Requirements for Client Puzzles to Defeat the Denial of Service and the Distributed Denial of Service Attacks

Vicky Laurens[1], Abdulmotaleb El Saddik[1], and Amiya Nayak[2]
[1]Multimedia Communications Research Laboratory University of Ottawa, Canada
[2]School of Information Technology & Engineering University of Ottawa, Canada

Abstract: *Client puzzle protocols represent a promising technique for defeating resource depletion Denial of Service (DoS) attacks. Practical implementations of client puzzle protocols not only reported positive results in achieving such a challenging goal (preventing DoS attacks), but also these implementations overcame, up to a certain degree, one of the first disadvantages of client puzzle protocols: Their interoperability with current Internet communication protocols. However, the question on whether client puzzle protocols can thwart the Distributed Denial of Service (DDoS) attacks is still under investigation. Due to the increasing number of DDoS attacks, their prevention has become very important. Based on the puzzle generation and verification processes, and focusing mainly on forestalling DDoS attacks, this paper classifies and analyzes current proposals of client puzzle protocols. The paper not only reveals and analyzes their limitations with regards to the prevention of DDoS attacks, but also outlines a general approach for addressing the identified limitations. We propose a solution based on the general principle that under attack legitimate clients should be willing to experience some degradation in their performance in order to obtain the requested service. Our proposal is based on including a puzzle-solution request in different states of a given connection such that the computational load for solving the puzzles will be noted but the clients' operations will not be totally interrupted.*

Keywords: *Security attacks, distributed denial of service.*

Received May 12, 2005; accepted August 3, 2005

1. Introduction

Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks are aimed to thwart legitimate users from having access to shared resources. In general, on the Internet, DoS/DDoS attacks stop genuine clients from having access to legitimate services such as web sites. For instance, when an attacker overloads a server through several requests, the server will consume its resources and deny its services to valid users. In general, there is a great diversity of DoS/DDoS attacks; in particular, this project focuses on DoS/DDoS attacks that deplete the server's resources such as network bandwidth, memory, and CPU. The CERT Coordinator Center web site presents a complete list of resources which can be depleted by a DoS/DDoS attack [4]. We consider that the three previously mentioned resources are the most relevant for our research. Even though DoS/DDoS attacks have been known for a long time, it is just recently that they have become widely known to the general public. In October 2002, the Internet root servers, the Domain Name Server (DNS), were victims of a DDoS attack [7]. In one month period, August 2003, Microsoft's main web site suffered two DDoS attacks [16]. The list of types of DoS/DDoS attacks' victims can extend for thousand pages; this is why preventing DoS/DDoS has become very important. A DDoS attack differs from a DoS attack in that several machines are impersonated and used by the attacker to send a great number of requests to the targeted server (without the legitimate machines' users knowing that their computers are being employed to mount an attack). In addition, it should be noted that when there are opportunities to mount a DoS attack, there are also opportunities to mount a DDoS attack, however, it is worth-mentioning here that a DoS attact prevention mechanism may not necessarily work for a DDoS attack.

For ease of understanding, the DoS attack will be explained by using the TCP SYN flooding attack which is a classic example of an attack targeted to consume the server's memory. A TCP connection begins with a three-way handshake: First, the connection starts with the SYN message sent by the client which is requesting the server's services; second, the server replies with a SYN-ACK message acknowledging that a SYN message was received; and third, the client then completes the TCP connection by sending an acknowledgement message. It is after the third message that the data communication exchange starts. When the server sends its SYN-ACK message, a

slot on the server's memory is also reserved for the TCP connection being requested. An attacker, desiring to consume the server's memory, only needs to send several concurrent SYN messages. However, no final ACK for those SYN messages would be sent. The server, as usual, will reserve some memory resources for each of the connections being requested; though, as the attacker does not complete any of the connections, the server will exhaust its memory resources [5]. In a DoS attack, these requests most likely come from the attacker's machine (frequently using a spoofed IP address); on the other hand, in a DDoS attack, the bogus requests come from the impersonated clients (using their real IP addresses).

It has been mentioned that DoS resource depletion attacks mainly succeed because the attacks occur before the authentication process is completed [11]. Therefore, many authentication protocols have been proposed to prevent the DoS attacks. Nonetheless, most authentication protocols require that the server commits its CPU resources by computing costly encryption/decryption operations. Attackers then only need to start simultaneous authentication processes and, as in the TCP SYN flooding attack, abandon the process at a given point. In general, network security should satisfy five requirements: Integrity, authorization, non-repudiation, secrecy, and authentication. From these elements, not only does the need for authentication open opportunities for the DoS attacks, but also the need for communicating in secrecy raises other possibilities for successfully mounting DoS connection depletion attacks.

