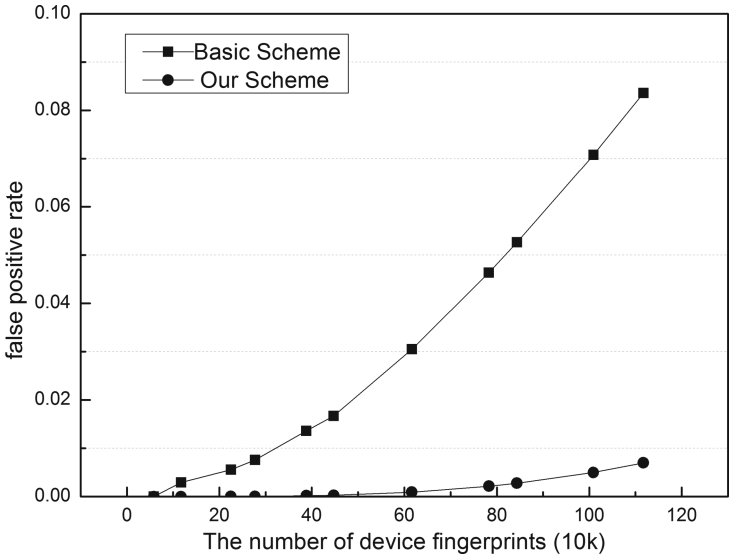

Fig. 9. The relationship between the number of device fingerprints and the false positive rate

6 Conclusion

This paper mainly introduces the IoT security enhancements based on fingerprint device under validation framework, including the architecture, design goal and the authentication service, after for device storage and fingerprint matching in framework, put forward a kind of efficient device fingerprint matching algorithm, introduces in detail the structure of the bloom filter design, the improved algorithm, the basic operation, etc. Finally, the experimental analysis proves that the algorithm has better accuracy and faster verification speed.

Acknowledgment. This work supported by science and technology project funding of State Grid Fujian Electric Power Co., Ltd (52135020002T).

References

1. Li, S., Da Xu, L., Zhao, S.: 5G internet of things: a survey. J. Ind. Inf. Integr. **10**, 1–9 (2018)

2. Luo, Y., Hu, H., Wen, Y., et al.: Transforming device fingerprinting for wireless security via online multitask metric learning. IEEE Internet Things J. **7**(1), 208–219 (2019)
3. Van Oorschot, P.C., Smith, S.W.: The internet of things: security challenges. IEEE Secur. Priv. **17**(5), 7–9 (2019)
4. Park, S.Y., Lim, S., Jeong, D., et al.: PUFSec: device fingerprint-based security architecture for internet of things. In: IEEE Conference on Computer Communications, pp. 1–9. IEEE (2017)
5. Liu, F., Shen, C., Liu, H., et al.: A flexible touch-based fingerprint acquisition device and a benchmark database using optical coherence tomography. IEEE Trans. Instrum. Meas. **69**(9), 6518–6529 (2020)
6. Ren, R.L., Gu, Y., Cui, J., et al.: Web features-based recognition specific-type IoT device in cyberspace. Commun. Technol. **50**(5), 1003–1009 (2017)
7. Oh, J., Yu, S., Lee, J., Son, S., Kim, M., Park, Y.: A secure and lightweight authentication protocol for IoT-based smart homes. Sensors **21**, 1–24 (2021)
8. Jiang, X., et al.: Enhancing IoT security via cancelable HD-sEMG-based biometric authentication password, encoded by gesture. IEEE Internet Things J. **8**(22), 16535–16547 (2021)
9. Azrour, M., Mabrouki, J., Guezzaz, A., Farhaoui, Y.: New enhanced authentication protocol for Internet of Things. Big Data Mini. Anal. **4**(1), 1–9 (2021)
10. Kwon, T., Na, S.: TinyLock: affordable defense against smudge attacks on smartphone pattern lock systems. Comput. Secur. **42**, 137–150 (2014)
11. Aura, T.: Cryptographically generated addresses (CGA). In: Boyd, C., Mao, W. (eds.) ISC 2003. LNCS, vol. 2851, pp. 29–43. Springer, Heidelberg (2003). https://doi.org/10.1007/10958513_3
12. Tan, P., Jia, H., Chen, Y., et al.: A hierarchical source address validation technique based on cryptographically generated address. In: IEEE International Conference on Computer Science and Automation Engineering, pp. 33–37 (2011)
13. Bonneau, J., Herley, C., Stajano, F.M.: Passwords and the evolution of imperfect authentication. Commun. ACM **58**(7), 78–87 (2015)
14. Uluagac, A.S., Radhakrishnan, S.V., Corbett, C., et al.: A passive technique for fingerprinting wireless devices with wired-side observations. In: IEEE Conference on Communications and network Security (CNS), pp. 305–313 (2013)
15. Kumar, K., Dalai, A.K., Panigrahy, S.K., et al.: An ANN based approach for wireless device fingerprinting. In: 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information and Communication Technology (RTEICT), pp. 1302–1307 (2017)
16. Dalai, A.K., Jena, A., Sharma, S., et al.: A fingerprinting technique for identification of wireless devices. In: 2018 International Conference on Computer, Information and Telecommunication Systems (CITS), pp. 1–5. IEEE (2018)
17. Junzhou, L., Ming, Y., Zhen, L., et al.: Cyberspace security system and key technologies. Sci. China: Inf. Sci. **46**(08), 939–968 (2016)
18. Gope, P., Sikdar, B.: Lightweight and privacy-preserving two-factor authentication scheme for IoT devices. IEEE Internet Things J. **6**(1), 580–589 (2018)
19. El-hajj, M., Fadlallah, A., Chamoun, M., Serhrouchni, A.: A survey of internet of things (IoT) authentication schemes. Sensors **19**, 1141 (2019)
20. Censys: A search engine based on Internet-wide scanning for the devices and networks (2015). https://censys.io/