Frontiers of Information Technology & Electronic Engineering
www.zju.edu.cn/jzus; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

A keyed-hashing based self-synchronization mechanism for port address hopping communication*

Yue-bin LUO‡, Bao-sheng WANG, Xiao-feng WANG, Bo-feng ZHANG

(*College of Computer, National University of Defense Technology, Changsha 410073, China*)

E-mail: luoyuebin@nudt.edu.cn; bswang@nudt.edu.cn; xf_wang@nudt.edu.cn; bfzhang@nudt.edu.cn

Received Sept. 16, 2016; Revision accepted Apr. 6, 2017; Crosschecked May 8, 2017

Abstract: Port address hopping (PAH) communication is a powerful network moving target defense (MTD) mechanism. It was inspired by frequency hopping in wireless communications. One of the critical and difficult issues with PAH is synchronization. Existing schemes usually provide hops for each session lasting only a few seconds/minutes, making them easily influenced by network events such as transmission delays, traffic jams, packet dropouts, reordering, and retransmission. To address these problems, in this paper we propose a novel self-synchronization scheme, called 'keyed-hashing based self-synchronization (KHSS)'. The proposed method generates the message authentication code (MAC) based on the hash based MAC (HMAC), which is then further used as the synchronization information for port address encoding and decoding. Providing the PAH communication system with one-packet-one-hopping and invisible message authentication abilities enables both clients and servers to constantly change their identities as well as perform message authentication over unreliable communication mediums without synchronization and authentication information transmissions. Theoretical analysis and simulation and experiment results show that the proposed method is effective in defending against man-in-the-middle (MITM) attacks and network scanning. It significantly outperforms existing schemes in terms of both security and hopping efficiency.

Key words: Synchronization; Port address hopping; Moving target defense; Network security
http://dx.doi.org/10.1631/FITEE.1601548 **CLC number:** TP393.08

1 Introduction

Moving target defense (MTD) (Chong *et al.*, 2009) was proposed as an innovative and proactive defense technique against attackers by continuously shifting the attack surfaces in a system. This approach forces the attackers into increasing their efforts and costs to exploit target systems. Port address hopping (PAH) is a powerful network MTD method inspired by the well-known frequency hopping mechanism used in wireless communications. PAH tries to hide host and service identities (i.e., IP addresses and port numbers) by performing synchronized PAH communication between authorized clients and servers. In this way, PAH spreads the data stream across multiple data connections called 'channels' to resist network reconnaissance and attacks for a communication session. One of the critical issues involved in the design of PAH systems is synchronizing the communication parties. Since a computer network is a widely distributed system with uncertain network transmission delays and traffic jams, it is a considerable challenge for the hopping system to accurately synchronize the communication parties when fulfilling the demands for security and performance.

The existing works provide two main kinds of synchronization mechanisms distinguished in whether a public third-party server is used or not.

‡ Corresponding author

* Project supported by the National Basic Research Program (973) of China (No. 2012CB315906) and the National Natural Science Foundation of China (No. 61303264)

ORCID: Yue-bin LUO, http://orcid.org/0000-0002-8194-5262

The first category called 'time synchronization' or 'timestamp-based synchronization' relies on a public third-party server (e.g., a time or timestamp server) to synchronize the communication parties. Kewley *et al.* (2001) provided a technique called 'dynamic network address translation (DyNAT)', which enables dynamic network address translation of the Internet protocol (IP) address and transmission control protocol (TCP) port number. This technique makes it difficult for the adversary to create and maintain a map of the network. Lee and Thing (2004) proposed random port hopping (RPH), which changes the server's port numbers dynamically as a function of time and a cryptographic key shared between the server and clients. Similar tactics such as network address hopping/mutation (Sifalakis *et al.*, 2005; Jafarian *et al.*, 2014) and port and address hopping schemes (Atighetchi *et al.*, 2003; Luo *et al.*, 2015b; 2017) have been presented to implement hopping communication relying on time synchronization and random number generators. This kind of synchronization uses the synchronized system time to determine the transmission channels (i.e., their addresses and ports) to be selected periodically over time. The potential problem in this case is that a precise clock is required for both the server and the client; thus, it cannot be used for communications by connecting multiple hosts with different local times. Moreover, the network congestion and delay can cause synchronization errors and further lead to protection failures. Shi *et al.* (2008) introduced a full service hopping (FSH) tactic and designed a timestamp-based synchronization scheme to overcome the flaws of time synchronization. Before launching the hopping service, the current timestamp of the server host is delivered to a public server (or agent), and all of the trusted clients must request the timestamp from the public server so that they can determine the right service communication identities. However, timestamp-based synchronization has two critical flaws, i.e., request overload and boundary failure. Furthermore, a common problem with this kind of synchronization is that instead of the application server, the synchronization server may become the weak point.

