Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 707548, 12 pages
http://dx.doi.org/10.1155/2015/707548

Research Article

A Novel WLAN Client Puzzle against DoS Attack Based on Pattern Matching

Ali Ordi, Mazdak Zamani, Norbik Bashah Idris, Azizah Abdul Manaf, and Mohd Shahidan Abdullah

Advanced Informatics School, Universiti Teknologi Malaysia, 54100 Kuala Lumpur, Malaysia

Correspondence should be addressed to Ali Ordi; ali.ordi@gmail.com

Received 17 September 2014; Accepted 7 November 2014

Academic Editor: Yudong Zhang

Despite the popularity of 802.11 based networks, they suffer several types of DoS attack, launched by an attacker whose aim is to make an access point (AP) unavailable to legitimate users. One of the most common DoS attacks on 802.11 based networks is to deplete the resources of the AP. A serious situation like this can occur when the AP receives a burst of connection requests. This paper addresses this common DoS attack and proposes a lightweight puzzle, based on pattern-matching. Using a pattern-matching technique, this model adequately resists resource-depletion attacks in terms of both puzzle generation and solution verification. Using a sensible series of contextual comparisons, the outcomes were modelled by a simulator, and the security definition and proofs are verified, among other results.

1. Introduction

Despite an unprecedented growth in popularity, using an open shared transmission medium makes wireless LANs (WLAN) extremely vulnerable to many attacks [1]. A series of security extensions to 802.11 have already been ratified to fix some of these vulnerabilities. However, these extensions primarily deal with vulnerabilities related to unauthorized access and confidentiality breaches. As our dependence on wireless access increases, it also becomes essential to consider the issue of availability as another important security requirement [2]. Denial-of-service (DoS) attacks strike against availability, attempting to prevent legitimate users from accessing the network. There are many 802.11-specific DoS vulnerabilities, which have been experimentally demonstrated in the literature of recent years.

Denial-of-service (DoS) attacks are a growing concern to networked services like the Internet. A DoS attack intends to deny access to shared services or resources by legitimate users [3]. A common form of WLAN DoS attack is a resource depletion attack, in which an attacker tries to overload the Access Point's (AP) resources, such as its memory-hosted association table, rendering the AP unable to service honest clients. A potential way to deal with this problem is for a defending server to identify and segregate malicious traffic as quickly as possible. Other forms of DoS attack are jamming attacks, semantic attacks, and implementation specific attacks.

To deal with DoS attack, a number of methods have been proposed by researchers. Particularly for resource depletion or connection request flooding attack, in [4] a number of countermeasures both in the physical and MAC layers have been discussed. These solutions are cryptographic protection, security protocol repair, intrusion detection systems (IDS), decreasing the retry limit, identifying with signal strength info, and identifying through RF fingerprint.

Client puzzles, also known as proofs of work, have been shown to be a promising tool to thwart DoS attacks on network protocols, particularly on authentication protocols. A puzzle is issued by the server in reply to each request when the server is under attack. After receiving a puzzle, the client has to solve it in order to convince the server to grant access to its resources. The main idea is that puzzle generation and solution verification should be easy for the server, while computing the puzzle solution should be somewhat hard, computationally speaking, for the client.

Many client puzzles have been proposed since they were first introduced by Dwork and Naor in 1992 [5]. An important recent development has been the analysis of client puzzles within the provable security framework [6, 7]. The main contribution of our paper is proposing a faster puzzle in verification and generation phases. Compared to other hash based puzzles, our work consumes at least 30% less resources in both generation and verification phases, all without the need for additional hardware. It also addresses the security definition of Chen et al. [6] and proves that our proposed puzzle is secure.

To emphasize the importance of CRF DoS attacks and to show how they are launched, we shall highlight parts of the 802.11 standard which present malicious users with opportunities to breach secure, fair, and efficient protocol operations.

The following section will describe how connection-request flooding attacks and spoofed disconnect attacks on 802.11 based networks occur. Section 3 will review the methods and approaches proposed by other researchers in the literature. However, more information related to other attacks and issues of network security can be found in [8–32]. More information related to patterns can be found in [33–36]. In Section 4, the details of the proposed puzzle will be discussed. The experimental results will be demonstrated in Section 5. Using security definition of Chen et al. [6] alongside the client puzzle protocol security properties, a security analysis of the proposed approach will be provided prior to the conclusion.

2. CRF DoS Attack on WLANs

Fundamentally, 802.11-based networks operate in two modes: Ad-hoc and Infrastructure. In addition, WLANs are deployed in three architectural models: Independent Basic Service Set (IBSS), Infrastructure BSS, and Extended BSS [37]. This paper will pay special attention to CRF DoS attacks on WLAN in Infrastructure mode, where a Basic Service Set (BSS) is orchestrated by the AP.