Secrecy is concerned with privacy which means that, on a given communication channel, only legitimate participants should be able to access and understand the information being transmitted. Secrecy is mainly accomplished by encryption techniques which could involve costly computational operations. Due to the computational costs of cryptography techniques a DoS/DDoS attack can be easily mounted by sending bogus encrypted messages and forcing servers to compromise their resources decrypting these messages.

Authentication, on the other hand, is concerned with confirming the identities of the entities which are participating in a given communication. The goal of an authentication protocol is to establish the legitimacy of the different parties involved in the communication. In other words, authentication means to verify the fact that information or messages really come from the sender and not from another source (an attacker). Authentication can be accomplished at different layers in the protocol stack; depending on the application, one protocol may be preferred over the others. Due to the increasing amount of DoS attacks, authentication protocols' designers are more concerned with the prevention of this type of attack at the designing stage, while this concern was previously left to the implementation stage [2]. However, authentication protocols usually employ cryptography which is time and resource consuming. Consequently, the use of cryptography makes authentication protocols vulnerable to those attacks which exploit this fact. The DoS attacks are an example of this attack. It is also acknowledged that public key algorithms are considerably slower than symmetric key algorithms; therefore, authentication protocols which employ public key methods are even more vulnerable to resource consumption attacks. In the case of DDoS attacks, authentication protocols are even less effective because attackers use the real IP addresses of the impersonated machines, and most authentication protocols are based on validating the IP addresses of the parties involved in a given communication.

The DoS/DDoS resource depletion attacks are as difficult to prevent as they are easy to mount. A relatively recent trend to defeat DoS attacks is based on the broad principle that a client, requesting a given service, should first compromise its resources before the server's resources are compromised [1]. These techniques are called client puzzles. A number of practical implementations of client puzzle protocols reported positive results in preventing DoS attacks [6], [17]. This paper focuses on analyzing current client puzzle protocols for counteracting the DoS/DDoS resource depletion attacks. In particular, the analysis is aimed to identify weaknesses of current client puzzle protocol proposals for preventing the DDoS. Based on the puzzle generation and verification processes, and focusing mainly on forestalling DDoS attacks, this paper classifies and analyzes current proposals of client puzzle protocols and not only reveals and analyzes their limitations with regards to the prevention of DDoS attacks, but also outlines a general framework for addressing the identified limitations.

The remainder of this paper will be organized as follows. Section 2 introduces the client puzzle approaches. Section 3 analyzes current client puzzle solutions and proposes extensions to improve today's client puzzle protocols. Section 4 presents the conclusions.

2. Defeating DoS/DDoS Through Client Puzzles

The DoS/DDoS resource depletion attacks are aimed to exhaust the servers' resources. A key factor for the success of DoS/DDoS attacks is the fact that, in open networks such as the Internet, the authenticity of the entities requesting services is uncertain. Therefore, servers compromise their resources processing bogus messages. By the time the bogus messages are identified, the attacker has most likely already succeeded. It should be concluded that any solution aimed to defeat the DoS/DDoS resource depletion attacks must avoid compromising the server' resources

before the clients have proven their legitimate interest of obtaining the server's services. It is important to note that we have referred to the need of clients proving their genuine interest of obtaining the requested services and not to the need of the clients providing proof of their legitimate identities.

Verifying clients' identities involves the use of authentication protocols that would force the servers to employ their computational power in different stages of the protocol. Whereas, asking clients to confirm their legitimate interest for receiving the requested services can be done by compromising the clients' resources before the servers' resources are committed. Client puzzles follow the principle that the clients must commit their resources before a server does [1]. A puzzle is a cryptographic problem that the servers might ask the clients to solve in order to provide the requested service. Even though client puzzles are not a new concept, the use of client puzzles to defeat the DoS/DDoS attacks is a relatively new idea.