The other category of synchronization mechanism is based on the transmitting synchronization information along with the communication data. Since this type of mechanism does not rely on a third party, we call it 'self-synchronization'. Badishi *et al.* (2007) presented a port rationing channel (PRC) that applies to two-party communications, and introduced an ACK-based method to overcome the problem of synchronization. ACK-based synchronization requires no precise clock, and the hopping information is transferred through an ACK frame. The drawback is that it cannot be applied to multi-party communications since a third party cannot join in the communication without the ACK frame. In addition, acknowledgement losses may lead to the two parties using a certain port for a longer period of time. Morris *et al.* (2012) proposed to use data length (DL) or packet count (PC) for hopping synchronization. This approach switches the ports based on DL or PC of a given transaction during the communication session. DL or PC randomly changes during the communication session and varies from one transaction to another. However, both the server and the client must precisely count DL or the packet number during the whole session, and any network event such as data packet dropout, packet reordering, or retransmission may cause synchronization failures.

Obviously, none of the previous techniques provide a perfect synchronization mechanism without using third-party servers, when adapting to complex network situations including transmission delays, traffic jams, packet dropouts, reordering, and retransmission. In this paper we present a self-synchronization scheme called 'keyed-hashing based self-synchronization (KHSS)', which overcomes the previous problems and implements synchronization based on the HMAC (Krawczyk *et al.*, 1997) mechanism for message authentication (Modares *et al.*, 2014). In KHSS, a cryptographic hash function, e.g., MD5 (Rivest, 1992) or SHA-1 (Eastlake and Jones, 2001), is used in combination with a secret shared key, where a message is hashed to a digest, which is further used as the synchronization information for PAH.

Our method possesses a one-packet-one-hopping synchronization ability and enables both the clients and servers to change their communication ports and IP addresses without relying on third-party servers or using additional communication channels to transmit synchronization information. The KHSS-based PAH system also inherits the message authentication function from HMAC, providing a secret and convenient way to check the integrity of information transmitted over an unreliable medium. In contrast

to traditional message authentication which transfers the message authentication code (MAC) with the communication data, the KHSS-based PAH scheme does not need to transmit any authentication information through the communication channel, and the keyed-hashing operations and port address encoding/decoding are performed only in the hopping converters. Therefore, KHSS-based PAH provides an invisible and more secure message authentication method, which is effective for defending against man-in-the-middle (MITM) attacks. We provide security analysis and conduct experiments on our campus network and a virtual network to evaluate the performance and overhead of the proposed method, and the results show that our method significantly outperforms conventional synchronization schemes in terms of both security and hopping efficiency while maintaining an acceptable overhead.

2 Keyed-hashing based self-synchronization scheme

2.1 System model and problem formulation

The general PAH model is shown in Fig. 1, where there is a PAH converter located at the edges of the client network and the server network (e.g., on the gateway in front of the subnet switch for the client or server domain), which performs port and address mutation/translation according to the port and address encoding/decoding algorithm. It contains four modules: (1) the synchronization module which synchronizes the communication parties using synchronization schemes described in Section 1, (2) the preprocessing module which unpacks the received packet and extracts its real communication identities (i.e., IP addresses and port numbers), (3) the port address mutation module which encodes/decodes the real identities to/from the temporary virtual identities based on the synchronization information and pseudo-random generators, and (4) the repacking module which repacks the transformed packet.

The PAH system relies on the synchronization module to synchronize the communication parties, and the service is available only if PAH is performed based on precise synchronization information. Therefore, there is a general consensus that the primary focus of the synchronization technologies must be: (1) to increase the hopping frequency

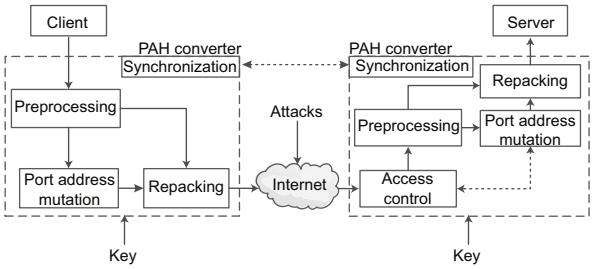

Fig. 1 General port address hopping system model (PAH: port and address hopping)

while reducing the rate of synchronization failures, (2) to develop synchronization methods that depend neither on the third-party servers nor on additional communication channels, and (3) to adapt to complex and constantly changing network environments.

Generally, a PAH system uses subnet addresses other than a global IP address for address hopping to avoid potential routing problems because: (1) it would explode the sizes of the routing tables, the number of routing updates, and the frequency of recomputing routes; (2) it would result in tremendous administrative overhead for reconfiguring mechanisms that make address-based decisions; (3) it would require global coordination to be implemented. Therefore, providing subnet-level address hopping is an appropriate choice and the packets after address hopping are still routable without routing updates. Subnet-level address hopping means that the only address translated is the host portion of the destination address, not the network address. Thus, the packet can be routed normally. The number of bits that are translated depends on the class of IP address in use. Specifically, the addresses of classes B and C are used. Thus, address hopping mutates the lower 16 bits in each class B destination address and the lower 8 bits in class C. In this study we describe the address mutation limited to subnet-level addresses.