To get access eligibility, STAs must initially authenticate themselves to the AP. This is the same sort of operation as connecting a PC to a certain wired network outlet. In order to simplify the attachment of wireless stations (STA) to a network, the connection procedure in wireless networks has been designed without providing an authentication mechanism on MAC frame header fields, [38], particularly in open authentication mode. This security hole makes forging the source address of an MAC frame so easy that identifying the source of traffic is virtually impossible. Sending false connection requests is much cheaper than validating those requests. If the authentication server does not protect the limited-resource AP against false requests (whose aim is to exhaust available resources), the solution becomes challenging.

In addition, the authentication and association procedure has been designed as a stateful process. This means that AP is required to allocate a certain amount of resources—normally memory capacity—for every connection request in order to track the current state, as shown in Figure 1.

Despite the significant benefits of the authentication and association procedure, there is a clear sign that this procedure could easily become a simple route to deny service [39]. An attacker can effortlessly launch a CRF DoS attack by forwarding a burst of bogus connection requests—whether probe, authentication, or association request frame—over a relatively short time, towards an unprotected AP. Consequently, the victimized AP runs out of resources (i.e. association table) quickly, so that legitimate requests remain unanswered [3, 40]. In addition, after making a wireless network disappear, a fake system belonging to an attacker may pose as the legitimate wireless infrastructure, which enables the attacker to launch a man-in-the-middle attack [41].

Even though a number of anti-CRF DoS schemes have already been proposed for wired networks, they are unsuitable to protect resource-limited wireless infrastructure [42].

3. Related Works

Over the past decades, a whole bunch of countermeasures have been proposed by researchers to mitigate or even eliminate the harmful effects of CRF DoS attacks on computer networks. In this section, we review the relevant literature on client puzzles, with an emphasis on hash-based puzzles.

Client Puzzle. The idea of client puzzle protocol is quite simple [43]. When server is not under attack, it follows its normal activity and accepts connection requests. When a server comes under attack, the server forwards a unique puzzle to each client wishing to attach to the network. Solving these puzzles imposes a certain amount of computation and storage cost on clients. If a client submits the valid solution to the server, the server will allocate the required resources to that client. As it turns out, a legitimate client has to undergo an insignificant computational cost when a server comes under attack, while an attacker needs large computational resources to make a noticeable interruption in the network service. The use of puzzles in cryptography was pioneered by Merkle [44], who used puzzles to establish a secret key between parties over an insecure channel. Client puzzles were first proposed by Dwork and Naor [5] as a countermeasure for email spam. The computational problems underlying most puzzles are either number-theoretic [5, 45, 46] or based on hash inversions [6, 47, 48]. Hash-based puzzles are very efficient—generation and verification typically requires only one or two hash function calls—but concrete realizations to date have been shown to be secure only in the random oracle model. Number-theoretic puzzles, on the other hand, have been shown to be secure in the standard model but have tended to be relatively inefficient, typically requiring the server to perform a large-integer modular exponentiation, making it unsuitable for resource-limited WLANs. Hence, some researchers focused on other payment schemes than CPU cycles, for example, memory-bound puzzles [49–51],

Table 1: DoS-resistant authentication based on extracting square roots.

Client		AP
	$\xleftarrow{\text{Beacon Frame}}$ (L, R, D, z)	AP periodically decides D (difficulty), generates $n = p \times q$ (2 large prime numbers), sets a time limit Δt, and chooses L, R, and z $\text{Rand}(z) \in \{1, 2, \ldots, w\}$, w is the size of n
Client solves X by brute force from equation $X^2 \equiv a \pmod{n}$ within Δt		
	$\xrightarrow{\text{Authentication Req.}}$ (X, L, R, D, z)	AP verifies that R and L are recent AP computes $a' \equiv X^2 \pmod{n}$ AP verifies that a' satisfies the condition AP may now commit resources

knapsack puzzles [52], NST puzzles [53], bandwidth puzzles [54, 55], or human interaction with so-called CAPTCHAs [56].

Hash-Based Puzzle. Juels and Brainard (see [57]) employed a cryptographic client puzzle protocol to protect servers against CRF DoS attack.

Aura et al. [48] employed the client puzzle in an authentication protocol. They proposed a CPU-bound puzzle based on partial collision in a one-way hash function, protected by digital signature. The client solves X from the following equation by brute force and sends the signed solution X back to the server:

$$h(C, N_s, N_c, X) = \overbrace{000 \cdots 000}^{\text{The } k \text{ first bits of the hash}} \quad Y. \tag{1}$$

The authors claimed that their method is a lightweight puzzle and consumes as little CPU-cycle as possible. However, using a public-key signature may increase significantly the likelihood of resource-exhaustion on the authenticator's side (AP or server), particularly when the network environment is WLAN [58]. Moreover, when using hash-based puzzles, some issues have to be taken into consideration, including the following:

(i) The puzzle difficulty is not fine-grain controlled [59]. In other words, the difficulty grows exponentially while the number of bits needed for partial collision increases.

