INFORMATICA, 2016, Vol. 27, No. 1, 31–48

© 2016 *Vilnius University*
DOI: http://dx.doi.org/10.15388/Informatica.2016.75

The Moderately Hard DoS-Resistant Authentication Protocol on Client Puzzles

Min-Shiang HWANG[1,2], Song-Kong CHONG[3], Hsia-Hung OU[4]*

[1]*Department of Computer Science & Information Engineering, Asia University*
1500, Lioufeng Rd., Wufeng, Taichung 41354, Taiwan, R.O.C.
[2]*Department of Medical Research, China Medical University Hospital*
China Medical University, No. 91, Hsueh-Shih Road, Taichung 40402, Taiwan, R.O.C.
[3]*Network & Multimedia Institute, Institute for Information Industry*
7F., No. 133, Sec. 4, Minsheng E. Rd., Taipei City 105, Taiwan, R.O.C.
[4]*Graduate Institute of International Sport Affairs, National Taiwan Sport University*
No. 250, Wenhua 1st Rd., Guishan, Taoyuan 33301, Taiwan, R.O.C.
e-mail: mshwang@asia.edu.tw, skchong@iii.org.tw, blueou@gmail.com

Received: June 2014; accepted: December 2014

Abstract. Denial-of-service (DoS) attacks against server resources exhaustion are a major security threat to the Internet. A number of defense mechanisms have been proposed against such attacks. Recently, Aura et al. proposed a solution to resist DoS attacks against an authentication protocol. However, their puzzle solution cannot guarantee that all of their clients have fair computation time to solve a puzzle. The solution may even render some clients unable to obtain the puzzle solution within the lifetime, resulting in a lack of service from the server. In this paper, a simple solution as well as an applied authentication protocol was proposed.

Key words: authentication, CGA, client puzzles, DoS attacks, DDoS attacks, fairness computation.

1. Introduction

Recently, denial-of-service (DoS) attacks have become a major problem (Agah and Das, 2007; Patel and Jinwala, 2015; Ren, 2007; Xuan *et al.*, 2010). These attacks degrade the services' quality or temporarily deny the victim's resource availability, rather than subvert the victim's data or service permanently (Douligeris and Mitrokotsa, 2004). Such attacks are difficult to solve because they are not aimed at any specific weakness of a computer system (Wang and Shin, 2003).

Through a very simple and powerful technique, distributed denial-of-service (DDoS), an attacker can easily exhaust the resources of a victim within a short time (Douligeris and Mitrokotsa, 2004; Jeyanthi and Iyengar, 2012). A DDoS attack system can usually be treated as a many-to-one dimension of the DoS problem (Gupta *et al.*, 2012; Lee *et al.*, 2009; Udhayan and Hamsapriya, 2011). In this attack mode, an attacker controls

*Corresponding author.

multiple handlers (masters) with a special program. With automatic scanning and propagation techniques, a handler is able to dominate multiple agents by searching out their system security holes and injecting malicious instructions (Wang and Shin, 2003). The agents then generate the bogus packets and send them to the victim. This consequently paralyzes the services provided by the victim.

DoS attacks to server resources exhaustion have become a major threat in today's Internet and other open communication systems (Aura *et al.*, 2001; Malekzadeh *et al.*, 2011; Mihajlov and Bogdanoski, 2014). These valuable resources usually include CPU cycles, disk space, memory, network bandwidth, network connectivity, and certain environmental resources such as power (Wang and Reiter, 2003). The service capabilities of a server are usually limited to the resources it holds.

Because the resources of a server are limited and valuable, eliminating a client can arbitrarily access the server resources. Therefore, the server usually implements an authentication protocol to verify the client before allocating any resources to it. Unfortunately, this creates a new opportunity for DoS attacks (Aura *et al.*, 2001). During the verification process, the server typically needs to spend its resources, such as buffer, storage (to store some specific session state), and some expensive computation (e.g. public-key-based operations) to accomplish an authentication before the client can gain access to the server. Therefore, an attacker can exploit DDoS techniques to launch and send many bogus authentication requests to a victim server. As a result, the server is paralyzed because of the exhaustion of its limited resources.

Client puzzles (Abliz and Znati, 2009; Aura *et al.*, 2001; Bocan, 2004; Fallah, 2010; Juels and Brainard, 1999; Wang and Shin, 2003) are one of the eminent solutions created to preclude such attacks. To solve the server resource exhaustion attacks by client puzzles protocol, a client is required to commit his or her resources to solve the puzzle before acquiring any reliable authentication from the server. Therefore, the protocol imposes a large computation burden on potential attackers, by requesting them to generate large volumes of legitimate service requests to exhaust the server resources. The client puzzles will increase the difficulty whether authentications are malicious or honest. The difference is that an honest client needs only to solve the puzzle once, so he/she has little influence; but a malicious client to launch Dos attack will be overloaded because many puzzles need to be solved by him; and therefore the server could also get breathing time.

Recently, Aura *et al.* (2001) proposed a client puzzle to defend the authentication protocol of a server from DoS attacks. However, this puzzle solution cannot guarantee that all of the clients will have adequate fairness computation time to solve the puzzle. Considering the different input parameters in Aura et al.'s protocol, it is found that the client may not be able to solve a puzzle solution within a particular period of time. That is, there are numerous opportunities for some clients to receive no service from the server.

In this paper, a moderately difficult client puzzle protocol has been proposed to solve such problems. By utilizing a unique puzzle for each client, the solution guarantees that each client will receive services from the server within a particular period of time. Thus, legitimate clients only slightly degrade a connection request confirmation when the server is under attack, while attackers have to commit a large computational resource to interrupt the normal services of the server.

The remainder of the paper is organized as follows: Section 2 presents related work, including a brief description of the weakness of Aura et al.'s client puzzles protocol. The flaws of Bocan's protocol are also included in this section. Section 3 describes the attack model to assist us in formulating the client puzzle protocol. Section 4 introduces the client puzzle protocol. Section 5 demonstrates a simple but efficient authentication protocol, which absorbs the characteristics of cryptographically generated address (CGA) into the proposed client puzzle protocol to defend against serial DoS attacks, for example, resource exhaustion attacks in server sites and malicious flooding. Section 6 valuates the security of the proposed protocol, and Section 7 concludes the paper.

2. Related Works

First, existing client puzzle techniques were explored. Second, the weaknesses of Aura et al. protocol were briefly described. To enhance the protocol, the flaws of Bocan's protocol were analyzed and some improvements to the weaknesses were made in Section 2.3.