When it comes to a next-generation network such as a software-defined network (SDN) (Lantz *et al.*, 2010), the route problem using a global IP can be solved through flow table distributions. Under this condition, address hopping will no longer be limited to the subnet level. Note that address hopping can use a global IP in address hopping, meaning that the arbitrary k-bit part of the 32-bit public IP address can be changed in the hopping; thus, we must guarantee that the addresses after hopping will not fall into the private or reserved address ranges.

KHSS provides a universal synchronization mechanism, where this kind of address hopping using a global IP can be easily implemented by performing address transitions for a larger number of bits in the address encoding/decoding phase.

2.2 Keyed-hashing based self-synchronization

In this study we introduce the HMAC message authentication mechanism (Krawczyk *et al.*, 1997) and design a self-synchronization mechanism for PAH, KHSS, where a message is hashed to a digest which is then used for port and address encoding/decoding in PAH. KHSS requires a cryptographic hash function denoted by H (e.g., MD5, SHA-1), in combination with a secret key K. To defend against MITM attacks, key-sharing can be achieved using public key cryptography (PKC) (Forouzan, 2009), which establishes a shared secret key between two parties based on an authentic third party, i.e., a public key infrastructure (PKI). The latest studies show that key-sharing can also be achieved using a secret handshake scheme (Gu and Xue, 2011) without any third party. The scheme establishes a shared session key based on pairings over elliptic curves (EC). Its security relies on the bilinear Diffie–Hellman assumption (BDH). After that, the communication parties can use the session key to perform PAH. For a new communication session, the two parties can establish a new session key or use the existing session key to perform KHSS. We assume that all communication parties already have their session keys based on a secret handshake scheme.

The general structure of the KHSS scheme describes how KHSS uses the HMAC message authentication mechanism to allow a PAH converter to synchronously compute variant communication identities (Fig. 2). Every time a PAH converter receives a message, the message is used as the input for HMAC, and HMAC generates an L-bit hash sum (L=128 for MD5, L=160 for SHA-1) as its output. Then, the hash sum is used to perform exclusive OR operations with the communication identities that are extracted from the message and determine the communication identities to which they should be changed.

Fig. 3 describes the detailed operations for KHSS, which can be broken down into three phases: a preprocessing phase, a keyed-hashing phase, and an identity encoding/decoding phase. In the preprocessing phase, the communication identities including the destination IP address (iph→daddr), source IP address (iph→saddr), destination port (tcph→dest), and source port (tcph→source) for the received packet are extracted and stored. Then the 17-byte variant fields of the IP datagram including the source and destination IP addresses and ports, the time to live (iph→ttl), the network layer checksum (iph→check), and the communication layer checksum (tcph→check) are set to 0 to form a transmission invariant message M.

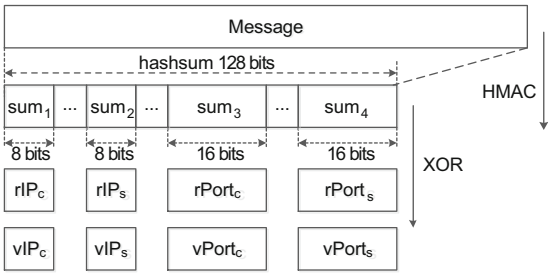

Fig. 2 General structure of the keyed-hashing based self-synchronization scheme (HMAC: hash based MAC)

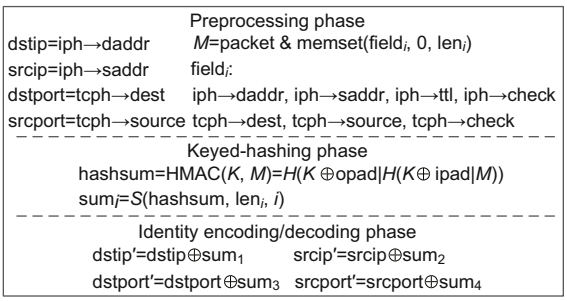

Fig. 3 Keyed-hashing based self-synchronization operations in detail

In the keyed-hashing phase, message M and the shared secret key K are used as the inputs for the HMAC to generate a hash output denoted by hashsum. In the HMAC function, ipad and opad are two fixed and different strings defined in Krawczyk *et al.* (1997). Furthermore, a random selection function S shared between the encoder and the decoder is used to randomly select the fixed-length parts of the hashsum, which we denote by sum_i, and len_i is its bit length, where i represents the index. For PAH in a class C network, two 8-bit parts and two 16-bit parts are randomly selected from the hashsum for address and port encoding/decoding, respectively. In the identity encoding/decoding phase, the hash digests generated and selected in the preceding phase

are used to perform exclusive OR operations with the identities extracted from the message in the preprocessing phase and to determine the communication identities to which they should be changed.