(ii) Like other CPU-bound puzzles, this puzzle suffers from the CPU-power disparity issue [59]. CPU-power disparity issues occur where a puzzle depends on computational power. In such circumstances, clients equipped with high resources are able to solve puzzles in a shorter time than others. Attackers take full advantage of this security hole.

(iii) This method is recognized as a parallelizable puzzle, so the attacker can solve the puzzle in a parallel manner using a number of employed clients.

To mitigate the overloading of Aura's puzzle on APs, Shi and Ma [60] designed a lightweight anti-DoS attack wireless authentication scheme on the basis of a hash function. They put a signature computation only after the puzzle verification. Still, the AP has to generate a puzzle for every client's request and to store it. To handle a burst of connection requests containing spoofed or unauthenticated MAC addresses, the AP has to exhaust more resources to generate more puzzles. This may lead to memory exhaustion. Moreover, this puzzle still suffers from the hash reversal puzzle issues mentioned above.

To eliminate the Aura puzzle defect, Dong et al. [58] proposed a hash based puzzle to handle two key problems: avoiding the possible memory-exhaustion in the earlier puzzle and combating the CRF attack. They took advantage of beacon frames to distribute the puzzle parameters. This protects the AP against a second DoS attack on puzzle generation where AP faces a sudden burst of puzzle requests. However, this scheme does not provide a solution to the problems raised by hash-based puzzles. The authors in [61] proposed a CPU-bound client puzzle based on extracting square roots. Table 1 exposes the proposed client puzzle in detail.

By using beacon frames as the puzzle carrier, this puzzle keeps the AP away from a puzzle request flooding attack. Moreover, it is claimed that by choosing two large prime numbers p and q along with Δt, it is ensured that forging multiple solutions in a short time is impossible. Despite these valuable achievements, a number of drawbacks should be taken into consideration:

(i) Using quadratic residue problems imposes a large computational load on an STA, which in the case of a PDA, tablet, or other limited-resource device can be a problem.

(ii) Allocating 77 bytes of management frame to a puzzle wastes wireless bandwidth.

(iii) Even though the verification is low cost, sending too many fake solutions can overburden the AP.

(iv) The results show the puzzle difficulty of this puzzle is poorly granular.

(v) Like other CPU-bound puzzles, this puzzle suffers from CPU-power disparity issue.

As the granularity of hash-based reversal puzzles is too coarse, Feng et al. [62] proposed the idea of a hint-based hash reversal puzzle to allow the granularity to be linear. The behind idea of the proposed puzzle is that the server gives

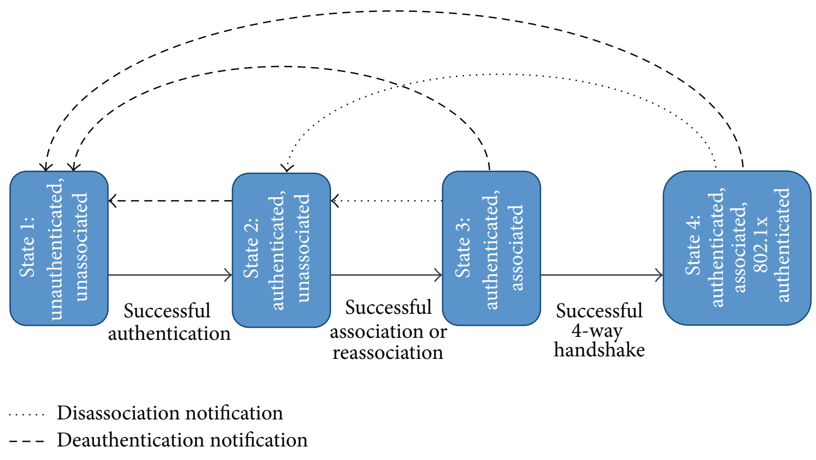
State 1: unauthenticated, unassociated
State 2: authenticated, unassociated
State 3: authenticated, associated
State 4: authenticated, associated, 802.1x authenticated
Successful authentication
Successful association or reassociation
Successful 4-way handshake
Disassociation notification
Deauthentication notification

Figure 1: Authentication and association states.