The idea of puzzles was introduced as early as 1978, and Merkle was the first to incorporate the concept of cryptographic puzzles into authentication protocols. Merkle's proposal was intended for key exchange in the presence of an eavesdropper [12]. Merkle introduced the idea that, in a given communication, one legitimate participant sends several cryptographic problems that would be broken by the other participant. The security against an eavesdropper is based on the fact that the attacker is forced to solve all the puzzles whereas the legitimate participant only needs to choose and solve one puzzle. Although Merkle did not name his technique "client puzzles", generally speaking, current proposals of client puzzles apply some of Merkle's ideas. Nevertheless, at the time of this writing, no practical implementation of Merkle's suggestion was found.

A more recent trend of the applications of client puzzles involves their use to defeat DoS resource depletion types of attacks [1, 3, 6, 8, 9, 10, 14, 17]. Client puzzle proposals could be divided by the manner in which puzzles are built and verified. Another possible classification could be based on what the client puzzle protocol is trying to protect (for example, an authentication protocol). Since we are interested in addressing whether or not client puzzle protocols can thwart DDoS attacks, the first classification method is the most appropriate and will be used. Current client puzzle proposals can be divided into two groups based on:

- Finding the missing bits of a pre-image of a hash function whose output is given. In other words, given part of z1 and all z2, find the missing bits of z1 such that z2 = hash (z1), where z1 = hash (connection dependant parameters). Clients must reverse the hash function by applying brute-force search. We refer to this method as multiple-hash puzzles.
- Finding a pre-image of a hash function fulfilling a given number of zeros on the output, and knowing part of its pre-image as well. In this case, the output of the hash is not relevant as far as its first "k" bits are zero. More precisely, the puzzle is based on finding X such that hash (Client Id | Server Nonce | Client Nonce | X) = $0_1 \ldots 0_k Y$; where Y can take on any value. We call this technique single-hash puzzles.

Table 1 illustrates the above described methods. As it can be seen on Table 1, both methods are based on reversing only one hash function; however, in the multiple-hash puzzle technique, two hash functions are required to build puzzles, and this is why we called this method multiple-hash puzzles. To our knowledge, Juels and Brainard [9] were the first in proposing the multiple-hash puzzle method; then, Lee and Fung [10] proposed an improved version of this technique. The first proposal using the single-hash puzzle method is from Aura, Nikander, and Leiwo [1]; Dean and Stubblefield [6] reported a practical implementation of this technique; Moskowitz *et al.* [14] proposed the Host Identity Protocol (HIP), an Internet draft, which uses single-hash puzzles as its first phase; and Wang and Reiter [17] reported another practical implementation of the single-hash puzzles. The client puzzle techniques will be explained by the previous classification, and the differences between proposals will be pointed out when required. Nevertheless, two aspects are shared by all the authors examined: The message flow and the puzzles' characteristics are greatly similar to one another.

Table 1. Classifications of puzzle mechanisms.

	Multiple-Hash Puzzles ([9, 10])		**Single-Hash Puzzles ([1, 6, 14, 17])**	
	Puzzle Generation	**Puzzle Solution**	**Puzzle Generation**	**Puzzle Solution**
Server	z1 = hash (Client Id | Server Secret | Timestamp) z2 = hash (z1)		Server Nonce	
Client		Receive: z1 < k + 1, L>, z2, t, k Find: z1 <1, k> such that hash (z1) = z2		Find X such that hash (Client Id | Server Nonce | Client Nonce | X) = $0_1 \ldots 0_k Y$

In general, the client puzzle message flow is as follows: First, the client sends a message asking for the provision of a given service. Second, if the server is under attack, a puzzle must be sent to the client. Third,

the client sends the puzzle's solution. After verifying that the submitted puzzle's solution is correct, the server can commit its resources when providing the requested service. It should be noted that in the second message no puzzle will be included if the server is not under attack.

Before going into detail of client puzzle solutions, the puzzles' characteristics will be presented in the next section.

2.1. Puzzle Characteristics

To prevent DoS attacks, puzzles should have the following characteristics:

- The computational costs employed by the server in generating and verifying the puzzles must be significantly less expensive than the computational costs employed by the client in solving the puzzles.
- The puzzle difficulty, which depends on the server's resources availability, should be easily and dynamically adjusted during attacks.
- Clients have a limited amount of time to solve puzzles.
- Pre-computing puzzle solutions should be unfeasible.
- Having solved previous puzzles does not aid in solving new given puzzles.
- Before a correct puzzle solution is submitted, the server does not keep a record of the connection's state.