2.1. *Client Puzzles*

The most effective defense mechanisms of eradicating security loopholes is from their origins. The Internet is based on TCP/IP that was not designed for security in original version, so its insecurity is unavoidable (Baltatu *et al.*, 2000). Although IPSec was introduced to protect the packet, it still has many problems that need to be solved (Baltatu *et al.*, 2000; Deng *et al.*, 2002; Nikander, 2001b). Internet service providers (ISPs) are typically not motivated to embed router-based defense schemes into their routers because there are no direct benefits for them, or for their clients (Wang and Shin, 2003). DoS as well as DDoS has now fully exploited the security weaknesses mentioned above to launch attacks by engaging in flooding attacks with IP spoofing.

Therefore, the most effective way is to develop the server-based defense mechanisms because this side usually experiences the greatest weight of an attack. For this reason, potential victims are always willing to sacrifice some of their resources and performances to reinforce their defense capability (Douligeris and Mitrokotsa, 2004). Client-puzzle was designed to enhance the capability of the server to defend against DoS and DDoS attacks.

According to Bocan (2004), the idea of cryptographic puzzles was developed from Merkle (1978). However, his puzzles were only applied to key agreement, not authentication. Juels and Brainard (1999) proposed their client puzzles to protect victims' machines. Their protocol focuses on defense against connection depletion attacks. This protocol is capable of combating attackers on internal networks, as well as the SSL protocol, which is vulnerable to computational exhaustion attacks. However, to complete a verification, a server requires m hash computations (m is number of sub-puzzles within a puzzle), and this is considered to be too expensive.

Recently, Aura et al. considered that Juels and Brainard only concentrated on connection depletion attacks, but ignored DoS attacks against authentication protocols. Therefore, they made some improvements to the protocol (Aura *et al.*, 2001). Aura et al. mini-

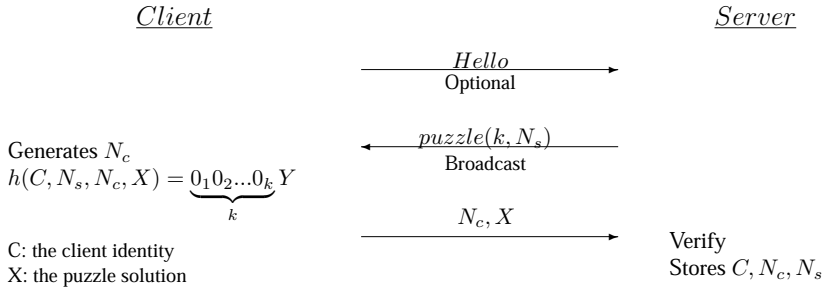

Fig. 1. Sketch of Aura et al.'s protocol.

mized the length of the puzzle and the number of hash operations needed in the verification phase. They used only one hash operation in their client puzzles protocol. Thus, their protocol is more efficient than that of Juels and Brainard.

Later, Bocan (2004), Bocan and Fagadar-Cosma (2005) considered that Aura et al.'s protocol is vulnerable to the "strong attacks"; therefore, he proposed an improvement known as the *threshold puzzle* to defend against the attacks. Their protocol forces the server to immediately cease communication with a client who submits the puzzle solution in less than an estimated time for solving the puzzle.

There are some studies to improve the client puzzles and to apply them to the actual application environment, for example, the Ma's Hash-chain-reversal puzzle (Ma, 1993), and the Lei et al.'s Quasi-Partial-Collisions puzzle (Lei *et al.*, 2006) for the UMTS communication protocol. Since the IP-based network was already the present network protocol tendency, an applied authentication protocol based on our client puzzle protocol was also proposed in this paper, and that is suitable at the IP-based network environment.

A number of approaches concentrate on the client's memory resources (Dwork *et al.*, 2003). Although this paper is confined to CPU-intensive puzzles herein, this research will be explored in future work.

2.2. *The Aura et al.'s Protocol*

A sketch of Aura *et al.* (2001) protocol is shown in Fig. 1. When the server suspects that it is under a DoS attack and its resources are becoming exhausted, it begins to broadcast a puzzle with a difficulty level greater than zero to all of its clients. The puzzle message contains a nonce N_s from the server, and a value k, which demonstrates the puzzle difficulty level. The server generates a nonce N_s periodically. The nonce should be unpredictable to prevent an attacker from pre-computing the puzzle solutions. The server sets $k = 0$ if no work is required for the client.

When a client receives a puzzle message and discovers that the value k is greater than zero, he or she should solve the puzzle according to the puzzle difficulty level k. The client first generates a nonce N_c, then tries to solve the puzzle by finding the hash result in which the most significant bits are conformed to zero. For example, if $k = 16$, the client should find a hash result in which the most significant bits are composed of 16 zero bits. The client must repeatedly apply a hash function with a various values of X until a solution

is found. This is the only efficient way to find a puzzle solution. No other short cuts exist (Aura *et al.*, 2001; Bocan, 2004; Juels and Brainard, 1999; Wang and Reiter, 2003).

After the client finds the puzzle solution X, he or she sends N_c and X to the server. The server first verifies that N_s is fresh, and that C, N_s, and N_c have not been used before. Afterwards, the server verifies the solution by employing only one hash function computation. If the solution is correct, the server stores C, N_s, and N_c. The server stores the correctly solved instances C, N_s, and N_c to prevent the client from re-using the same solution for many service requests.

2.3. *The Bocan's Improved Protocol*

Bocan first proposed a "strong attack" (Bocan, 2004; Bocan and Fagadar-Cosma, 2005) in 2004. A strong attack is defined as a DoS attack launched by an attacker who is able to access a massive computing power. With this attack, an attacker can gradually raise the puzzle up to impossible difficulty. Therefore, legitimate clients will never solve the puzzle. Bocan realized that protocol of Aura *et al.* (2001) was vulnerable to such strong attacks. Therefore, he suggested that when a server received a puzzle solution from a client, it should estimate the time that a client needed to solve a puzzle of difficulty k, using the formula:

$$T_{estimated} = \left(2^k - 1\right) * T_{operation}.$$

In the formula, $T_{estimated}$ represents the estimated time for solving the puzzle, and $T_{operation}$ indicates the minimum time needed to perform a cryptographic operation. Bocan named a client puzzle by the above formula as *threshold puzzle*.