the client a hint about the range within which the solution lies. For example, suppose a randomly generated number x is used as the input to the hash $h(x)$. To generate a puzzle with $O(D)$ difficulty, the server passes the client the hash and a hint, $x - u(0, D)$. Where $u(0, D)$ is a randomly chosen number uniformly distributed between 0 and D. Instead of checking every possible solution, the client starts at the hint and searches the range linearly for the answer. Apart from the last process, all remaining processes are similar to Jules and Brainard's puzzle. Hence, the simple puzzle generation and verification, as well as the linear granularity for fine grained control are the strengths of this solution. However, it is still susceptible to parallel processing attacks. Also it is suspicious of DoS puzzle attacks on the AP during puzzle-generation, where an attacker sends a burst of puzzle requests and the AP has to produce a unique puzzle for every request. Moreover, like other CPU-bound puzzle, a hint-based hash-reversal puzzle suffers from CPU speed disparity issue.

Similarly, Lei et al. [63] introduced the quasi-partial collision concept to provide a granularity of puzzle difficulty in a more controlled way. The results show a marked improvement on earlier puzzles; however, the hash-based CPU-bound problems still remain.

Number-Theoretic Puzzle. Rivest et al. [64] proposed a non-parallelizable time-lock puzzle based on repeated-squaring to enable time-release cryptography. The idea behind it is that a client has to allocate a certain amount of computational resource time to execute repeated squaring. First, a server calculates roughly how many squaring operations a client is able to run per second: S. Then it specifies the time required by a client to find the puzzle's solution: T. Considering this information, server computes how many times a client must run the squaring function to find the solution: $t = T \times S$. Finally, the server encrypts a message M into a cipher text C as follows: $C = M + X^{a^t} \bmod N$, given an integer X, an exponent a, a large integer t, and an appropriate semiprime modulus N. To acquire M from C, the client needs to compute $X^{a^t} \bmod N$ given X, a, t, and N in $\log(a^t) \approx t$ modular multiplications. This computation can be performed efficiently using the trapdoor offered by Euler's function: $X^{a^t} \bmod N \equiv X^{a^t \bmod \varphi(N)} \bmod N$. Surprisingly, time-lock puzzles are able to provide a precise and fixed amount of work. To verify the puzzle, the server checks the solution through the trapdoor offered by Euler's function in $O(\log(N))$ modular multiplications.

Although this nonparallelizable puzzle eliminates the CPU-power disparity problem, there are some fundamental drawbacks which make this puzzle impractical in the real world.

(i) Generating and storing big prime numbers may be resource-exhausting on the server side. As a result, time-lock puzzles meet the inefficiency threshold in puzzle generation in the order of microseconds [65].

(ii) To verify the puzzle, the server has to allocate a significant amount of resources to compute $e = a^t (\bmod \varphi(N))$ and $b = X^e (\bmod N)$ for every received puzzle solution, which is undesirable in WLANs.

(iii) The server has to find out some information, such as CPU-power, from the clients in order to calculate the precise puzzle parameters. Attaining this information from a heterogeneous environment of clients, such as in WLANs, is almost impossible.

To reduce the costs of puzzle verification of Rivest's puzzle, Karame and Čapkun [45] adapted Rivest's puzzle by employing an RSA key pair with small private exponent. This puzzle is based on the assumption that it is computationally intractable to compute a small private exponent d when the public exponent e is larger by several orders of magnitude than the modulus N. The server must still perform a modular exponentiation but, given a semiprime modulus N, the number of multiplications is decreased by a factor of $|N|/k$ where k is a security parameter. For example, a factor 12.8 for a 1024 bit modulus results in 120 modular multiplications instead of 1536.

In spite of this significant reduction in the verification and generation cost, still the verification costs remain too high to provide viable DoS protection for WLANs where resources are much limited [66]. In particular, the puzzle setup is extremely costly. Also, this method suffers from difficult granularity issues. Furthermore, this puzzle wastes the network's bandwidth due to employing large numbers.

Recently, Rangasamy et al. [67] proposed a modular exponentiation-based client puzzle which can be seen as an efficient alternative to Rivest et al.'s time-lock puzzle. Unlike the Rivest et al. and Karame-Čapkun puzzle, Rangasamy et al.'s puzzle does not require the server to perform any online exponentiations. In fact, the server has to perform a total of two hash operations and a few modular multiplications for the puzzle generation and verification. Although it is a significant improvement over the Karame-Ĉapkun puzzle construction, the security of the puzzle does not rely on the standard security assumptions.

Kuppusamy et al. [68] also proposed a DLPuz puzzle based on the interval-discrete logarithm problem. This puzzle is claimed to be efficient while remaining secure in the standard model. Among other number theoretic puzzles, DLPuz puzzles demonstrate an extremely efficient verification algorithm, although the results show a costly puzzle generation which is prone to DoS attack.