2.3 Benefits of keyed-hashing based self-synchronization based port address hopping

The keyed-hashing based self-synchronization provides PAH with one-packet-one-PAH and an invisible message authentication function. The former provides PAH with a network scanning defense ability. By deploying fast PAH, the real service identities are replaced by temporary virtual identities. This functionality is called 'service hiding', whereby the underlying correlation between the IP address and the server host, and the correlation between the port number and the service application are broken down. As a result, attackers cannot directly identify the server/service information through the IP address or port number in the packet. The identity information that is changed in the hopping consists of (1) the host portion of the server IP address and (2) the TCP/UDP port number. This enables us to conceal, from layers 2 and 3's perspectives, the identity of the actual server machines and services. Such concealment is sufficient to defeat a large pool of network-level traffic analysis tools. Furthermore, one-packet-one-hopping is the maximum hopping frequency that a PAH system can reach, and it is also the safest scheme, which is known as perfect hopping (Luo *et al.*, 2015b), since the lifetime of each hop can be as short as several microseconds. Only the packet whose message digest and communication identities fulfill the KHSS-based PAH mechanism (i.e., its port and address can be decoded and mapped to a port and an address that are valid and active on the receiver side) can reach the right server host and service application. Meanwhile, messages using random/invalid ports and addresses can be easily detected and controlled.

An additional benefit of this tactic is that it increases the likelihood of detecting attackers. Since attackers cannot accurately locate the target service, they are forced to re-scan the target network frequently at different times. These recurring probes significantly slow down the attack progress, and at the same time increase their chances of being detected. The KHSS-based PAH combines invisible message authentication with PAH. In this scenario, although an attacker is capable of monitoring the network traffic and obtaining communication information at any given time, the attacker cannot change its context since any change in the packet will result in an invalid decoding result on the receive side.

The PAH scheme pseudo-randomly spreads a peer-to-peer connection across the port address space. In the pseudo-random PAH scheme the port address values are changed in a manner that is unknown to intermediary attackers but known by the two endpoints or peers. Flow sniffing schemes depend on the stability of flow identification through a 5-tuple that includes source and destination IP addresses, source and destination port addresses, and a protocol type. The peer-to-peer flows that use the PAH scheme are no longer bound to these identifiers. Thus, an intermediary attacker cannot build up the necessary state to manipulate the flow. In addition, the KHSS-based PAH system inherits the message authentication function from HMAC. In contrast to traditional message authentication in which MAC is transferred with the communication data, KHSS-based PAH provides an invisible and more secure message authentication scheme in combination with PAH, in which the message digest is computed and used only in the PAH converters without transmission through any communication channels. Therefore, the attackers cannot perform brute force attacks to find the secret key or hash collisions since they know nothing about the real MAC used in the PAH communication. As an example, a packet with its communication identities encoded is sent to a network, and an attacker in the middle might launch attacks such as modification and masquerading to threaten the integrity of the message. These changes will induce different outputs of HMAC on the receiver side, and further determine illegal/invalid communication identities decoded from the message. Consequently, this illegal message can be easily detected and controlled. Therefore, the proposed method can effectively defeat a large class of MITM attacks.

3 Security analysis

3.1 Description of HMAC

HMAC, which is a standardized hash-based MAC algorithm, was designed by Bellare *et al.* (1996)

and later standardized by ANSI, IETF, ISO, and NIST (Krawczyk *et al.*, 1997). HMAC has also become very popular and has been widely used and implemented in SSL, SSH, IPsec, and TLS among others. HMAC takes a message of an arbitrary bit-length and hashes it to a fixed-length output with a secret key. HMAC applies, in both its inner and outer parts, the iterated message digest construction of the hash function H, which is defined by

$$\mathrm{HMAC}(K, M) = H(K \oplus \mathrm{opad} \mid H(K \oplus \mathrm{ipad} \mid M)),$$

where M is the message, K is the secret key, and opad and ipad are two fixed constants. In practice, the function H can be replaced by the compression function of cryptographic hash functions such as MD4, MD5, SHA-0, and SHA-1.

3.2 Security for keyed-hashing based self-synchronization mechanism

The HMAC algorithms usually inherit the security and efficiency of their underlying primitives. HMAC has been proven to be secure as long as the compression function of the underlying hash function is a pseudo-random function. Under this condition, HMAC is proved to be pseudo-random. Note that the security proof of pseudo-randomness provides the KHSS security. It has been proven that HMAC-MD5 and HMAC-SHA-1 can provide greater security than their underlying hash function MD5 and SHA-1. Furthermore, the attackers know nothing about the message digest that is generated by HMAC in KHSS and used for port address encoding/decoding, since the message digest is computed and used only in the PAH converters without transmission through any communication channels. Therefore, it would be much harder for an attacker to attack the KHSS mechanism than to attack the original MAC mechanism.

The security of the KHSS mechanism presented here depends on the cryptographic properties of the underlying keyed hash function HMAC, whose underlying cryptographic hash function, e.g., MD5 or SHA-1, satisfies the basic hash function criteria: preimage resistance, second preimage resistance, and collision resistance. The security of HMAC was discussed and proved by Krawczyk *et al.* (1997). Apparently, different stages and application situations present different security requirements, which can be achieved through using hash functions at different security levels, such as SHA-256, SHA-384, and SHA-512.

