2-2015

A Cloud-Based Secure Authentication (CSA) Protocol Suite for Defense against Denial of Service (DoS) Attacks

Marwan Darwish Mr.
Western University, mdarwis3@uwoc.a

Abdelkader Ouda
University of Western Ontario

Luiz Fernando Capretz
University of Western Ontario, lcapretz@uwo.ca

Follow this and additional works at: https://ir.lib.uwo.ca/electricalpub

Part of the Computer Engineering Commons, Electrical and Computer Engineering Commons, and the OS and Networks Commons

Citation of this paper:
https://ir.lib.uwo.ca/electricalpub/66
A cloud-based secure authentication (CSA) protocol suite for defense against Denial of Service (DoS) attacks

Marwan Darwish*, Abdelkader Ouda*, Luiz Fernando Capretz
Department of Electrical and Computer Engineering, The University of Western Ontario, London, Canada

Abstract
Cloud-based services have become part of our day-to-day software solutions. The identity authentication process is considered to be the main gateway to these services. As such, these gates have become increasingly susceptible to aggressive attackers, who may use Denial of Service (DoS) attacks to close these gates permanently. There are a number of authentication protocols that are strong enough to verify identities and protect traditional networked applications. However, these authentication protocols may themselves introduce DoS risks when used in cloud-based applications. This risk introduction is due to the utilization of a heavy verification process that may consume the cloud's resources and disable the application service. In this work, we propose a novel cloud-based authentication protocol suite that not only is aware of the internal DoS threats but is also capable of defending against external DoS attackers. The proposed solution uses a multilevel adaptive technique to dictate the efforts of the protocol participants. This technique is capable of identifying a legitimate user's requests and placing them at the front of the authentication process queue. The authentication process was designed in such a way that the cloud-based servers become footprint-free and completely aware of the risks of any DoS attack.

1. Introduction and related works
Cloud computing is the utilization of hardware and software to provide services to end users over a network, such as the Internet. Cloud computing includes a set of virtual machines that simulate physical computers and provide services, such as operating systems and applications. However, configuring the virtualization within a cloud computing environment is critical when deploying a cloud computing system. A cloud computing structure relies on three service layers: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) (Fig. 1) (Mell and Grance, 2011). IaaS provides users with access to physical resources, networks, bandwidth and storage. PaaS builds on IaaS and provides end users access to the operating systems and platforms necessary to build and develop applications, such as databases. SaaS provides end users with access to software applications (Mell and Grance, 2011).

DoS attacks represent major security risks in a cloud computing environment, where the resources are shared by many users. A DoS attack targets the resources or services in an attempt to render them unavailable by flooding the system with heavy amounts of artificial traffic. Dealing with DoS...
attacks at all layers of cloud systems is a major challenge due to the difficulty of distinguishing the attackers’ requests from legitimate user requests, particularly when the data are transferred between the layers of the cloud computing systems (Modi et al., 2013). Therefore, detecting a DoS attack in its early stage, in the upper layer (SaaS), is a significant approach to avoid the destruction caused by DoS attacks on the other layers. However, all service requests for SaaS must be authenticated to be approved.

Many authentication protocols can be used in the SaaS layer. The OAuth protocol (Hardt, 2012) is currently a widely used authentication protocol that controls the access of third-party applications to an HTTP service. In OAuth, the resource owner can allow a third-party client to access the resources through the owner. For example, a user as a photo owner (resource owner) can grant permission to a printing service (client) to access the user’s photos. The photos are stored on a photo exchange server (resource server). Rather than sharing the user’s credential with the printing service, the user is authenticated to a server that is trusted by the photo exchange server (authorization server), which then issues a credential (such as an access token) to access the resources.