4. PatPuz: A Superfast Hash-Based WLAN Puzzle

This section describes our new client puzzle construction PatPuz, which is based on finding a pattern key through reversing a one-way hash function. First, we will review the definition of a client puzzle and then present our construction.

The PatPuz will be able to manage connection requests so that they lose their importance as a valuable target for launching a DoS attack. Furthermore, this puzzle resists bogus puzzle solution flood attack where attackers strive to overload an AP during the puzzle verification stage.

Notation. If n is an integer, then we use $|n|$ to denote the length in bits of n. The puzzle makes use of a family of keyed hash function $\mathscr{H} := \{H_k\}_{k\in K}$, where each H_k is a function mapping G to $\{0,1\}^{\ell}$ and k is security parameter. We let $x \leftarrow_R S$ denote an uniformly random choosing of x from S where S is a set. We write $x \leftarrow \mathscr{A}(y)$ to assign the output of algorithm $\mathscr{A}$ to x when run with the input y. If k is a security parameter, then $negl(k)$ denotes a function that is negligible in k, namely asymptotically smaller than the inverse of any polynomial in k. Let $P_{x,z}(R)$ be a pattern function and $V(p_n)$ denotes the p_n-th bit of R, then the following patterns are applied on R:

(i) Extract n values of length ℓ from $h(x)$ as shown in Figure 4.

(ii) R changes as follows:

If $z = 1$ then $V(p_{2n}) = V(p_{2n+1})$ otherwise $V(p_{2n}) \neq V(p_{2n+1})$.

If $x = x_0, x_1, \ldots, x_i, \ldots, x_j, \ldots, x_n$ is a bit string then we let $x\langle i, j\rangle$ denote the substring $x_i, \ldots, x_j$. We use $y = \sigma_{s,\alpha}(x)$ to denote $y = \sigma_s(x)\langle 1, \alpha\rangle$.

4.1. Client Puzzle Formal Definition.

In [6], Chen et al. defines a client puzzle formally as follows.

Definition 1 (client puzzle). A client puzzle Puz is a tuple consisting of the following algorithms:

(i) Setup(1^k): a p.p.t. setup algorithm that generates and returns a set of public parameters *params* and a secret key s, the former of which includes a puzzle difficulty parameter space *QSpace*.

(ii) GenPuz(s, Q, str): a p.p.t. puzzle generation algorithm which accepts a secret key s, difficulty parameter Q, and a session string str and returns a puzzle puz.

(iii) FindSoln(puz, τ): a probabilistic puzzle solving algorithm that returns a potential solution $soln$ for puzzle puz after running time at most τ.

(iv) VerAuth(s, puz): a d.p.t. puzzle authenticity verification algorithm that returns true or false.

(v) VerSoln$(s, str, puz, soln)$: a d.p.t. puzzle solution verification algorithm that returns true or false.

For correctness, we require that if $(params; s) \leftarrow$ Setup(1^k) and $puz \leftarrow$ GenPuz(s, Q, str) then there exists $\tau \in \mathbb{N}$ such that VerSoln$(s, str, puz, soln)$ is true with probability 1 where $soln \leftarrow$ FindSoln(puz, τ).

4.2. The PatPuz Puzzle.

This scheme aims to provide a unique puzzle for every connection request received by an AP. The task of an STA is to find a pattern key through inversing a one-way hash function and to use that pattern key to create a unique pattern as the puzzle solution. Then, the AP only verifies those patterns. In many scenarios, it is essential that the GenPuz, VerAuth, and VerSoln algorithms be extremely efficient. In a denial-of-service setting, these algorithms are run online by the AP many times, and if they were expensive, then an attacker could induce a second resource depletion attack by asking for many puzzles to be generated or verified.

Setup(1^k). The various spaces are chosen; $sSpace \leftarrow \mathscr{K}$, $QSpace \leftarrow \mathbb{N}$, $strSpace \leftarrow \{0,1\}^*$, $solnSpace \leftarrow \mathscr{X}$, $paternSpace \leftarrow \mathscr{P}$, and $puzSpace \leftarrow QSpace \times strSpace \times \{0,1\}^k \times \mathscr{Y}$. The value s is chosen as $s \overset{\$}{\leftarrow} sSpace$ then output.

GenPuz(s, Q, str). A nonce as a timestamp for current cycle is selected $n_s \overset{\$}{\leftarrow} \{0,1\}^k$. Let $str \leftarrow MAC_{\text{AP}}$. Next x is computed as $x \leftarrow F_{s,Q}(Q, n_s, str)$. The value $y \in \mathscr{Y}$ is computed as $y \leftarrow \varphi(x, n_s, Q)$ and the puzzle assigned to be $puz = (Q, str, n_s, y)$ and output.