In addition, Feng [8] suggests three more factors to be taken into account when implementing client puzzles. First, the server's ability for generating puzzles must not be able to be flooded by the attacker; in other words, the server should be able to handle several concurrent requests from clients. Second, when a puzzle is delivered to a given client, the client must not be able to circumvent the puzzle mechanism. Third, the concept of fairness is introduced which consists of making puzzles' difficulty dependable on the clients' hardware. More precisely, the author suggests that a "thin client" (cell-phone, PDA, etc.) should be given less difficult puzzles to solve. Nevertheless, we believe that this idea of puzzle fairness should be carefully handled otherwise it could open opportunities for DoS/DDoS attacks.

It should be noted that Bencsáth *et al.* present an analysis, using game theory of client puzzle protocols [3]. They first present an abstract model of client puzzle approach which consists of several steps to carry out before the server can compromise its resources. In other words, this abstract description is what we have referred to as the client puzzle's message flow. The authors suggest two different puzzle generation/solution methods, but there is no detail on how to choose between these two levels of difficulty. The most remarkable difference between these authors' proposal and the others is that Bencsáth *et al*. propose to implement puzzles with different computational costs for the server and obviously for the client as well. To choose the puzzle difficulty, the server needs to keep measures of attacks. Evidently, this differentiation makes the process of generating and verifying puzzles more complex because it implies combining different solutions, and the more complex the system is, the more chances to open opportunities for DoS/DDoS attacks. The authors point out that they are planning to extend the Internet Key Exchange (IKE) protocol using the proposed puzzles. Nonetheless, at the time of this writing, no further work was reported.

3. Client Puzzles Analysis

Analyzing client puzzle protocols is still a difficult task due to the fact that these protocols have not been widely implemented yet. Therefore, most of these techniques' flaws have been pointed out by the authors themselves, or for others developing their proposals. Before some of the practical implementations were reported, one of the major weaknesses of using client puzzles was the interoperability with current communication protocols. However, as Dean and Stubblefield [6], and Wang and Reiter's [17] solutions showed, it is possible to achieve some degree of interoperability. In the first case, the authors' solution remains compatible with all clients when the server is not under attack. When an attack is running, only clients able to solve puzzles will be given the requested service. Ideally, all legitimate clients should receive the server's services, but if no client puzzle protocol were implemented, any client would receive the server's services under attack. In the case of Wang and Reiter, the authors' implementation not only remains full compatible with the existing TCP protocol, but also legitimate clients that do not have the ability to solve puzzles might receive the server's services. Both of these practical implementations showed that it is possible to overcome the interoperability concerns. In addition, if future client puzzle protocols' implementations keep achieving more positive results, there will be more incentive to develop any required client-side-software for solving puzzles.

Table 2 shows the steps involved in the processes of puzzle generation, puzzle solution, and puzzle verification for current client puzzle protocols. Comparing client puzzle protocols based on the way puzzles are generated and verified rises the following concerns (see Table 2).

The single-hash puzzle based protocols use the simplest puzzles generation and verification processes but there is a trade-off: Puzzles could not have solutions [10, 14]. On the one hand, due to the simplicity of puzzle generation, the same "puzzle" could be given to several different clients since the

puzzle solution will depend on the clients' identities. It should be recalled that in the Aura proposal the server periodically generates a nonce and the puzzle consists of the nonce plus some other client-dependant parameters. On the other hand, as puzzles are not verified during their generation, a given puzzle could not have solution. In other words, there is not guarantee that a given client will find a pre-image that will provide the required number of zeros on the output of the hash function. Overcoming this problem is a challenging task because puzzle generation and verification should be kept fast to avoid resource depletion attacks; thus, the puzzles' solutions could not be verified during the generation phase; it is also not possible to pre-compute puzzles since they could be pre-computed by attackers as well. In addition, most of the pre-computation security seems to rely on the server nonce, if the mechanism for generating the nonces is not carefully designed, the client puzzle protocol could be compromised.