There are limitations when the owner shares credentials, such as a username and password, with the third-party client to access restricted resources. The first limitation is that the access information includes the password, which is most likely stored by a third-party client as clear-text for future access. The second limitation is that the server should only use a password as an authentication method. The third limitation is that the resource’s owner cannot limit the access of a third-party client and also cannot control the duration of the access. Finally, if the password is accessible in a third-party client, all of the resources will be accessible, as well. Therefore, OAuth allows a third-party client to access the resources of the server with privileges and rules without using the resource owners’ access information. The process of the protocol, as shown in Fig. 2, operates in the following manner:

1. The protocol process starts when the resource owner receives an authorization request from the client.
2. The resource owner sends back the authorization grant to the client. The client’s authorization request determines the type of grant. Examples of the different types of grant are as follows:
   - **Authorization code grants** are given to the client by the resource owner after the resource owner has been authorized by the authorization server. With such a code, the client does not require the resource owner’s credentials, and it is a secure grant type.
   - **Implicit grants** depend on browser implementation using a scripting language, such that the access token is issued directly to the client. An implicit grant minimizes the flow process of the protocol but also leads to security issues.
   - **Resource owner password credentials grants** use the resource owner’s credentials (username and password) to issue an access token to the client. This type of grant can be used when the client is highly trusted by the resource owner.
   - **Client credentials grants** can be used for a limited scope of access to protected resources on the server. This type of grant is used when the client is a resource owner or when the client previously had privileges to access the protected resource.
3. The client sends an authorization grant and the authentication to an authorization server to obtain the access token.
4. The access token will be provided to the client once the client is authenticated and the authorization grant is validated by the authorization server. In this case, the access token replaces the typical authentication, such as username and password, and is also recognized by the resource server. The access token can be used in different methods based on the security requirements of the server, such as with different types of cryptography. When the token is expired or becomes invalid, the token can be refreshed as an optional process by sending the authorization grant to an authorization server, whereupon the client receives a request with the access token and refresh token.
5. The client sends the access token to request restricted resources from the resource server.
6. The server will respond to the request when validating the access token.

However, any insecure implementation of the OAuth protocol can lead to the possibility of a DoS attack.

Many authentication protocols have been proposed for the SaaS layer, but they would be unaware of a DoS attack. Yassin et al. (2013) proposed an authentication process based on a one-time password (OTP) with mutual authentication of the user and the cloud server. Yassin's authentication schemas defended against the possibility of a replay attack but not against a DoS attack. Various cloud-based authentication protocols for DoS prevention have been proposed, such as those by Choudhury et al. (2011), Hwang et al. (2010), Jaidhar (2012), and Tsaur et al. (2012), but they use a smart card reader for the authentication process. Furthermore, the Yassin et al. (2012) schema also recommends the use of an extra physical device, such as a fingerprint scanner.

On their own, the authentication protocols can lead to vulnerability to a DoS attack. Therefore, it is necessary and significant to verify the DoS-resistance in every process of the protocol.
authentication protocol. For example, on the one hand, verifying a large number of signed messages via the server consumes the resources of the server to a significant degree, particularly when the attacker sends a massive number of forged signed messages. On the other hand, sending a typical client credential with each request in the authentication protocol will force the server to verify these requests based on the stored information at the server. As a consequence, the server resources will be exhausted when dealing with a large number of requests.

An example of authentication protocols that can introduce internal DoS risks on their own is shown in Fig. 3. The goal of this protocol is to authenticate both the client and the server to each other. This protocol uses the ephemeral Diffie-Hellman key-exchange (Diffie and Hellman, 1976), where a, b, p, and g are the values of Diffie-Hellman, as shown in Fig. 3. In this protocol, once the server receives a request from a client, the server will begin generating the secret value b. Subsequently, the server will compute the exponential value $g^b \mod p$. Moreover, the server will encrypt the nonce of the client and the exponential value via the client public key. Finally, the server will digitally sign the encrypted message. All of these processes will be executed by the server, which consumes a great deal of resources without determining whether the request is legitimate. This mutual authentication, which is vulnerable to DoS attack, is similar to the two-way authentication version of the Transport Layer Security (TLS) protocol (Dierks, 1999). Another example of a protocol that introduces DoS risk on its own is Kim et al.’s protocol (Kim et al., 2009), which aims to securely authenticate the key exchange between participants. In this protocol, once the server receives the first message, the server will begin computing an exponential value and generate the key, and, as such, the server resources can become exhausted by the initial requests.

An example of an authentication protocol that would be aware of a DoS attack in the traditional network is a Host Identity Protocol (HIP) (Moskowitz et al., 2008). However, this protocol cannot be implemented in the application layer because it is based on the host identity on the network layers in the OSI reference model, and it needs to be configured and controlled at an operating system level. Moreover, any authentication protocol that is based on IP address verification, such as the IPSec protocol, makes it difficult to hide the identity of the participants.