3.3 Security of keyed-hashing based self-synchronization based port address hopping

Assume that an attacker knows the KHSS-based PAH scheme, without the shared secret key wanting to launch attacks on the host. In this scenario, the attacker tries to access the service and constructs a message that simultaneously fulfills the KHSS-based PAH and the message authentication mechanisms. There are two kinds of attacks the attacker can perform: (1) The attacker monitors the network link and obtains a packet with valid communication identities, tries to use the packet's identities, and constructs a new packet whose message digest fulfills the PAH mechanism to connect the target host. However, this cannot succeed since it disobeys the preimage resistance of the hash criteria. (2) The attacker tries to construct a packet using a random port and an address, and hopes that the packet's message digest and identities fulfill the KHSS-based PAH mechanism.

Let n be the byte length of an IP datagram. Since an Ethernet frame contains at least 46 and up to 1500 bytes of data, we have $46 \le n \le 1500$. Hence, the transmission invariant message M used in KHSS contains $m = 8(n-17)$ bits of information ($232 \le m \le 11\,864$), and M is hashed to an L-bit digest by HMAC in KHSS. Considering the second kind of attack, the attacker constructs an n-byte message and randomly generates its L-bit hash digest D. The message and digest D will fulfill the HMAC function with probability

$$P_1 = \frac{2^L}{2^{8(n-17)}}. \tag{1}$$

Furthermore, the attacker randomly selects two 8-bit parts and two 16-bit parts from digest D to encode the original source and the destination port and the address of the constructed message. Given an L-bit message digest D, there are $\binom{L-7}{1}$ and $\binom{L-15}{1}$ probabilities for the random selection function S to select an 8-bit part and a 16-bit part from the L-bit digest D, respectively. Therefore, the client and server address hopping has a $\binom{L-7}{1}^2$ probability and the port hopping has a $\binom{L-15}{1}^2$ probability. This means that the success probability denoted by P_2 for the attacker to encode the communication identities

to the right identities becomes

$$P_2 = \frac{1}{\binom{L-7}{1}^2 \binom{L-15}{1}^2}. \quad (2)$$

Consequently, the attacker constructs a message that fulfills the second kind of attack, and the message can arrive at the target service with a success probability denoted by P_3, which is

$$P_3 = P_1 \cdot P_2 = \frac{2^L}{2^{8(n-17)}} \cdot \frac{1}{(L-7)^2(L-15)^2}. \quad (3)$$

Let x denote the number of malicious packets that fulfill the second kind of attack. The average value of x, $E(x)$, is given by

$$E(x) = \frac{mrt}{2^{8n-L-136}(L-7)^2(L-15)^2}, \quad (4)$$

where m is the number of attackers, r is the rate of packets generated by the attackers, and t refers to the duration of the attack.

For example, assuming that the attacker can send packets close to the theoretical limit of an Ethernet card, reaching approximately 1.5×10^6 packets/s using a 1 Gb/s network card (Karlin and Peterson, 2002), $m = 10$, $t = 3600$ s, and $L = 128$, we have $E(x) = 1.42 \times 10^{-29}$. Obviously, the attack success rate is too low to succeed.

Meanwhile, the validity of the constructed message cannot be authenticated until the message arrives at the PAH converter on the server side, where every connection attempt disobeying the PAH mechanism will induce an anomaly record, and the administrator can take action (e.g., deny/discard, redirect to honey net, or provide fake service) to cope with the attacks. Therefore, the practical attack success rate will be far below the probability depicted in Eq. (4).

Furthermore, the implementation, the choice of random (or cryptographically pseudo-random) keys, a secure key exchange mechanism, frequent key refreshments, and good secrecy protection of keys are all essential ingredients for the security of the synchronization and integrity authentication mechanisms provided by KHSS. These questions are outside the domain of this study, and we can find appropriate solutions in the conventional information security areas.

4 Performance evaluation

4.1 Simulation tests

In this section, we evaluate the effectiveness of the KHSS-based PAH in resisting network reconnaissance and attacks through theoretical analysis and simulation.

Table 1 shows the hopping period for KHSS-based PAH versus existing port and/or address hopping schemes including RPH (Lee and Thing, 2004), PRC (Badishi *et al.*, 2007), DL/PC-based PH (Morris *et al.*, 2012), network address space randomization (NASR) (Antonatos *et al.*, 2007), and random port and address hopping (RPAH) (Luo *et al.*, 2015b). Compared to the existing schemes, our KHSS-based PAH system provides the maximum hopping frequency. Because the minimum amount of time from the end of a packet to the end of the next packet is 6.72 μs for a 100 Mb/s full-duplex Ethernet link (Karlin and Peterson, 2002), the hopping frequency for KHSS-based PAH in this scenario is as high as 1.49e5.