The multiple-hash puzzles based proposals do not have the problem of puzzles not having solutions because the generation and verification processes are more complex. At the time of this writing, no practical implementation was found and, therefore, the costs associated to the complexity of the suggested method have not been measured. In particular, we consider that the Lee and Fung's proposal have some weaknesses mainly because the authors integrate the client puzzle protocol with their authentication protocol. The problem is that to shorten the run of the proposed authentication protocol, some information needed for the authentication process is included in the third message. However, if the submitted puzzle solution is not correct, the message will be discarded but more bandwidth will have been consumed for no reason. Depending on the type of network, the extra bandwidth consumption would be negligible, but in a high traffic network this solution might not be convenient. In fact, attackers could alternately submit correct and wrong solutions and combine other techniques for attaining the exhaustion of the network bandwidth. Moreover, for the proposed authentication protocol even when the server does not need to send a puzzle (k = 0), the bit-string z1 should be built because z1 will be used later on in the protocol for validating messages freshness and preventing replay attacks. Therefore, attackers could take advantage of this extra load for exhausting the computational power of the server. It should be recalled that at the beginning of a given attack, the puzzles will be less difficult to solve, and by combining different techniques the attacker could succeed. For example, for k equal zero, the server needs to do some extra work and the attacker will easily get to the step where the server needs to compromise its computational power for the authentication process; for small values of k, the attacker can allow clients to compute valid answers and sends also some wrong answers to force the server to do some extra verifications; for higher values of k, some pre-computed solutions will be submitted (whenever possible) and wrong solutions will be submitted as well for forcing the server to do some extra work. Combining all of these techniques might allow the attacker to succeed. For these reasons, it is highly recommended to implement the client puzzle protocol as an initial phase, where all the required computations should only be related to the client puzzle protocol.

As can be seen in Table 2, all puzzle mechanisms need to determine a value for k, the puzzle difficulty. However, determining the values of the puzzle difficulties when the server is under attack has not been completely addressed. Paradoxically, this open issue comes from one of the best properties of client puzzles: Its optional character. This is an important point because the puzzle difficulty will depend on the server's resources availability. If the procedure to determine this availability is not carefully designed, the hardness of puzzles could be wrongly set. Consequently, legitimate clients could experience a harmful degradation of services, or even worst more attackers could succeed.

Table 2. Comparison of client puzzle protocols.

	Puzzle	Puzzle Generation	Puzzle Solution	Puzzle Verification
Multiple-Hash	Find z1 <1,k> such that hash (z1< 1, k> | z1 <k + 1, L >) = z2	Determine k 2 hash	2^k hash	2 hash
Single-Hash	Find X such that hash (Client Id | Server Nonce | Client Nonce | X) =0 $_1$…0$_k$Y	Determine k Generate Nonce	Generate Nonce 2^k hash	1 hash
	Find x <1,k> such that MD5 (x < 1, k > | x <k + 1, L >) = MD5 (x)	Determine k Compute 1 MD5	2^k MD5	1 MD5
	Find X such that SHA-1(Server Nonce | Client Id | Server Id | X) = 0$_1$…0$_k$Y	Determine k Generate Nonce	$2^{(k+2)}$ SHA-1	1 SHA-1

Notes:
1. The term "hash" refers to any collision-free hash function such as MD5, and SHA-1.
2. It should be noted that SHA-1 computation is slower than MD5 computation [16].

Concerning the pre-computation attacks, some practical tests are required for measuring not only how

feasible pre-computation is, but also what will be the required storage space for saving the pre-computed puzzle solutions. Precisely, pre-computing solutions have the trade-off of saving the solutions. However, attackers might only pre-compute puzzle solutions for high values of k to reduce the amount of pre-computation as well as the amount of information to be saved. One possible method for counteracting the pre-computation of puzzle solutions could be established by incorporating some alternate changes in the puzzle generation process. Specifically, when a new server nonce is generated, the hash function used to compute the puzzles could be also changed. In particular, the server could alternately use MD5 and SHA-1; even though computing SHA-1 is slower than MD5, the server's workload will not be highly affected whereas attackers will be forced to pre-compute twice the amount of information.

3.1. Defeating Client Puzzle Protocols

Even though all the practical implementations have accounted for positive results in counteracting the DoS attack by using client puzzles, none of the implementations have accounted any results for the DDoS which has become of great concern of study. In a DDoS attack, the attacker impersonates several clients (called zombies) and uses their real IP addresses when mounting the attack. Specifically, the attacker can send and receive messages from the zombies, and the legitimate zombies' owners are not aware that their clients are being used by the attacker. It has been mentioned that the tools employed for DDoS are designed in a way that the zombies' performances are not affected and, thus, the legitimate users do not notice any change in their clients' behaviours [13, 17]. The strength of client puzzles to defeat a DoS attack is based on the fact that the attacker will be asked for solving one puzzle for each service request, and solving all the puzzles will exhaust the attacker's resources. Nevertheless, in a DDoS, the attacker will use every zombie to compute its own puzzle, and only when puzzles difficulty get too hard, the impersonated computers' operations will be interrupted. Thus, interrupting zombies' operations also means that legitimate clients' operations will be interrupted since the server can not distinguish among legitimate clients and impersonated clients (zombies). The worst situation will be when the attack is too severe that the server sends out puzzles that would not be solved feasibly. If it is not possible for zombies to solve the puzzles, it would not be possible for legitimate clients to solve the puzzles either, and as a result, legitimate clients will receive a denial of service, which will be caused by the client puzzle mechanism.