One approach to investigate the strength of the authentication protocol against DoS attacks is the cost-based model approach. This approach was proposed by Meadows (2001) mainly to demonstrate the effectiveness of the protocols in preventing DoS attacks. The Meadows approach relies on the idea that one of the participants (the requester or the responder) will get computationally exhausted first. As such, the computation costs for both the requester (client) and the responder (server) need to be determined. The total computation cost of the requester is the total estimated cost of each
operation involved in the authentication process on the requester’s side during the life of the authentication protocol. However, the total computation cost of the responder is the total estimated cost of each operation during the authentication process until the requester is determined to be either a legitimate requester or attacker.

Meadows (2003) proposed the following categories to classify the computation cost: expensive, medium, and inexpensive. The expensive computation class includes exponential operations and signature operations along with its verifications. The medium computation class includes encryption and decryption operations. The inexpensive computation class includes all other operations not mentioned above.

The rest of this paper is structured as follows. Section 2 explains the research methodology of this work. Section 3 describes the CSA protocol against DoS attacks in the SaaS layer. Section 4 analyzes the protocol. Section 5 discusses the proposed work. Finally, Section 6 briefly summarizes the work.

2. Research methodology

SaaS layer cloud-based applications are likely to be vulnerable to DoS attacks. This is because the SaaS layer is considered to be the most popular type of cloud resource (Rogers, 2012). We propose a cloud-based DoS-resistance protocol suite that securely authenticates cloud users. This protocol is designed such that the cloud server is required to do lightweight computation work without the need to store any data during the authentication process. In addition, it requires the cloud user to do a process that is computationally expensive.

To realize the Meadows cost-based approach, this protocol considers a “subset sum” problem as a “client puzzles” technique (Juels and Brainard, 1999). A “subset sum” problem is a kind of cryptographic knapsack problem, and it is not only a strong one-way function, but it also has a flexibility property to be adaptive (Salomaa, 1996). The complexity of the subset sum knapsack problem depends on the size of the knapsack (the total number of its items, n) and on the number of items (the subset sum size, m) involved in puzzle solution. If the number of items n is small, then an exhaustive search for the solution is practical. Also, if the number of m is small compared to n, then a solution can be found in a reasonable time. Consequently, by adjusting the values of n and m, determining the difficulty level of the knapsack problem and hence the cost-based approach can be adaptively realized.

Therefore, the proposed cloud-based DoS-resistance protocol suite has the ability to be configured such that the more sensitive the services requested are, the greater the computation cost required from the requester is. In other words, greater computation cost can be achieved by asking the requester (client) to perform an expensive operation, such as solving an expensive subset sum puzzle. The responder (cloud server) should conduct inexpensive to medium-cost operations, such as generating subset sum puzzle elements, checking the solution, or decryption operations.

The dynamic programming algorithm is our tool to solve and assess the computational cost of both participants (client and cloud server) involved in the subset sum problem. Other researcher Tritilanunt et al. (2007) implemented the L3 algorithm developed by Lenstra et al. (1982) to solve the subset sum problem. However, the L3 algorithm considers various computational solutions as expensive, but, in fact, and based on our experimental results (Section 4.2), it can be inexpensive when using the dynamic programming algorithm. In addition, we were able to experimentally determine different values of n and m that provide different levels of the subset sum solution’s complexity. This makes our protocol adaptive enough to protect different levels of sensitive services.

3. Cloud-based secure authentication (CSA) protocol suite

The CSA protocol was developed so that the total computational cost of the client’s side will be greater than the resource operations cost of the cloud-based server when they participate in the authentication process together. Table 1 shows the notations that are used in the CSA protocol suite.