Table 1 Comparison of the hopping times

Scheme	Hopping period (100 Mb/s Ethernet)
RPH	UDP: 0.5 s; TCP: 12–19 s
PRC	At least one round-trip time (168.9 ms)
DL/PC-based PH	1–4 packets time (6.72–26.88 μs)
NASR	15 min
RPAH	5 s
KHSS-based PAH	One packet time (6.72 μs)

PH: port hopping; RPH: random PH; PRC: port rationing channel; DL: data length; PC: packet count; NASR: network address space randomization; PAH: port and address hopping; RPAH: random PAH; KHSS: keyed-hashing based self-synchronization

From the previous section, we know that the probability of an attacker constructing a random message that fulfills the synchronization and hopping rules is P_3, as depicted in Eq. (3). For a successful network reconnaissance or attack, the attacker would connect to the target host or service multiple times. Generally, some security systems would be deployed to protect the server host; therefore, the number of scans (N) the attacker can perform to reach the server host is limited. In this scenario, for N scans, the scanner will discover the target server host with probability

$$P = 1 - (1 - P_3)^N. \quad (5)$$

Let us consider the scenario of using MD5 (i.e.,

$L = 128$). The maximum value of P_3 occurs when $n = 46$, and $P_3 \approx 2.64 \times 10^{-40}$. Practically, P_3 is far below 2.64×10^{-40} because the average length of the IP packet in the network is 557 bytes.

In this section, to make comparisons between our KHSS-based PAH and existing hopping techniques as well as the static legacy networks, we conducted simulations to evaluate their effectiveness against network reconnaissance, where an attacker with the capacities stated above in a class C network was simulated to perform host discovery and port scanning to a server host in another class C network, and the results are shown in Figs. 4a and 4b, respectively. Fig. 4a shows the scan success rate for the attacker in performing host discovery to the target network deployed with KHSS-based address hopping compared to the existing address hopping/mutation mechanisms including static, NASR (Antonatos *et al.*, 2007), STAM (Jafarian *et al.*, 2014), and address hopping in RPAH (Luo *et al.*, 2015b). Fig. 4b shows the scan success rate for the attacker in performing port scanning to a target host compared to the existing port hopping/mutation mechanisms including static, RPH (Lee and Thing, 2004), PRC (Badishi *et al.*, 2007), DR/PC-based port hopping (Morris *et al.*, 2012), and port hopping in RPAH (Luo *et al.*, 2015b). The results show that our KHSS-based address hopping provides the best defensive performance for the same number of scans compared to the existing address hopping schemes. KHSS-based port hopping also provides a better defensive capability compared to the existing port hopping schemes, even though the number of scans is up to 200 000. Most often, port scanning is performed as a subsequent reconnaissance phase after host discovery. It is used to find vulnerable services in a target host discovered in the host discovery. The total scan success rate of the whole network reconnaissance equals the scan success rate for host discovery multiplied by the scan success rate for port scanning. Therefore, the KHSS-based PAH mechanism can provide a more effective and active defense method for the host and service hiding compared to the existing schemes.

4.2 Real network tests

The proposed mechanism has flexible implementation models. For the purpose of defending against internal attacks and scanning, the PAH on the server side can be divided into the port hopping located on the server host directly and the address hopping located on the gateway in front of the subnet switches. To be transparent to the client, PAH on the client side is located on the gateway in front of the subnet switches. In this scenario, the gateways of the client and server subnets and the server host should keep their synchronization. For simplified implementation, PAH on both the client and server sides can be located on the gateway in front of the subnet switches. In this scenario, only the gateways need to be synchronized other than the client–server synchronization. The drawback of this implementation model is that both communication ports and IP addresses in the subnet network are real and static, making the server hosts suffer from internal attacks and scanning.

To study and demonstrate the feasibility of our approach, we have conducted experiments on two designated class C subnets within our university campus network to test the effectiveness and evaluate the synchronization performance of our KHSS

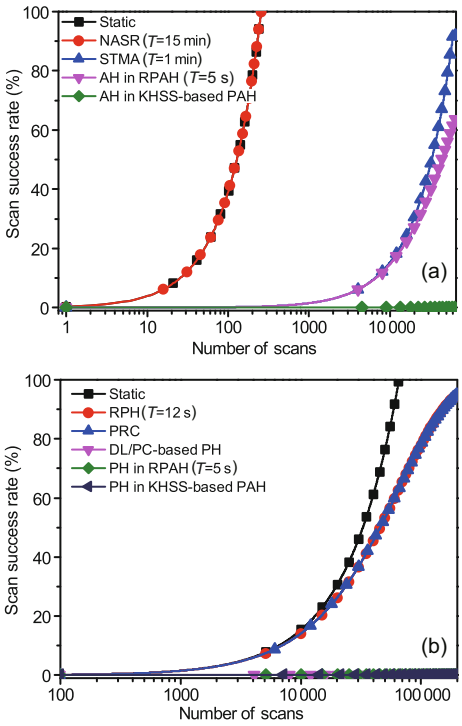

Fig. 4 Scan success rate of different address hopping (AH) schemes (a) and different port hopping (PH) schemes (b) vs. the number of scans (NASR: network address space randomization; PAH: port and AH; RPAH: random PAH; KHSS: keyed-hashing based self-synchronization; RPH: random PH; PRC: port rationing channel; DL: data length; PC: packet count)