In addition, for attackers there is one possible way to counteract the above mentioned problem, the attackers could divide all the zombies into different groups and use each of the groups in an alternating manner. By using this technique, the attacker is decreasing the load for each zombie, or at least, the noticeable computational load will be delayed. To make things worse, by combining this *switching-load* method with some pre-computed solutions, the attacker is most likely to succeed either by exhausting the server's resources or by forcing the server to send out puzzles impossible to solve. Mirkovic and Reiher describe and classify a similar technique as a variable agent set [13]; the authors explain that an attacker using the variable agent set technique is aiming to avoid or delay the attack detection due to the high rate of packets coming from the same sources. Basically, the attacker switches the set of agents being used in the attack such that the overall attack rate is constant, but the packets are coming from diverse impersonated clients which form part of the attack at different instants. In the *switching-load* method, the attacker is aiming to avoid or delay attack detection due to the performance decrease of the impersonated clients.

The fact that it is not possible for the server to distinguish between a legitimate client and an impersonated client poses a great challenge in client puzzle protocols as a countermeasure against the DDoS attacks. On the one hand, puzzles cannot be too hard because legitimate clients will experience a harmful degradation on the services. On the other hand, puzzles cannot be so simple that attackers can easily solve the puzzles and compromise the server's resources. We propose a solution based on the general principle that under attack legitimate clients should be willing to experience some degradation in their performance in order to obtain the requested service since obtaining the service is preferable than obtaining a denial of service from the server. Our proposal is based on including a puzzle-solution request in different states of a given connection such that the computational load for solving the puzzles will be noted but the clients' operations will not be totally interrupted. For example, if a client is repeatedly requesting the server's resources, at any time that the server needs to perform a costly computational operation, the server sends a new and harder puzzle to solve even if the client has already submitted a correct puzzle's solution earlier in the protocol run. The value of k (puzzle's difficulty) for the new required puzzle could be set higher than the actual value (to avoid saving information related to each connection). Where to include a client puzzle protocol could be analyzed by using the framework proposed by Meadows [11]. We are aware that the major drawback of this method is that legitimate clients are unjustly penalized; nevertheless, the main goal of continuously providing the services will be accomplished. In addition, we consider that *client discrimination* should be added to the puzzle characteristics list mentioned in section 2.1.

4. Conclusions

By analyzing current client puzzle protocol proposals, their weaknesses and strengths have been pointed out. Practical implementations not only reported positive achievements, but also these implementations illustrated that interoperability with current protocols is attainable up to a certain degree. The effectiveness of client puzzles against the DDoS might be the greatest unresolved issue, and perhaps, more practical implementations are required for addressing this concern as well as for identifying new challenges. However, as explained before, discriminating between legitimate clients and impersonated clients is the biggest challenge that client puzzle protocols face with regards to forestall the DDoS attacks. We consider that *client discrimination* should be added to the list of characteristics for puzzle mentioned in section 2.1. In addition, this study has raised three possible extensions for client puzzle protocols: First, it is highly recommended to implement the client puzzle protocol as an initial phase, where all the required computations should only be related to the client puzzle protocol; second, to make pre-computation attacks even more difficult, the hash function used to compute puzzles could be alternately changed; and third, to counteract the *switching-load* technique that attackers might use to defeat the client puzzles during a DDoS attack, the server might ask clients to solve more than one puzzle during the protocol run. Nevertheless, further work is required for modelling our enhanced client puzzle protocol.

References

[1] Aura T., Nikander P., and Leiwo J., "DOS-Resistant Authentication with Client Puzzles," *in Proceeding of the Cambridge Security Protocols Workshop'2000*, LNCS, Springer-Verlag, Cambridge, UK, April 2000.