FindSoln(puz, τ). One typical method for a legitimate client to implement the FindSoln algorithm is a brute-force search. While this algorithm is within the allowed number of clock cycles of execution, it randomly samples elements from the set of possible solutions without replacement and for each potential $x' \in \mathscr{X}$ computes $y' \leftarrow \varphi(x', n_s, Q)$. If $y' = y$ then it computes $R' \leftarrow \mathscr{P}_{x',z}(R)$ output R' then halts and

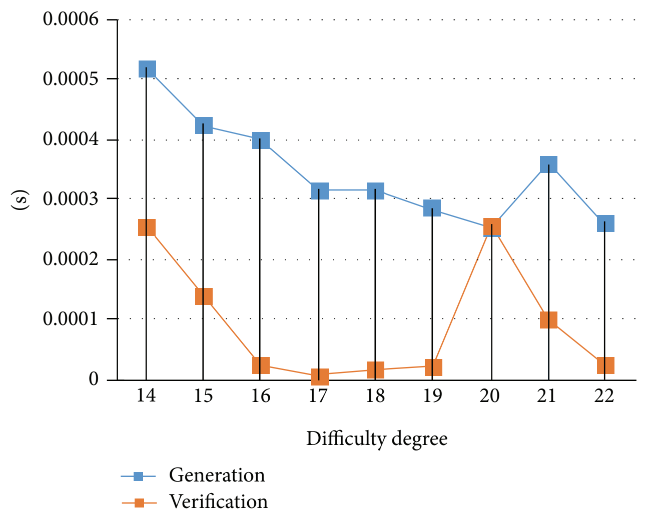

FIGURE 2: The curve relationship between difficulties and time-consuming on generating and verifying the puzzle.

otherwise continues with random sampling. If this algorithm reaches the last clock cycle of execution then it outputs an R' computed from a random element of the remaining unsampled preimage space.

VerAuth(s, x, puz'). For a puzzle $puz = (Q', str', n'_s, x', y')$ this computes $x \leftarrow F_{s,Q}(Q', n'_s, str')$; then if $x = x'$ this output is true and otherwise the output is false.

VerSoln(puz', R'). Given a potential solution R' this checks if $R' \in \mathcal{P}_x$ and if so outputs are true and otherwise output is false.

Remark 2. In PatPuz, the AP has to store the short-term secrets x and P_x in order to speed up the verification phases. These secrets are expired when a cycle times out and the new values are replaced.

When the pattern sent by the STA passes the verification step, the AP will send an authentication response frame back to the STA and allocates the required resources.

5. Performance Experiment of PatPuz

NS2 simulates our scheme over a core i5 2.27 GHz/4G Ubuntu system. Since our goal is to balance the resource consumption between AP and STA, we focus on the time consumed on the puzzle generation and verification on the one hand and puzzle solving on the other. We will consider the puzzle difficulty Q to be less than 22 for simplicity reasons. The difficulty of a client puzzle gives a measure of the likelihood of an adversary finding a solution to a given puzzle within a given number of clock cycles of execution. Intuitively, the difficulty of solving a puzzle is ensured by the hardness of inverting the one-way function.

Figure 2 demonstrates the link between difficulties and time-consumption in both puzzle generation and solution verification. The obtained results are important for two

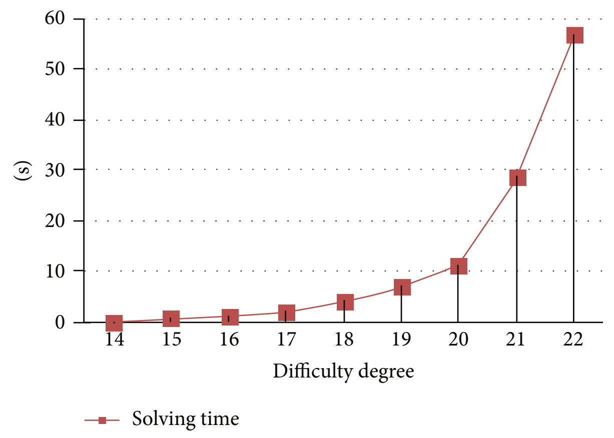

FIGURE 3: The curve relationship between difficulty degree and time-consuming on solving the puzzle.

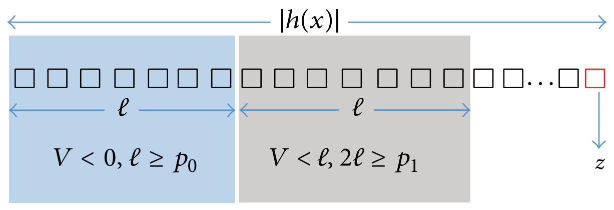

FIGURE 4: Extracting the values from $h(x)$ to calculate the correct positions for applying the pattern accordingly.

reasons. First of all, the time the AP spends to generate and verify a new puzzle is considerably shorter. It is less than 0.6 and 0.3 milliseconds in generation and verification procedure, respectively. Secondly, the fluctuation in generation and verification time is almost constant when the difficulty degree goes up or down. Therefore, increasing or decreasing the difficulty degree does not affect the time needed by the CPU for generating and verifying the puzzle.