2.4. *The Flaw of Their Protocols*

In the Aura et al.'s protocol, the puzzle difficulty is represented by parameter k. To solve a puzzle of difficulty k, the expected number of steps for a client will be $2^k/2 = 2^{k-1}$, whilst the worst case will be 2^k. Because the server changes the value of N_S periodically, we consider if some clients need to solve a puzzle under the worst case with insufficient lifetime of N_S, these clients will get nothing after they solve the puzzle. In Aura et al.'s protocol, the server decides only the value N_s and the puzzle difficulty level k. The other parameters such as C and N_c are unique to each client. Because each client will employ their own C and N_c to find the solution X, it is known that inputting various parameters into a hash function cannot guarantee a fair computation time. With improper settings of N_S, some clients may still be rejected by the server.

However, Bocan's protocol originated from Aura et al.'s idea, therefore, the weaknesses mentioned above were inherited. That is, Bocan's protocol may also cause a difference probability in solving the puzzle. It is difficult to estimate the time needed to solve a puzzle.

It is noted, however, that our moderately difficult client puzzle protocol needs two additional hash computations in the server site when compared with Aura *et al.* (2001) and

Bocan (2004). However, the protocol ensures that each client will have fair and adequate computation time to find the puzzle solution. Furthermore, the protocol provides a better way to estimate the time for solving the puzzle, which enhances the capability of Bocan's *threshold puzzle*.

3. Attack Model

To formulate our client puzzle protocol, the following assumptions were made: It is assumed that clients C_i come from the Internet, intending to receive some services from server S. Server S provides the services to C_i after he or she has been verified. An attacker Adv (who controls a number of attack daemon agents) from a public network is trying to exploit the weaknesses of the authentication protocol, and launches a DDoS attack to deplete the resources (CPU cycles and memory) of the server S. The following assumptions were made about the situation of the attacker Adv and server S.

Assumption 1. The attack daemon agents controlled by Adv are generally external to the victim's own network.

The attack daemon agents are commonly located outside the victim's own network. This helps the attacker avoid detected by the victim's network operator, and avoid any liability if the attack source is traced back (Douligeris and Mitrokotsa, 2004).

Assumption 2. The attack daemon agents commonly perform IP spoofing.

IP spoofing is a fundamental component of many DDoS attacks. Adv sends a high-volume of authentication request messages to S with fake or randomized IP address sources to render the victim unable to locate attackers.

Assumption 3. When participating in a DDoS attack, a normal user cannot feel any discomfort using the agent computer.

When participating in a DDoS attack, the agent program, located in the attack daemon agents, should use only a small amount of the resources (e.g. CPU cycle, memory and bandwidth). This assumption is related to Assumption 2. If Assumption 2 does not hold, the daemon agents would make efficient responses to S. As a result, a user of the agent computer will experience minimal changes in the computer's performance. The agent program may be removed later so to affect the attack plan of Adv.

Assumption 4. The daemon agents will not receive the messages sent to the spoofed IP address.

According to Assumptions 1 and 2, it is assumed that the daemon agents, which use the spoofed source IP address, will not receive and can only intercept a limited number of messages sent to the spoofed IP address. Adv may control the default router nearby S and attempt to copy the related messages; however, if he or she can do that, then he or she can mount a DoS attack by blocking all the traffic originating from S without having to overload it.

ASSUMPTION 5. *S* is able to cope with any puzzle generation and verification.

Although this is a strong assumption, it remains reasonable for a current technology, and is essential for many defense protocols (Aura *et al.*, 2001; Juels and Brainard, 1999; Montenegro and Castelluccia, 2004; Wang and Reiter, 2003). If *Adv* can flood *S* with a large volume of related messages and paralyze the puzzle generation or verification, none of the defense protocols could resist the attacks. This assumption is considered to express the same idea in different words with Assumption 3 of Juels and Brainard (1999), and Assumption 2 of Wang and Reiter (2003).

It is believed that DoS attack instances can be reduced significantly if the source can be verified through authentication protocols. However, the authentication process may become another potential source of DoS attacks (Nikander, 2001b). Therefore, to provide a DoS-resistant mechanism despite the authentication protocol employed by the server, a client puzzle mechanism to mitigate the destruction of DoS attacks was proposed.

4. The Proposed Client Puzzle Protocol

To provide DoS-resistant to authentication protocols, the client puzzle protocol is described fully in this section. Our protocol mitigates the destructiveness of DoS attacks to authentication protocols, which perform computationally expensive operations when verifying a client. The defense protocol can be employed not only to authenticate protocols, but also to other protocols vulnerable to DoS attacks during their execution.

We do not intend to introduce the authentication protocol suitable for our defense protocol in this paper. A server administrator should make this decision if he or she believes that the server is vulnerable to DoS attacks. The administrator can implement the defense protocol independently, or integrate the defense protocol into the authentication protocol, as described in Section 5.

The notations used in the proposed protocol and the remainder of this article are listed in Table 1. Note that some of the notations, already defined in Section 3, are ignored here.

Table 1
The notations used in the proposed protocol.

Notations	Description
v_i	Solution of the puzzle, represents the puzzle difficulty
t_l	Lifetime of v_i
t_s	Time stamp of server S, extracts from t_l
t_r	Round-trip time issues a puzzle and receives its solution
$T_{operation}$	Time needed to perform a hash operation
$T_{estimated}$	Acceptance threshold of the puzzle solution response time
IP_c	IP address of client C_i
$h(\cdot)$	One-way Hash function, e.g. SHA-1
h_{l104}	Leftmost 104-bit of hash output
h_{r24}	Rightmost 24-bit of hash output
sk_s	128-bit secret key conserved by S
$cookies_{sc}$	Message authentication code (MAC) issues by S to C_i

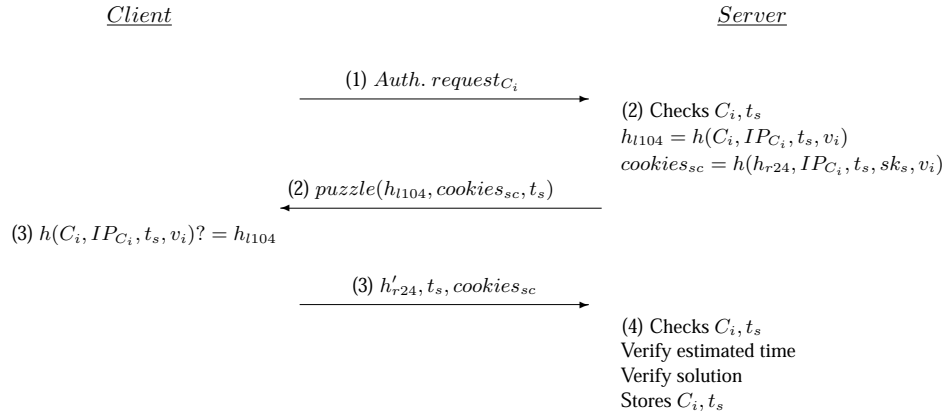

Fig. 2. The proposed client puzzle protocol.