CSA consists of sets of protocols. The first protocol is used for the registration process, which is an agreement process between the participants (client and cloud server) about certain shared information. Thus, the participants can use that

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>client</td>
<td>The cloud user</td>
</tr>
<tr>
<td>cloudserv</td>
<td>The cloud server/service provider</td>
</tr>
<tr>
<td>CID</td>
<td>Client ID</td>
</tr>
<tr>
<td>UET</td>
<td>Unique encrypted text; the key of the UET is known only by cloudserv</td>
</tr>
<tr>
<td>SK</td>
<td>Session key</td>
</tr>
<tr>
<td>A</td>
<td>A set of random integers of the server challenge function that is the subset sum problem</td>
</tr>
<tr>
<td>S</td>
<td>An integer to which a subset will sum if there is a solution for the server challenge function</td>
</tr>
<tr>
<td>B</td>
<td>A binary vector that states the number of binary place values that it takes to state the challenge function (i.e. the subset sum solution)</td>
</tr>
<tr>
<td>Rcloudserv</td>
<td>The nonce that is generated by cloudserv</td>
</tr>
<tr>
<td>T</td>
<td>Timestamp</td>
</tr>
<tr>
<td>MK</td>
<td>Master secret key of cloudserver</td>
</tr>
</tbody>
</table>
information during the operation of other CSA protocols. The second protocol is an adaptive-based identification protocol that works against DoS attacks. This protocol was developed based on the cost-based model approach. The third protocol is used for the authentication process, which includes all operations that occur based on the initially agreed-upon information of the previous protocols. As a result, cloudserv can confirm the identity of the client and then complete the authentication process, or it can detect and then prevent an intruder in the case of a DoS attack.

### 3.1. Registration protocol

In the CSA registration protocol, the client and cloudserv will share the required identity data to register the client into the cloudserv database. As shown in Fig. 4, the registration process begins when the client submits all of the required information to cloudserv. This information includes the first name, last name, organization name, email address and/or any other information that is required by the cloud service provider. Cloudserv will then verify the received information, store it in a database and then send a validation email message to the client to confirm the client's information. After validation, cloudserv will activate the client's account. At the same time, cloudserv will generate a Unique Encrypted Text (UET) that is encrypted by the cloudserv's master key (MK), which is known only by the cloudserv. The UET contains client information, such as the Client ID (CID), as well as any other information that will be created by cloudserv during the processes of the CSA protocol. The UET is a piece of information that will not be stored on cloudserv; rather, it will be sent to the requesting client. Once the client receives the required data from cloudserv, both client and cloudserv will agree regarding the pre-shared key. The pre-shared key will be created using a key derivation function and a shared secret. The client and cloudserv will agree upon the key derivation function and a shared secret at the end of the registration protocol, and they will be exchanged via a secure channel in a very restricted environment. This approach is very much similar to the pre-shared key agreement (PSK) used in the UMTS and WPA2 protocols (Southern et al., 2013). Consequently, the client will store the UET and pre-shared key for future authentication processes.

Even if a client is registered to the cloudserv, the client cannot access the services available through the cloudserv unless cloudserv authenticates the client. To perform the authentication protocol that is ready to defend against any internal or external DoS attacks, CSA provides an outer shield to the authentication protocol to help identify the legitimate clients from the DoS attackers. The CSA adaptive-based identification protocol is designed to provide this outer shield in the manner described in the following sub-section.

### 3.2. CSA adaptive-based identification protocol

The CSA adaptive-based identification protocol utilizes the cost-based model approach, which can be briefly re-stated as follows. Before applying the computational power of the authentication protocols on the server side, the clients are asked to prove their sincere commitment in receiving the cloudserv services. This validation of commitment can be achieved by any technique that can force the clients to utilize a significant amount of computational power, before the servers utilize them, to confirm their genuine requests. The CSA protocol considers $n = 512$ items in the subset sum puzzle problem. These items are public integer values that both parties should agree upon during the registration protocol. Based on the experimental result (see Section 4.2), obtaining vector B will force the client to become involved in finding the solution to $2^{512}$ subsets, which is a highly time and resource consuming process. The number of subsets of items is adjustable based on the required efforts to be performed by the participants. Moreover, the chosen items that are used during the summation process are determined by hashing the values of CID, MK and $R_{cloudserv}$ using SHA2-512, where MK is the master secret key of the cloudserv. Note that the result of the hash function is a 512-bit stream. Moreover, the subset of the 512-bit stream that includes a specific number of ones ($m$) represents the required vector, B, of the subset sum problem. For example, if the protocol is developed to let $m = 55$, the cloudserv will take the subset of the 512-bit stream that includes the first 55 ones. Note that this number of ones is the key that dictates the adaptability of the protocol; for example, increasing $m$ makes the process of solving the puzzle harder while also increasing the computational cost of the puzzle-solving process. The hashing process is mandatory to verify the subset summation value (S) of the client after the calculation process. Note that the cloudserv generates the vector B and hence knows the value S; however, the client will receive S, which is needed to independently re-create the vector B.