Consequently, the proposed scheme eliminates any chance for an attacker to make puzzle generation and verification phases into a valuable target to launch DoS attacks by using fake solutions.

The most important impact of our scheme is to force STAs to cost their resources for every connection request. Therefore, the chance of attacker exhausting AP's resources by sending a burst of fake connection requests is very small. As shown in Figure 3, the solving time increases considerably, by nearly exponential growth, when difficulty of degree goes up.

6. Security Analysis of PatPuz

In this section, we analyse the PatPuz puzzle using the security model of Chen et al. [6]. Chen et al. introduced two security properties that a client puzzle should satisfy: unforgeability and difficulty. We shall give a brief description of these two properties. Intuitively, the unforgeability of PatPuz is ensured by the use of a pseudo-random function and the difficulty of solving PatPuz puzzles is ensured by difficulty of inverting the one-way function.

6.1. Unforgeability. This experiment measures the ability of an adversary to produce a valid client puzzle and force a server to accept it as one that was not originally generated by a server in a probabilistic way. In general, unforgeability can easily be provided by using a message authentication code (MAC) or pseudo-random function to tag puzzles generated by the server, and this is what is done in PatPuz. First we review the formal definition of puzzle unforgeability in the next section. The results show that PatPuz is indeed unforgeable. We shall make use of a sequence of games [69] to prove the security properties.

Definition 3 (unforgeability [6]). Let k be a security parameter, $\mathscr{A}$ a probabilistic algorithm, and Puz a client puzzle. Define the experiment $Exec^{\text{UF}}_{\mathscr{A},Puz}(k)$ as follows:

(1) $(params, s) \leftarrow Setup\ (1^k)$.

(2) Run $\mathscr{A}(params)$ with oracle access to CreatePuz$(\cdot)$ and CheckPuz$(\cdot)$, which are answered as follows:

  (a) CreatePuz(str, Q): $puz \leftarrow GenPuz(s, Q, str)$. Return Puz to $\mathscr{A}$.

  (b) CheckPuz(puz): if puz was not an output for any of the CreatePuz(str) query made previously and VerAuth(s, puz) = true then stop the experiment and output 1. Otherwise, return false to $\mathscr{A}$.

(3) Output 0.

We say that $\mathscr{A}$ wins the game if $Exec^{\text{UF}}_{\mathscr{A},Puz}(k) = 1$ and loses otherwise. The advantage of $\mathscr{A}$ is defined as:

$$Adv^{\text{UF}}_{\mathscr{A},Puz}(k) = \Pr\left(Exec^{\text{UF}}_{\mathscr{A},Puz}(k) = 1\right). \quad (2)$$

A puzzle is said to be unforgeable if this advantage is negligible in k for probabilistic algorithms $\mathscr{A}$ running in time polynomial in k.

In this unforgeability experiment, the adversary is allowed to query the CreatePuz oracle by choosing puzzle difficulty level Q at will. This is to ensure that even after seeing puzzles with different difficulty levels, the adversary cannot create a valid looking puzzle.

Theorem 4 (unforgeability of PatPuz puzzle). *The PatPuz puzzle is unforgeable.*

Proof. We prove the theorem using a sequence of games. Let $\mathscr{A}$ be a probabilistic algorithm with running time t. Let S_i be the event that $\mathscr{A}$ wins in game G_i.

Game G_0. Let G_0 be the original unforgeability game $Exp^{\text{UF}}_{\mathscr{A},\text{PatPuz}}(k)$. Then

$$\Pr\left(Exec^{\text{UF}}_{\mathscr{A},OROD}(k') = 1\right) = \Pr(S_0). \quad (3)$$

Game G_1. In this game, we modify game G_0 by replacing the F_s with a truly random function $\mathscr{R}$ to compute the pattern key x. This change has a negligible effect on adversary $\mathscr{A}$ because of the pseudo-randomness of F_s. Hence,

$$\left|\Pr(S_0) - \Pr(S_1)\right| \leq Adv^{F_s}_{\mathscr{B}}(k) \leq negl(k), \quad (4)$$

where $\mathscr{B}$ is an algorithm running in time $O(t)$, and the second inequality follows whenever F_s is a pseudo-random function.