scheme based on the latter implementation model. First, we implemented a KHSS-based PAH system in C and deployed it on Ubuntu Linux (kernel 3.17.3) using iptables (version 1.4.21) and the netfilter framework (http://www.netfilter.org). Further, we performed address and port scanning 100 times using Nmap (https://nmap.org), and none of the server hosts or services in the target network were scanned out. The scanning test shows that our KHSS-based PAH is effective in resisting network scanners. Third, to evaluate the influence of services introduced by the proposed method, we performed various network activities during PAH including web browsing and downloading files. These network connections were not affected or interrupted and they even persisted for more than one hour. We also used ApacheBench (http://httpd.apache.org) to perform concurrent connection tests. Specifically, we used ApacheBench to perform 50 tests of 1000 web requests with 10 concurrences, and the connection success rates were 100% for all the tests. Our service tests simply prove that the proposed method is an effective PAH synchronization mechanism.

To measure the overhead in the network communication introduced by the proposed method, we conducted experiments to compare and evaluate the proposed method with existing schemes including RPAH (Luo *et al.*, 2015b) and TAP-based PAH (TPAH) (Luo *et al.*, 2015a). In the experiment, two distinct hosts in the two class C networks were selected to form a client-to-server connection. Then, the throughput for communications with and without the KHSS-based PAH was measured, and the throughput data was obtained using the speed test tool Iperf (https://iperf.fr) 50 times for each test. The results show the impact of introducing the KHSS-based PAH mechanism on the throughput (average overhead was about 0.05% for using MD5 and about 0.18% for using SHA1) (Fig. 5a). Note that the bandwidth of our campus experimental network was 100 Mb/s. To test their overheads at a higher network bandwidth, we also constructed a virtual network whose average throughput was as high as 1740.33 Mb/s using the VMware platform with two Ubuntu virtual machines to perform the same throughput test, and the results are depicted in Fig. 5b. The test results for a higher bandwidth virtual network show the impact of introducing the KHSS-based PAH mechanism on the throughput (average overhead was about 41.52% for using MD5 and about 48.20% for using SHA1). Although when introducing our KHSS-based PAH scheme in the high bandwidth virtual network, the overhead seemed very high, the throughput can also reach 1 Gb/s when the MD5 hash algorithm was used in KHSS. This synchronization performance and the communication throughput are good enough for the vast majority of application scenarios as well as network conditions. Therefore, the real network test data proves that our KHSS-based PAH mechanism outperforms existing hopping schemes and provides sufficient communication and synchronization performance.

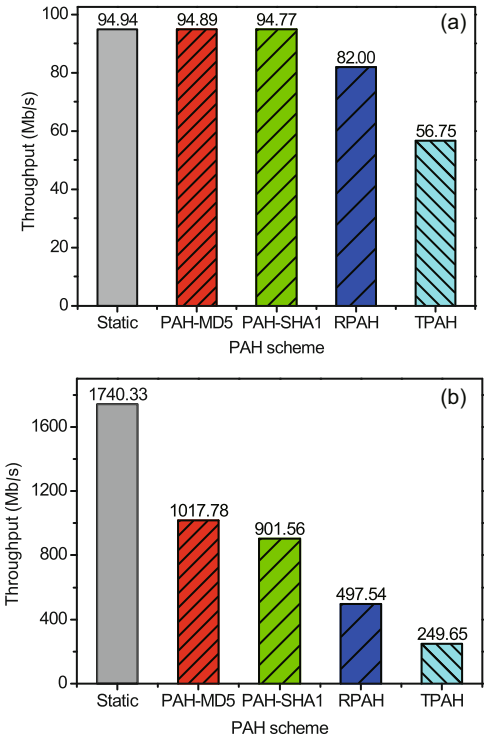

Fig. 5 Results of the throughput tests on our campus network (a) and on a virtual network (b) (PAH: port address hopping; RPAH: random PAH; TPAH: TAP-based PAH)

5 Conclusions

We have proposed a self-synchronization mechanism called KHSS, where the keyed hash function, the hash based MAC (HMAC), creates a digest of the message that passes through the PAH converter and its communication identities are encoded/decoded based on the generated message digest. The

proposed scheme enables one-packet-one-PAH and invisible message authentication schemes without synchronization information and message authentication code transmissions. Meanwhile, KHSS adapts to complex network environments and overcomes inevitable transmission problems including transmission delays, traffic jams, packet dropouts, reordering, and retransmission. The simulation and experiment analyses of the KHSS performance and the security were made, and the results showed that the proposed scheme presents significant advantages over existing synchronization schemes in terms of both security and hopping efficiency. Our evaluation also proved that the overhead when introducing the KHSS mechanism in a traditional network is quite low (below 0.2%) and that the proposed method is good enough to be deployed in most of application scenarios in a high bandwidth network.