The adaptive-based identification protocol process shown in Fig. 5 functions as follows:

1. Client sends a request for service with a CID to cloudserv.
   At this point, cloudserv will block any CID that has performed three consecutive requests within a low time threshold to prevent DoS attacks. The attacker may...
attempt to launch a DoS attack by sending requests with randomly generated CID values. In this case, the cloudserv resources will be less affected than when checking each request for information from a database system because cloudserv will simply reply to each request with an S value.

2. Cloudserv will reply directly to the client by sending the puzzle element as a challenge, which is the subset summation value (S) along with a cloudserv nonce, Rcloudserv.

Cloudserv will ask the client to prove its sincere commitment regarding receiving the cloudserv services by asking for the UET as well as the puzzle solution to the (S) value. The expected solution for this challenge is the vector B.

3. Once the client performs a calculation and obtains vector B, the client will send the UET, vector B, the value of S, the Rcloudserv, the CID and the encrypted timestamp T to cloudserv for validation. Note that the notation E(T, Kpre-shared) means that the timestamp T is encrypted by the pre-shared key K.

At this point, cloudserv has all of the information required to validate the authentication requests, so cloudserv can apply the validation process to only a few operations, such as the following:

- Cloudserv will check the subset of item a_i by securely hashing (CID, MK, Rcloudserv) and comparing the result vector with the received vector B to determine whether they are similar.
- Cloudserv will check the time difference between the received encrypted timestamp T and the current time stamp to determine whether it is a reasonable time difference in which to find the solution.

If any of the two previous conditions do not apply, cloudserv will drop the request and consider it to be an attacker’s request. However, once the client request passes the two conditions, cloudserv will decrypt the UET and validate the decrypted information that contains the CID.

To complete the authentication process after the adaptive protocol determines the client as being legitimate, the CSA protocol develops an authentication protocol. The participants in the authentication protocol will agree on the session key for future interactions. In addition, they can agree on the sub-session key if they require a refresh process later.

3.3. Authentication protocol

After the validation process in the previous protocol, cloudserv will generate the Session Key (SK), which is encrypted via a pre-shared key. Moreover, cloudserv will add both the SK and T information to the UET. Consequently, cloudserv is protected against DoS attacks to the storage space because UET will never be saved in the cloudserv. Furthermore, cloudserv can apply the refreshment property of the session key for future communication by adding the SK to the UET. Therefore, the authentication protocol, as shown in Fig. 6, is performed as follows:

1. Cloudserv will send to the client the generated SK that is encrypted by the pre-shared key, along with the modified UET.
2. Cloudserv will check the subset of item a_i by securely hashing (CID, MK, Rcloudserv) and comparing the result vector with the received vector B to determine whether they are similar.
3. Cloudserv will check the time difference between the received encrypted timestamp T and the current time stamp to determine whether it is a reasonable time difference in which to find the solution.

Later, the two parties can agree regarding the sub-session keys by re-applying the processes of the authentication protocol so that the cloudserv can generate a sub-session key and add it to the UET without storing it in the cloud system.

4. Analysis of the CSA protocol suite

Assessment of the CSA protocol entails evaluation of the protocol’s efficiency against DoS attacks by applying a cost-based model approach. In addition, the evaluation process measures the computation cost when the client participates in the puzzle-solving process during the authentication process.

4.1. Validation of the CSA protocol suite via a cost-based model approach

As shown in Table 2, based on the cost-based model approach, the operation cost of the client for the CSA protocol is
categorized as expensive, particularly when the client solves the puzzle. The other operations of the client are listed within the medium or inexpensive categories. However, the maximum operation costs of cloudserv, including the precalculation and decryption operations, are within the medium category. As a result, the CSA protocol suite is an effective protocol against DoS attacks, in which the consumption cost for the requester is higher than the consumption cost for the cloud service provider during the authentication process.