In the proposed client puzzle protocol, server S should decide the puzzle difficulty v_i, according to its computational loading. The v_i is an N bits random number generated according to the puzzle difficulty level. N is determined by the difficulty of puzzle. S stores v_i–t_l as an entry in its memory. If t_l has expired, S will change v_i into a new value, according to its computational loading. Moreover, S keeps both the current value of v_i–t_l and a set of previous entries v_{i-1}–t_{l-1}. The storage of v_{i-1}–t_{l-1} prevents some puzzles from being issued close to the expired time of v_{i-1}. S should accept the solution generated within t_{l-1} according to $T_{estimated}$. For the sake of simplicity, v_i / t_l is exploited instead of the situation of v_{i-1}/t_{l-1} in the description below. Readers should be cautious of this abbreviation. Details of the defense protocol are described below. Figure 2 demonstrates our client puzzle protocol.

Step 1: To request services from server S, client C_i sends its authentication request to server S.

Step 2: When S receives the authentication request, if it is overloaded (e.g. under attack), it sends C_i a puzzle to solve. Otherwise, S authenticates C_i directly. We consider S under attack herein.

Before starting any computation, S first checks the entry C_i-t_s stored in its memory. This is to ensure that one C_i can only launch a corresponding puzzle generation before v_i expires. Afterward S computes $h(C_i, IP_{C_i}, t_s, v_i)$, using a hash operation. S obtains h_{l104}. S then uses the remainder of h_{r24} to form cookies $cookies_{sc}$. Note that $cookies_{sc}$ are issued for each authentication request. To combat IP spoofing, S extracts the IP_{C_i} from the authentication request message as an input parameter into $cookies_{sc}$. S sends h_{l104}, $cookies_{sc}$ and t_s as a puzzle to C_i. S remains stateless to any authentication request from C_i. The server gives the client the answer expected $cookies_{SC}$ in the proposed protocol, then uses the secret key SK_S and Hash function to protect this answer. Therefore, the server does not need to store any information concerning the client applying the connection. This reduces the overload of server. Therefore, S is stateless to any authentication request from C_i.

Step 3: After C_i receives the puzzle, he or she finds the solution v_i, which is the leftmost 104-bit of a hash result equivalent to h_{l104}, received in Step 2. The way of client usually to find solutions is the brute-force attack to test one by one until all possible solutions have to meet to find the solution. Once the solution has been found, C_i sends the remaining 24-bit h'_{r24} of the hash result, t_s and $cookies_{sc}$ to S.

Step 4: Upon receipt of the puzzle solution, S first checks C_i and t_s to prevent double verification. If t_s is fresh, S verifies the estimated time that C_i should take to solve a puzzle of difficulty v_i through a formula $T_{estimated} = (v_i * T_{operation} + t_r)$. Any puzzle solution, which arrives earlier than $T_{estimated}$, is rejected, and the remaining verification process is ignored. Then, S verifies the puzzle solution using h'_{r24}, the IP_{C_i} of the response message and another related value as the input parameter to compute $cookies'_{sc} = h(h'_{r24}, IP_{C_i}, t_s, sk_s, v_i)$. If $cookies_{sc} \equiv cookies'_{sc}$, then the puzzle solution is correct. S stores C and t_s as long as the corresponding v_i/v_{i-1} and t_l/t_{l-1} remain in its memory. S may now commit its resources to execute authentication protocol to verify C_i.

In the simple modification of Aura *et al.* (2001) and Bocan (2004), we proposed the client protocol described above. Through a basic method placing the solution v_i into the puzzle, the defense protocol guarantees that all clients have adequate time to solve the puzzle. Any client engaging in malpractice is rejected by S, according to $T_{estimated}$.

Note that the value of v_i is represented by a set of binary numbers. S should control the value of v_i as an appropriate length. Most importantly, S should ensure that its client can solve the puzzle within t_l. For C_i, he or she should start guessing from 1-bit, and gradually increase the bit length, for example, 0, 1, 01, 10, 11, ... Of course, C_i should know the rules of how to begin the value of v_i. Because v_i is set as the solution of a puzzle, C_i could ultimately achieve the solution v_i. Because S knows the accurate value of v_i, it can be pre-computed and stored $T_{estimated}$ for future use.

5. The Proposed Authentication Protocol

In the next section, an authentication protocol based on IPv6 (Deering and Hinden, 1998) is proposed. The protocol absorbs the characteristics of CGA and employs them in the proposed moderately difficult client puzzle mechanism to defend against serial DoS attacks, for example, resource exhaustion attack in a server site and malicious flooding.

To open the forthcoming discussion, some understanding of the IPv6, as well as the related background, is necessary. We assume the reader is familiar with basic IPv6 architecture and function. Thus, we concentrate on stateless auto-configuration, how the CGA can be used in IPv6 and how it can be integrated into our client puzzle protocol to provide a simple authentication function.

In the basic stateless auto-configuration process, a booting host chooses its 64-bit interface identifier from the interface's MAC address as a link local IP address. The host then performs *Duplicate Address Detection*. Once the host has a link local address, it enters into the *Router Discovery* phase. The host learns the routing prefixes

from the *Router Advertisements*. It is able to create globally routing addresses for itself. Therefore, the 128-bit IP address is divided into a routing prefix and an interface identifier. Because a host cannot change the routing 64-bit address prefix, the CGA focuses on the lower 64-bit interface identifier (link local IP address). The CGA address holds some good characteristics to prove that the address is not only used but also owned by a host. Therefore, IP spoofing can be eliminated (Chen *et al.*, 2008; Montenegro and Castelluccia, 2004).

CGA was first proposed by O'Shea and Roe (2001) to secure binding updates in Mobile IP Version 6 (MIPv6) (Johnson *et al.*, 2004) when IPSec AH was not available. The later improvements can be found in Deng *et al.* (2002), Nikander (2001b). In CGA, it assumes that a global or centralized public key infrastructure (PKI) or key distribution centre (KDC) is not available. All of its securities are based on the probability difficulty of producing duplicate addresses, as described in Montenegro and Castelluccia (2004). We make some variations herein to make CGA not only solve the address ownership problem (Nikander, 2001a), but also to authenticate itself to the server. We named the CGA with a certificate from a trust third party as VCGA (Variant-CGA).