References

Antonatos, S., Akritidis, P., Markatos, E.P., *et al.*, 2007. Defending against hitlist worms using network address space randomization. *Comput. Netw.*, **51**(12):3471-3490. http://dx.doi.org/10.1016/j.comnet.2007.02.006

Atighetchi, M., Pal, P., Webber, F., *et al.*, 2003. Adaptive use of network-centric mechanisms in cyber-defense. Proc. 6th IEEE Int. Symp. on Object-Oriented Real-Time Distributed Computing, p.183-192. http://dx.doi.org/10.1109/ISORC.2003.1199253

Badishi, G., Herzberg, A., Keidar, I., 2007. Keeping denial of service attackers in the dark. *IEEE Trans. Depend. Sec. Comput.*, **4**(3):191-204. http://dx.doi.org/10.1109/TDSC.2007.70209

Bellare, M., Canetti, R., Krawczyk, H., 1996. Keyed hash functions for message authentication. *LNCS*, **1109**:1-15. http://dx.doi.org/10.1007/3-540-68697-5_1

Chong, F., Lee, R.B., Acquisti, A., *et al.*, 2009. National Cyber Leap Year Summit 2009 Co-chairs Report. NITRD Program.

Eastlake, D.III, Jones, P., 2001. US Secure Hash Algorithm 1 (SHA1). Internet Society, Washington DC, USA. http://dx.doi.org/10.17487/RFC3174

Forouzan, B.A., 2009. Cryptography & Network Security. McGraw-Hill, Inc., New York, USA.

Gu, J., Xue, Z., 2011. An improved efficient secret handshakes scheme with unlinkability. *IEEE Commun. Lett.*, **15**(2):259-261. http://dx.doi.org/10.1109/LCOMM.2011.122810.102229

Jafarian, J.H., Al-Shaer, E., Duan, Q., 2014. Spatio-temporal address mutation for proactive cyber agility against sophisticated attackers. Proc. MTD Workshop at CCS, p.69-78. http://dx.doi.org/10.1145/2663474.2663483

Karlin, S., Peterson, L., 2002. Maximum Packet Rates for Full-Duplex Ethernet. Technical Report TR-645-02, Department of Computer Science, Princeton University, Princeton, USA.

Kewley, D., Fink, R., Lowry, J., *et al.*, 2001. Dynamic approach to thwart adversary intelligence gathering. Proc. DARPA Information Survivability Conf. and Exposition, p.176-185. http://dx.doi.org/10.1109/DISCEX.2001.932214

Krawczyk, H., Bellare, M., Canetti, R., 1997. HMAC: Keyed-Hashing for Message Authentication. IETF Internet Request for Comments 2104 (RFC-2104).

Lantz, B., Heller, B., McKeown, N., 2010. A network in a laptop: rapid prototyping for software-defined networks. Proc. 9th ACM SIGCOMM Workshop on Hot Topics in Networks, p.19:1-19:6. http://dx.doi.org/10.1145/1868447.1868466

Lee, H.C.J., Thing, V.L.L., 2004. Port hopping for resilient networks. Proc. IEEE 60th Vehicular Technology Conf., p.3291-3295. http://dx.doi.org/10.1109/VETECF.2004.1404672

Luo, Y.B., Wang, B.S., Wang, X.F., *et al.*, 2015a. TPAH: a universal and multi-platform deployable port and address hopping mechanism. Proc. Int. Conf. on Information and Communications Technologies, p.214-219. http://dx.doi.org/10.1049/cp.2015.0230

Luo, Y.B., Wang, B.S., Wang, X.F., *et al.*, 2015b. RPAH: random port and address hopping for thwarting internal and external adversaries. Proc. 14th IEEE Int. Conf. on Trust, Security and Privacy in Computing and Communications, p.263-270. http://dx.doi.org/10.1109/Trustcom.2015.383

Luo, Y.B., Wang, B.S., Wang, X.F., *et al.*, 2017. RPAH: a moving target network defense mechanism naturally resists reconnaissances and attacks. *IEICE Trans Inform. Syst.*, **E100-D**(3):496-510. http://dx.doi.org/10.1587/transinf.2016EDP7304

Modares, H., Moravejosharieh, A., Lloret, J., *et al.*, 2014. A survey of secure protocols in Mobile IPv6. *J. Netw. Comput. Appl.*, **39**:351-368. http://dx.doi.org/10.1016/j.jnca.2013.07.013

Morris, C.C., Burch, L.L., Robinson, D.T., 2012. Techniques for Port Hopping. US Patent 8 301 789.

Rivest, R.L., 1992. The MD5 Message Digest Algorithm. Internet Engineering Task Force, Fremont, USA.

Shi, L.Y., Jia, C.F., Lü, S.W., 2008. Full service hopping for proactive cyber-defense. Proc. IEEE Int. Conf. on Networking, Networking, Sensing and Control, p.1337-1342. http://dx.doi.org/10.1109/ICNSC.2008.4525425

Sifalakis, M., Schmid, S., Hutchison, D., 2005. Network address hopping: a mechanism to enhance data protection for packet communications. Proc. IEEE Int. Conf. on Communications, p.1518-1523. http://dx.doi.org/10.1109/ICC.2005.1494598