4.2 Computation cost analysis of solving a subset sum problem

One experiment has been conducted to analyze the time complexity of the subset sum (knapsack) problem. The subset sum can be described briefly as follows: given a set of positive integers A of size n and a positive integer value S, does any non-empty subset of size m sum to S? For example, let A be (13, 54, 28, 73, 3, 36), n = 6, and S = 89. It is obvious that the subset (13, 73, 3), m = 3, solves the problem because their summation is equal to 89. In other words, finding a binary vector B such that \( A \times B = S \) solves the problem. In this example, B is the vector (1, 0, 0, 1, 1, 0), and hence \( A \times B = 13 + 73 + 3 \), which is 89. Note that, in our identification protocol, the vector B is generated as the output of the Hash function (see Section 3.2).

We have mentioned earlier in Section 2 that the values of n and m are key factors that play a significant role in the complexity of the subset sum problem. In this experiment, different values of n and m were chosen, for which the time complexity of the subset sum problem was analyzed. The values of n were 128, 256 and 512, while the values of m ranged from 1 to 64. The dynamic programming algorithm is used to solve the puzzle; it is coded in C# and run on a 4-core desktop computer with the Windows 8 64-bit operating system, a Core i7-4770 CPU running at 3.4 GHz, and 32 GB of RAM.

Our experiment indicates that, when \( n = 128 \) and for all values of m, the algorithm solves the puzzle in less than 8 s; when \( n = 256 \) and for all values of m, the algorithm solves the puzzle in less than 12 s; and, finally, when \( n = 512 \) and for all values of m, the algorithm solves the puzzle in less than 24 s. The detailed execution times when \( n = 512 \) are shown by the graph in Fig. 7. The graph shows that, when \( m \) is equal to 30, the puzzle is solved in approximately 10 s, and, when \( m \) was between 50 and 60, the puzzle is solved in approximately 20 s of execution time. It was noticed also that, when \( n = 512 \) and m

| Table 2 – Validation of the CSA protocol suite via a cost-based model approach. |
|--------------------------------|-------------------|--------------------------------|-------------------|
| Operation                      | Cost category     | Operation                      | Cost category     |
| Send the initial request.      | Inexpensive       | Reply directly to the request  | Inexpensive-Medium |
| Solve the puzzle until the    | Expensive         | Verify the received elements.  | Medium            |
| result is obtained. Then,     | Medium            | Decrypt the UET. Generate and  | Medium            |
| send the result and the UET    |                   | encrypt the session key (this   |                   |
| to cloudserv.                  |                   | operation occurs after         |                   |
| Decrypt the session key (this  |                   | prevention of possible DoS      |                   |
| operation occurs after the    |                   | requests).                      |                   |
| prevention of possible DoS     |                   |                                 |                   |
| requests).                     |                   |                                 |                   |
is chosen to be higher than 64, the algorithm hangs due to the full consummation of the system memory, i.e. the system resources were exhausted.

Based on these figures and the corresponding system resource consumption, we were able to classify the computation cost of solving the subset sum problem to three main categories: inexpensive, medium and expensive, as is also shown in Fig. 7.

According to the findings of the surveys conducted by Nielsen (1993), 15–20 s is an acceptable response time that keeps the user’s attention focused on a given application. Therefore, choosing \( n = 512 \) and \( m = 55 \) is a good configuration of the subset sum problem that makes the legitimate user perform expensive computation that is acceptable in terms of the system response time (as per the Nielsen study) specially with his/her initial request.

It is worth mentioning that other researchers (Tritilanunt et al., 2007) showed that the L3 algorithm that was developed by Lenstra et al. (1982) can solve the subset sum problem of \( n = 100 \) and \( m = 80 \) in 2700 s, claiming that this configuration makes the puzzle hard to be solved. However, this is not true, especially when the dynamic programming algorithm is used instead. Therefore, the L3 algorithm is not recommended to be utilized in our CSA protocol suite.

It seems reasonable to conclude that, by adjusting the values of \( n \) and \( m \), determining the difficulty level of the subset sum (knapsack) problem and hence the cost-based approach can be adaptively realized.

5. Discussion

In this work, we have proposed an authentication protocol suite to identify and authenticate cloud users at the SaaS layer and provide a strong shield against DoS attacks. By integrating the client puzzle problem and the utilization of the unique encrypted text (UET), we were able to avoid the security breaches that may lead to DoS attacks as mentioned in Section 1 and illustrated in Fig. 3.