To request services from server S, client C_i should ensure that the IP address (more accurately, the part of interface identifier) is derived from the following computation:

$$host\,ID = HASH_{62}(public\ key|imprint).$$

The *imprint* acts like a *salt* in UNIX system to limit certain types of attacks (Montenegro and Castelluccia, 2004). As suggested in Nikander (2001b), O'Shea and Roe (2001), only 62 bits of the lower IPv6 address can be used to store a cryptographic hash of a public key. The other two bits have the semantics defined for EUI-64 global identifiers (O'Shea and Roe, 2001). We use these 62 bits *host ID* instead of the *interface identifier* in IPv6. Through the characteristics of VCGA, an attacker is hard to perform IP spoofing. Additionally, a simple authentication can be achieved. We show the related security analysis in Section 6.

The proposed authentication protocol shown in Fig. 3 is a variation of the proposed client puzzle protocol described in Section 4. It is a practical DoS-resistant authentication protocol, built to defend against resources exhaustion attacks in server sites.

Step 1: To request services from server S, client C_i sends its authentication request to server S.

Step 2: When S receives the authentication request, if it is overloaded (e.g. under attack), it sends C_i a puzzle to solve. Otherwise, S authenticates C_i directly. We consider S under attack herein.

Before starting any computation, S first checks the entry C_i–t_s stored in its memory. This is to ensure that one C_i can only launch a corresponding puzzle generation before v_i expires. Afterward S computes $h(C_i, IP_{C_i}, t_s, v_i)$, using a hash operation. S obtains h_{l104}. S then uses the remainder of h_{r24} to form cookies $cookies_{sc}$. Note that $cookies_{sc}$ are issued for each authentication request. To combat IP spoofing, S extracts the IP_{C_i} from the authentication request message as an input parameter

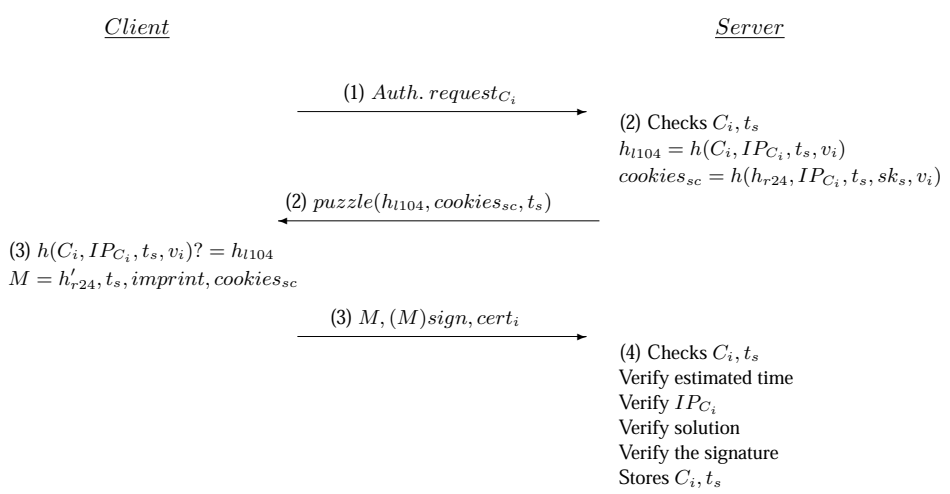

Fig. 3. The proposed authentication protocol.

into $cookies_{sc}$. S sends h_{l104}, $cookies_{sc}$ and t_s as a puzzle to C_i. S remains stateless to any authentication request from C_i. The server gives the client the answer expected $cookies_{SC}$ in the proposed protocol, then uses the secret key SK_S and Hash function to protect this answer. Therefore, the server does not need to store any information concerning the client applying the connection. This reduces the overload of server. Therefore, S is stateless to any authentication request from C_i.

Step 3: After C_i receives the puzzle, he or she performs a brute force to search out the solution v_i, which the leftmost 104-bit of a hash result equivalent to h_{l104} received in Step 2. Once the solution is found, C_i sends the remaining 24-bit h'_{r24} of the hash result, t_s, *imprint* as well as $cookies_{sc}$ as M to S. A signed M and $cert_i$ are also sent to S. The public key PK_i and *imprint* should be the elements used to create the 62 bits *host ID*. The VCGA employed herein also signifies that C_i is prevented from using a spoofed IP address IP_{C_i} in Step 2.

Step 4: Upon receipt of M, $(M)sign$ and $cert_i$, S checks C_i and t_s and verifies $T_{estimated} = (v_i * T_{operation} + t_r)$. Afterwards, S verifies IP_{C_i} by checking $host\ ID? = HASH_{62}(public\ key|imprint)$. If the verification holds, then S takes IP_{C_i} into the puzzle solution verification. Only after determining if the above verification is positive, S verifies the signature. If the signature verification is correct, S believes that C_i not only uses but also owns the IP address. S stores C and t_s as long as the corresponding v_i/v_{i-1} and t_l/t_{l-1} remain in its memory. The services provided by S may now be accessible by C_i.

Because the authentication protocol employs a public key cryptosystem to verify client C_i, with a masterly employment of certificate $cert_i$, the proposed authentication protocol not only verifies C_i but also proves that S is communicating with the owner of the IP address. A malicious client who intends to launch flooding attacks (e.g. requests services from video streaming servers) against a victim will be identified by his or her $cert_i$ and signature. Because the original CGA assumes that a trustworthy PKI is not available,

its self-created public or private key by C_i is difficult to declare the crime of an attacker judicially through the law (because there are no trusted third parties to prove the identity of an attacker).

Because the server only accepts those clients who can prove their ownership of the source IP address, the authentication protocol eliminates IP address spoofing. A malicious client is prevented from flooding a victim, because now he or she needs to prove ownership of the IP address with a signature.

6. Security Analysis

In the protocols above, upon receipt of the authentication request from client C_i, server S begins with a $h(C_i, IP_{C_i}, t_s, v_i)$ computation. S will then take the leftmost 104-bit of hash results to perform a puzzle requirement. Afterward C_i is required to find the v_i, which results in the leftmost 104-bit of the hash output (say h'_{l104}) being equal to h_{l104}. S will then verify the solution of the puzzle. We assume that the hash function $h(\cdot)$ used in the defense protocol is a random function. For each input, the output of $h(\cdot)$ is uniformly distributed in $\{0, 1\}^b$, where b is the security parameter.

To simplify our security analysis, we state the analysis in the form of a proposition and divide it into two aspects: the analysis of the proposed client puzzle protocol, and the analysis of the authentication protocol.