[2] Aura T., Nikander P., and Leiwo J., "Towards Network Denial of Service Resistant Protocols," *in Proceeding of the 15Th International Information Security Conference (IFIP/SEC' 2000)*, Kluwer, Beijing, China, August 2000.

[3] Bencsáth B., Vajda I., and Buttyán L., "A Game Based Analysis of the Client Puzzle Approach to Defend Against DoS Attacks," *in Proceedings of IEEE Conference on Software, Telecommunications and Computer Networks (SoftCom'2003)*, Split, Dubrovnik, Ancona, Venice, October 2003.

[4] CERT Coordination Center, "Denial of Service Attacks," *Tech Tips*, June 2001.

[5] CERT Coordination Center, "TCP SYN Flooding and IP Spoofing Attacks," *Tech Tips*, November 2000.

[6] Dean D. and Stubblefield A., "Using Client Puzzles to Protect TLS," *in Proceedings of the l0th USENIX Security Symposium*, August 2001.

[7] Duffy Marsan C., and Garretson C., "Net Security Gets Root-Level Boost," *Network World Fusion*, October 2003.

[8] Feng W., "The Case for TCP/IP Puzzles," *in Proceedings of ACM SIGCOMM 2003 Workshops*, August 2003.

[9] Juels A. and Brainard J., "Client Puzzles: A Cryptographic Defense Against Connection Depletion Attacks," *in Proceedings of Networks and Distributed Security Systems (NDSS'99)*, pp. 151-165, 1999.

[10] Lee M. C. and Fung C. K., "A Public-Key Based Authentication and Key Establishment Protocol Coupled with a Client Puzzle," *Journal of the American Society for Information Science and Technology*, vol. 54, no. 9, pp. 810-823, 2003.

[11] Meadows C., "A Formal Framework and Evaluation Method for Network Denial of Service," *in Proceedings of the 12th IEEE Computer Security Foundations Workshop*, Mordano, Italy, 1999.

[12] Merkle R. C., "Secure Communications Over Insecure Channels," *Communications of the ACM*, vol. 21, no. 4, pp. 294-299, April 1978.

[13] Mirkovic J., and Reiher P. "A Taxonomy of DDoS Attack and DDoS Defense Mechanisms," *in ACM SIGCOMM Computer Communications Review*, vol. 34, no. 2, April 2004.

[14] Moskowitz R., Nikander P., Jokela P., and Henderson T., "Host Identity Protocol," Network Working Group Internet Draft, available at: http://www.ietf.org/internet-drafts/draft-ietf-hip-base-01.txt, 2004.

[15] Roberts, P., "Microsoft.com Falls to DoS Attack," *Network World Fusion*, available at: http://www.nwfusion.com/news/2003/0815microfalls.html, August 2003.

[16] Stallings and William, *Cryptography and Network Security, Principles and Practices*, Upper Saddle River, NJ, Prentice Hall, 2003.

[17] Wang X., and Reiter M. K., "Defending Against Denial-Of-Service Attacks with Puzzle Auctions," *in Proceedings of the 2003 IEEE Symposium on Security and Privacy*, May 2003.

Vicky Laurens is a Master's candidate at the Univeristy of Ottawa. She holds a certificate in Internet technologies from the University of Ottawa and a Bachelors of engineering degree in electronics from Simon Bolivar University in Venezuela. She has 7 years of experience in television engineering working with HBO Latin America Group. She is also a research volunteer at the

Canadian Internet Policy and Public Interest Clinic (CIPPIC). Her research interests include computer and Internet security.

Abdulmotaleb El Saddik is an associate professor at the School of Information Technology and Engineering (SITE) at the University of Ottawa, and the director of the Multimedia Communications Research Laboratory (MCRLab). He has authored and co-authored two books and more than 80 publications in the area of software engineering development of multimedia artefacts and collaborative virtual environments. He is a senior member of IEEE and the recent winner of the prestigious Canadian Premier's Research Excellence Awards (PREA).

Amiya Nayak is an associate professor at the School of Information Technology and Engineering (SITE) at the University of Ottawa since 2002. He received his BSc in Mathematics from University of Waterloo in 1981, and PhD in computer engineering from Carleton University in 1991. He has over 16 years of industrial experience in software engineering, real-time software development, simulation and system level performance analyses, avionics and navigation systems, telecommunication protocols, network traffic analysis. His research interests are in the area of fault tolerance, distributed systems, and mobile ad hoc networks, with over 70 publications in refereed journals and conference proceedings.