In the CSA protocol suite, we rely on the computational complexity to determine different levels of client-puzzle solution difficulties. We were thus able to design the identity protocol such that the computational cost incurred by the cloud resources is minimized and the computation cost incurred by cloud users is adjustable based on the service’s sensitivity. Note that high computational cost will influence an attacker launching a DoS attack with a massive number of requests from his device. However, if the attacker uses many different devices to launch DoS attacks, the cloud system will not be exhausted because the attack will be detected at an early stage of the authentication process.

Practically, the proposed CSA protocol suite can be implemented in the SaaS layer of cloud computing systems. Because the protocol simply relies on basic hardware and software requirements of both the cloud systems and cloud users. Our experiment showed that the traditional software and hardware tools were sufficient to fully implement the CSA protocol and that the dynamic programming algorithm was able to solve the required difficulty levels of the puzzle.

However, this work does not consider the possible DoS risks faced by the other cloud layers, such as PaaS or IaaS. Therefore, the CSA protocol suite can be implemented on private cloud architectures at this stage, but it needs to be developed for implementation on public or hybrid cloud architectures.

6. Conclusions

The use of software systems in a cloud-computing environment is increasingly common. Verifying users via an authentication protocol is considered to be an initial stage to access these systems. Consequently, the authentication protocol is a main target of attackers implementing a DoS attack that decreases the availability of cloud services. Using existing strong authentication protocols of traditional network systems in cloud-based applications may lead to DoS attack vulnerability because the initiation of a massive amount of authentication processes could exhaust the cloud’s resources and render the cloud-based application unreachable. In this study, the proposed CSA protocol suite aims to prevent internal and external risks to DoS attacks. The CSA protocol uses an adaptive challenge technique based on the required efforts of the participants. Using this technique allows the system to identify legitimate requests and pass them to the cloud applications. This CSA protocol suite does not require any external physical device for the authentication process. The effectiveness of the CSA protocol was experimentally analyzed in this work using a cost-based model approach. Finally, the ability of the protocol to fortify the system against a DoS attack was demonstrated.

Acknowledgment

This work was partially supported by King Abdulaziz University through the Cultural Bureau of Saudi Arabia in Canada. This support is greatly appreciated.

References


Juels A, Brainard J. Client puzzles: a cryptographic countermeasure against connection depletion attacks. In:
Salomaa A. Public-key cryptography. EATCS monographs on theoretical computer science. 2nd ed. Berlin: Springer; 1996.

Marwan Darwish is currently pursuing his Ph.D. degree in Software Engineering in the Department of Electrical and Computer Engineering at Western University. He received his B.Sc. and M.Sc. degrees in Electrical and Computer Engineering from King Abdulaziz University, Jeddah, Saudi Arabia, in 2003, and 2008 respectively. His research interest includes network security, information security and cloud computing security. Mr. Darwish worked for more than 3 years in the Technical and Vocational Training Corporation in Saudi Arabia, and he worked for 1 year at King Abdulaziz University for security and cloud computing security. Mr. Darwish also has a strong experience in project management, system analysis/consulting, and database design.

Dr. Abdelkader Ouda is an Assistant Professor in the Department of Electrical and Computer Engineering at Western University. He received his B.Sc. (Hons) and M.Sc. degrees in Computer Science and Pure Mathematics from Ain Shams University, Cairo, Egypt, in 1986, and 1993 respectively, and his Ph.D. degree in Computer Science from the Western University, Canada in 2005. Dr. Ouda has been studying, researching, and teaching software engineering at Western University for more than 12 years. His current research is mainly in Information security, including cryptography, network security, big data security and privacy, and data warehousing and big data analytics. Dr. Ouda was working for over 14 years in the software developing industry, where he has gained rich experiences in project management, system analysis/consulting, and database design.

Dr. Luiz Fernando Capretz has vast experience in the software engineering field as practitioner, manager and educator. Before joining Western University (Canada), he worked at both technical and managerial levels, taught and did research on the engineering of software in Brasil, Argentina, England, Japan, and the UAE. He is currently a professor of Software Engineering and Assistant Dean (IT and e-Learning). His research interests span the fields of software engineering, human aspects of software engineering, software analytics, and software engineering education.