6.1. *Analysis of the Proposed Client Puzzle Protocol*

The analysis made in this subsection does not take the succeeding authentication method into account. We ignore the existence of the proposed authentication protocol, as demonstrated in Section 5 herein.

Proposition 1. *The probability to find a collision of $h(\cdot)$ is no more than $2^{-(104-1)}$.*

Note that if we curtail the leftmost 104-bit of the hash function output into a small number of sizes, say 8-bit, for every hash function output, whose leftmost 8-bit fits with the 8-bit puzzle requirement sent by S, will be an acceptance solution. Therefore the probability of finding an 8-bit collision of $h(\cdot)$ becomes $2^{-(8-1)}$. This is an unacceptable probability because of its very large size.

We exploit the leftmost 104-bit of hash function output as a puzzle requirement to C_i. This is to ensure that the collision of the hash function becomes negligibly small. In this situation, the probability of finding a collision of $h(\cdot)$ is no more than $2^{-(104-1)}$. Therefore, the only efficient way to find h_{l104} is through brute force until the solution v_i is achieved.

Although 104-bit increases the message length in transmission, it can provide a higher security to resist collision. The server administrator should consider this tradeoff.

Proposition 2. *The maximum time taken by C_i to obtain $h'_{l104} \equiv h_{l104}$ is limited to v_i.*

In the defense protocol, the number of times taken by C_i to find the correct value of v_i is set by S in advance. S can estimate the time needed to complete the corresponding puzzle, using a simple calculation $v_i * T_{operation}$. Henceforth S can accurately control a difficult puzzle. Because S randomly varies v_i when t_l has expired, the efficient way to obtain v_i is through brute force.

Note that the increasing length of value v_i demonstrates that C_i should take more time to seek out the solution. S should control the size of v_i as an appropriate difficulty, and ensure that each client can solve the puzzle within the lifetime t_l of v_i.

Proposition 3. *Under appropriate estimation to t_r, S can assure that each client will have adequate time to solve the puzzle.*

We make Proposition 3 essentially to defeat potential *Adv*, who can launch *strong attacks* (Bocan, 2004). We suggest the calculation of $T_{estimated}$ should have some cushioning to mitigate the affect of various CPU powers of each client, for example, loosen the restriction of $T_{operation}$. On the other hand, appropriation of the estimated value of t_r is beyond the scope of this paper. We reject the discussion of t_r herein.

Proposition 4. *Adv will be defeated if he or she has a puzzle solution v_i but lacks $cookies_{sc}$.*

Although *Adv* can copy the v_i and multiply many puzzle solutions to S, the lack of *cookies* issued to each previous authentication request message will make S discover the deception of *Adv*. Therefore, to limit the reuse of puzzle solutions, S places its secret key sk_s into the computation of $cookies_{sc}$. Any client who has the puzzle solution should affix the related $cookies_{sc}$ to S. The server uses the $cookies_{sc}$ to prevent IP spoofing. Therefore, any *Adv* who uses spoofed IP must make sure that he or she can obtain the corresponding $cookies_{sc}$. Because of *Adv* with a randomized source of IP is hard to control the flow of corresponding messages. *Adv* makes it difficult to obtain enough $cookies_{sc}$ in the environment of the Internet.

Although *Adv* can use a real IP address to receive $cookies_{sc}$ and perform the related puzzle solving, he or she may confront liability if the attack is traced back. In the meantime, *Adv* also violates Assumptions 2 and 3 in Section 3 which are the common rules to observe when participating in a DDoS attack.

Proposition 5. *With stored entry C_i-t_s, C_i as well as Adv is prevented from performing double puzzle generation and verification.*

Because the CPU cycles of S is valuable, we examine the C_i and t_s in Steps 2 and 4 of the proposed protocol before we began any computation. There are two reasons for this. First, a client can only achieve his puzzle generation or verification no more than one time before t_l expires. Upon receipt of the authentication request message from C_i, S first records the particular time stamp, say t_c. If t_c is placed within the t_l of v_i, then S ignores the puzzle generation. Similar action is taken by S when it receives a puzzle solution

from C_i. The examination of C_i and t_s protects against resources wasted by unintentional clients.

Second, lures S to perform a heavy authentication protocol computation, *Adv* should submit a puzzle solution with a different identity to C_i, as well as the corresponding $cookies_{sc}$ which have not been used before. However, he/she will face difficulties as we described in Proposition 4.

6.2. *Analysis of the Proposed Authentication Protocol*

The authentication protocol inherits all of the security virtues of the proposed client puzzle protocol described above. We ignore the duplicate descriptions herein.

Theorem 1. *The probability of the duplicate VCGA address is small and acceptable in reality.*

By birthday paradox, there is a 50% probability of collision after trying $1.2 \times 2^{n/2}$ (Menezes *et al.*, 1997). In VCGA addresses, only 62 bits are actually usable to store a cryptographic hash of a public key. If $n = 62$, we need 1.2×2^{31}, that is, 2.58 billion hosts on an average before any two of them produce identical addresses. Therefore, it is acceptable in reality, if we consider this collision harmful only if the two hosts are in the same site (e.g. they are using the same 64-bit prefix) and have the same correspondent server. This probability is very unlikely. Additionally, the *duplicate address detection* prevents this collision from happening.

Theorem 2. *The owner of an IP address can be identified by VCGA.*

In an original CGA, a global or centralized PKI or KDC is not available; therefore a client self-creates a public or private key pair. The CGA reduces the construction of trusted third parties. It seems efficient, but it is, in fact, insecure.

The security of CGA is based on the belief that to obtain an IP address via a hash of the public key is safe, and that it is difficult for an attacker to generate a corresponding private key to make a signature. Because 62 bits are too few to gain strong security (Nikander, 2001b), and *imprint* is not a secret value as expected for HMAC use, therefore, by brute force, an attacker can generate many public or private key pair to find a collision (the hash result of a public key is a collision with another client, that is, two *host ID*). As a result, the attacker can claim that he/she is the owner of some IP addresses, because he or she can provide a correct signature to the corresponding parties. The lack of trusted third parties makes these weaknesses unavoidable.

Our authentication protocol is based on general public key cryptosystems; therefore each client obtains his or her public or private key pair and a corresponding certificate $cert_i$. With $cert_i$, there is no way for an attacker to impersonate someone, because now he or she needs to obtain the actual private key, corresponding to the public key, enrolled in $cert_i$. Because server S verifies its clients when they request services, only the true clients obtain the authority of S and are permitted to access the resources. According to

this concept, the proposed authentication protocol employs the public key enrolled in $cert_i$ to improve the security of CGA.

Because S can verify the identity of a client via VCGA address and the related signature, the owner of an IP address can be identified unquestionably through the utilization of $cert_i$. It is useless for an attacker to find a collision of someone's IP address by brute force, and claim that he or she is the owner of the IP address with a fake signature.

Theorem 3. *With the characteristics of VCGA, the malicious client flooding can be precluded in the proposed authentication protocol.*

Because *host ID* is derived from $HASH_{62}(public\ key|imprint)$, with a fixed *public key* from the certificate $cert_i$, an attacker cannot generate a victim's IP address and launch flooding attacks if he or she does not know the victim's value of $imprint$. Although the $imprint$ is not a secret value as expected for HMAC use, it severely limits the attacks to only those victims with a privileged location and within a certain time period (Montenegro and Castelluccia, 2004).

The only way to launch malicious client flooding (Deng *et al.*, 2002) is to lure the server S into believing that the service request is coming from a proper client. By doing so, an attacker first needs to obtain the designated victim's *imprint* and generate the related IP address. Second, the signature on the related message should be approved by S. Because the attacker does not know the private key of the victim, a verifiable signature can be created, using the attacker's true $cert_i$ and signature. The consequences of the attacks are serious, the true certificate $cert_i$ and signature will make the attacker punishable by law. The original CGA lacks these advantages.

7. Discussions and Conclusions

To simplify our performance analysis, we state the analysis in two aspects: the analysis of the proposed client puzzle protocol, and the analysis of the authentication protocol. The Client (C_i) of the proposed client puzzle protocol only needs 2 times Hash operations ($h(C_i, IP_{C_i}, t_s, v_i)$ and h'_{r24}) at Step (3) in Fig 2. In a similar way, the Client (C_i) of the authentication protocol also only needs 2 times Hash operations ($h(C_i, IP_{C_i}, t_s, v_i)$ and h'_{r24}) at Step (3) in Fig. 3.

Undoubtedly, DDoS attacks against the server resource exhaustion are already becoming a major security threat. In a business environment, a server needs to verify its clients through various authentication protocols, before providing any services. However, such a verification may result in DDoS attacks, especially for those heavy loading authentication protocols, for example, public-key-based operation. Therefore, it is important to provide a defense mechanism to resist such attacks.

We have presented a defense protocol to combat DDoS attacks. The protocol provides a guarantee to all the clients, which has adequate computation time to find the puzzle solutions. We improve the shortcomings of Aura et al.'s protocol by bringing up an idea *fairness computation time*. Moreover, S is stateless to any authentication request from C_i

in the proposed. It can reduce the load of the server even more in the face of numerous connections of the DoS attack.

To improve the weaknesses of CGA, we propose an idea called VCGA. The proposed authentication protocol with VCGA not only eliminates the drawbacks from IP spoofing and malicious client flooding, but also inherits all the security virtues of the proposed client puzzle protocol.

Each client is able to access the authentication protocol after he or she solves the puzzle. However, an attacker who intends to lure a server into performing heavy authentication protocol computations will be confronted with various difficulties.

Acknowledgments. The authors wish to thank many anonymous referees for their suggestions to improve this paper. Part of this research was supported by the National Science Council, Taiwan, R.O.C., under contract Nos. MOST 103-2221-E-468-026, NSC103-2622-E-468-001-CC2, and NSC103-2622-H-468-001-CC2.

References

Abliz, M., Znati, T. (2009). A guided tour puzzle for denial of service prevention. In: *2009 Annual Computer Security Applications Conference (ACSAC'09)*, pp. 279–288.

Agah, A., Das, S.K. (2007). Preventing DoS attacks in wireless sensor networks: a repeated game theory approach. *International Journal of Network Security*, 5(2), 145–153.

Aura, T., Nikander, P., Leiwo, J. (2001). DOS-resistant authentication with client puzzles. In: *Security Protocols, 8th International Workshop Cambridge, UK, April 2000, Revised Papers*. *Lecture Notes in Computer Science*, Vol. 2133. Springer-Verlag, Berlin, pp. 170–177.

Baltatu, M., Lioy, A., Maino, F., Mazzocchi, D. (2000). Security issues in control, management and routing protocols. *Computer Networks*, 34(6), 881–894.

Bocan, V. (2004). Threshold puzzles: the evolution of DOS-resistant authentication. *Transactions on Automatic Control and Computer Science*, 49(63).

Bocan, V., Fagadar-Cosma, M. (2005). Adaptive threshold puzzles. In: *The International Conference on Computer as a Tool. EUROCON'05*, Belgrade, Serbia and Montenegro.

Chen, Y., Das, S., Dhar, P., Saddik, A.E., Nayak, A. (2008). Detecting and preventing IP-spoofed distributed DoS attacks. *International Journal of Network Security*, 7(1), 69–80.

Deering, S., Hinden, R. (1998). *Internet protocol, version 6 (IPv6)*. Technical report RFC 2460, IETF.

Deng, R.H., Zhou, J.Y., Bao, F. (2002). Defending against redirect attacks in mobile IP. In: *Proceedings of the 9t ACM Conference on Computer and Communications Security, Washington, USA*, pp. 59–67.

Douligeris, C., Mitrokotsa, A. (2004). DDoS attacks and defense mechanisms: classification and state-of-the-art. *Computer Networks*, 44(5), 643–666.

Dwork, C., Goldberg, A., Naor, M. (2003). On memory-bound functions for fighting spam. In: *Proceedings of the 23rd Annual International Cryptology Conference (CRYPTO'03), Santa Barbara, USA*, pp. 426–444.

Fallah, M. (2010). A puzzle-based defense strategy against flooding attacks using game theory. *IEEE Transactions on Dependable and Secure Computing*, 7(1), 5–19.

Gupta, B.B., Joshi, R.C., Misra, M. (2012). ANN based scheme to predict number of zombies in a DDoS attack. *International Journal of Network Security*, 14(2), 61–70.

Jeyanthi, N., Iyengar, N.C.S.N. (2012). An entropy based approach to detect and distinguish DDoS attacks from flash crowds in VoIP networks. *International Journal of Network Security*, 14(5), 257–269.

Johnson, D., Perkins, C., Arkko, J. (2004). *Mobility support in IPv6*. RFC 3775, requset for comments.

Juels, A., Brainard, J. (1999). Client puzzles: a cryptographic defense against connection depletion attacks. In: *Proceedings of NDSS '99 (Networks and Distributed Security Systems)*, pp. 151–165.

Lee, M.C., He, Y.J., Chen, Z. (2009). Towards improving an algebraic marking scheme for tracing DDoS attacks. *International Journal of Network Security*, 9(3), 204–213.

Lei, Y., Pierre, S., Quintero, A. (2006). Client puzzles based on quasi partial collisions against dos attacks in umts. In: *2006 IEEE 64th Vehicular Technology Conference*, pp. 1–5.

Ma, M. (2005). Mitigating denial of service attacks with password puzzles. In: *International Conference on Information Technology: Coding and Computing (ITCC 2005)*, pp. 621–626.

Malekzadeh, M., Ghani, A.A.A., Subramaniam, S., Desa, J. (2011). Validating reliability of *OMNeT*$^{++}$ in wireless networks DoS attacks: simulation vs. testbed. *International Journal of Network Security*, 13(1), 13–21.

Menezes, A., Oorschot, P.V., Vanstone, S. (1997). *Handbook of Applied Cryptography*. CRC Press, Boca Raton.

Merkle, R.C. (1978). Secure communication over an insecure channel. *Communication of ACM*, 21(4), 294–299.

Mihajlov, B., Bogdanoski, M. (2014). Analysis of the WSN MAC protocols under jamming DoS attack. *International Journal of Network Security*, 16(4), 304–312.

Montenegro, G., Castelluccia, C. (2004). Crypto-based identifiers (CBIDs): concepts and applications. *ACM Transactions on Information and System Security*, 7(1), 97–127.

Nikander, P. (2001a). An address ownership problem in IPv6. *draft-nikander-ipng-address-ownership-00.txt*.

Nikander, P. (2001b). Denial-of-service, address ownership, and early authentication in the IPv6 world. In: *9th International Workshop on Security Protocol, Cambridge University*, pp. 12–26.

O'Shea, G., Roe, M. (2001). Child-proof authentication for MIPv6 (CAM). *ACM SIGCOMM Computer Communication Review*, 31(2), 4–8.

Patel, H., Jinwala, D.C. (2015). Automated analysis of internet key exchange protocol v2 for denial of service attacks. *International Journal of Network Security*, 17(1), 66–71.

Ren, W. (2007). Pulsing RoQ DDoS attacking and defense scheme in mobile Ad Hoc networks. *International Journal of Network Security*, 4(2), 227–234.

Udhayan, J., Hamsapriya, T. (2011). Statistical segregation method to minimize the false detections during DDoS attacks. *International Journal of Network Security*, 13(3), 152–160.

Wang, X., Reiter, M.K. (2003). Defending against denial-of-service attacks with puzzle auctions. In: *Proceedings of the 2003 IEEE Symposium on Security and Privacy, Berkeley, USA*, pp. 78–92.

Wang, H., Shin, K.G. (2003). Transport-aware IP routers: a built-in protection mechanism to counter DDoS attacks. *IEEE Transactions on Parallel and Distributed Systems*, 14(9), 873–884.

Xuan, Y., Shin, I., Thai, M.T., Znati, T. (2010). Detecting application denial-of-service attacks: a group-testing-based approach. *IEEE Transactions on Parallel and Distributed Systems*, 21(8), 1203–1216.

M.-S. Hwang received the BS in Electronic Engineering from the National Taipei Institute of Technology, Taipei, Taiwan, Republic of China (ROC.), in 1980; the MS in Industrial Engineering from the National Tsing Hua University, Taiwan, in 1988; and a PhD in Computer and Information Science from the National Chiao Tung University, Taiwan, in 1995. He also studied Applied Mathematics at the National Cheng Kung University, Taiwan, from 1984–1986. Dr. Hwang passed the National Higher Examination in field "Electronic Engineer" in 1988. He also passed the National Telecommunication Special Examination in field "Information Engineering", qualified as advanced technician the first class in 1990. From 1988 to 1991, he was the leader of the Computer Center at Telecommunication Laboratories (TL), Ministry of Transportation and Communications, ROC. He was also the chairman of the Department of Information Management, Chaoyang University of Technology (CYUT), Taiwan, during 1999–2002. He was a professor and chairman of the Graduate Institute of Networking and Communications, CYUT, during 2002–2003. He was a professor and chairman of the Department of Management Information Systems, National Chung Hsing University (NCHU), during 2005–2009. He was an outstanding

professor of the department of Management Information Systems, NCHU, during 2007–2011. He obtained the 1997, 1998, 1999, 2000, and 2001 Outstanding Research Award of National Science Council of the Republic of China. He is currently a Chair Professor of the department of Computer Science & Information Engineering, Asia University. He is a member of IEEE, ACM, and Chinese Information Security Association. His current research interests include Information Security, electronic commerce, database and data security, cryptography, image compression, and mobile computing. Dr. Hwang has published over 170+ articles on the above research fields in international journals.

S.-K. Chong received the BS degree in Information Management and MS in Graduate Institute of Networking and Communication Engineering from Chaoyang University of Technology (CYUT), Taichung, Taiwan, Republic of China, in 2002 and in 2004. He received the PhD in Computer Science and Information Engineering from National Cheng Kung University, Taiwan, Republic of China, in 2010. He is a senior engineer of Institute for Information Industry, Taiwan. His current research interests include cryptography, information security, and network security.

H.-H. Ou received the BS and MS in Information Management from Chaoyang University of Technology, Taiwan, Republic of China, in 1999 and 2001, respectively. He received the PhD in Computer Science and Engineering at National Chung Hsing University, Taiwan, Republic of China, in 2009. Currently, he is an associate professor of National Taiwan Sport University, Taiwan; his current research interests include mobile agent, network security, mobile communication and sport information management.

Autentifikavimo protokolas, apsaugantis vartotojus nuo atakų prieš serverį

Min-Shiang HWANG, Song-Kong CHONG, Hsia-Hung OU

Interneto saugumui svarbi grėsmė yra ataka prieš serverio išteklius, kuri sumažina aptarnavimo kokybę arba net laikinai neleidžia vartotojui naudotis serverio paslaugomis. Prieš šias atakas kovoti yra sunku, nes jos nepriklauso nuo kompiuterinės sistemos parametrų. Buvo pasiūlyta keletas apsaugos būdų nuo šių atakų. Neseniai Aura ir kt. kovai su tokiomis atakomis pasiūlė autentifikavimo protokolą, tačiau šis protokolas negarantuoja, kad visi vartotojai turės pakankamai laiko apsiginti nuo atakos, o kai kurių vartotojų serveris netgi neaptarnaus. Straipsnyje pasiūlytas paprastas šios problemos sprendimas ir jį įgyvendinantis autentifikavimo protokolas.