Since the function $\mathscr{R}$ in game 1 is truly random, the probability that an adversary without access to $\mathscr{R}$ can guess an output is negligible:

$$\Pr(S_1) \leq \frac{1}{2^k}. \quad (5)$$

Combining (3) through (5), we obtain the final result that the adversary's success in forging a puzzle is negligible. □

6.2. Difficulty. To prove our puzzle difficulty, we shall mainly focus on generating a valid pattern without having to find. Like the previous property, we shall first define the formal form of the puzzle difficulty presented by Chen et al., then we will show that our puzzle is a difficult puzzle.

Definition 5 (puzzle difficulty). Let k be a security parameter and let Q be a difficulty parameter which is kept fixed through the experiment. Let $\mathscr{A}$ be a probabilistic algorithm and puz be a client puzzle. The game $Exec^{Q,DIFF}_{\mathscr{A},Puz}(k)$ is defined for each hardness parameter $Q \in \mathbb{N}$ as follows:

(1) $(params, s) \leftarrow Setup\ (1^k)$

(2) Run $\mathscr{A}(params)$ with oracle access to CreatePuzSoln$(\cdot)$ and Test$(\cdot)$, which are answered as follows:

  (a) CreatePuzSoln(str): $puz \leftarrow GenPuz(s, Q, str)$. Find a solution $soln$ such as VerSoln$(puz, soln)$ = true. Return $(puz, soln)$ to $\mathscr{A}$.

  (b) Test(str^*): This query may ask once, at any point during the game. The challenger generates a puzzle $puz^* \leftarrow GenPuz(s, Q, str)$ and returns puz^* to $\mathscr{A}$. Then $\mathscr{A}$ may continue to ask CreatePuzSoln queries.

(3) $\mathscr{A}$ output a potential solution $soln^*$.

(4) Output 1 if VerSoln$(puz^*, soln^*)$ = true and 0 otherwise.

We say that $\mathscr{A}$ wins the game if $Exec^{Q,DIFF}_{\mathscr{A},Puz}(k) = 1$ and loses otherwise. We define the success of an adversary $\mathscr{A}$ against puz as

$$Succ^{Q,DIFF}_{A,Puz}(k) = \Pr\left[Exec^{Q,DIFF}_{\mathscr{A},Puz}(k) = 1\right]. \quad (6)$$

Let $\epsilon_{k,Q}(t)$ be a family of functions monotonically increasing in t. A puzzle is $\epsilon_{k,Q}(\cdot)$-*difficult* if, for all probabilistic algorithm $\mathscr{A}$ running in time at most t,

$$Succ^{Q,DIFF}_{A,Puz}(k) \leq \epsilon_{k,Q}(t). \quad (7)$$

Theorem 6 (difficulty of PatPuz puzzle). *Let k be a security parameter and let Q be a difficulty parameter. Let $\mathscr{H} := \{H_k\}_{k\in K}$ be a family function of keyed hash function, where each H_k is a function mapping $\{0,1\}^{\ell_1}$ to $\{0,1\}^{\ell_2}$, and let $\mathscr{F}$ be a pseudo-random function. Then PatPuz puzzle is $\epsilon_{k,Q}(t)$-difficult for all probabilistic algorithms $\mathscr{A}$ running in time at most τ.*

Proof. Like previous properties, we shall employ a sequence of games to prove our theorem. In game G_0 the adversary tries to break our construction whereas in game G_1 the adversary works against an idealized version of our construction, where random function $\mathscr{F}$ has been replaced with a truly random function. Let $\mathscr{A}$ be a probabilistic algorithm with running time τ. Let S_i be the event that $\mathscr{A}$ wins the game G_i.

Game G_0. This game represents the original difficulty game $Exec^{Q,DIFF}_{\mathscr{A},\mathrm{PatPuz}}(k)$. For clarity, we shall write the full definition of this game:

(1) The challenger $\mathscr{C}$ first runs the Setup on 1^k and obtains $\xleftarrow{\$}$ *sSpace*. s is kept secret.

(2) The adversary $\mathscr{A}$ will then starts to ask CreatePuzSoln() queries. To answer each $\mathscr{C}$ select a random nonce $s \xleftarrow{\$} \{0,1\}^k$ and a random number $\xleftarrow{\$} \{0,1\}^*$. Then computes x through $x \leftarrow F_{s,Q}(Q, n_s, str)$. Next $\mathscr{C}$ computes $y \leftarrow \varphi(x, n_s, Q)$ and $R' \leftarrow \mathscr{P}_{x',z}(R)$ then returns $puz \leftarrow (Q, str, n_s, y)$ and $soln \leftarrow R'$ to $\mathscr{A}$.

(3) At any time during the game, $\mathscr{A}$ is allowed to issue a Test() query for which $\mathscr{C}$ generates a puzzle $paz^* = (Q, str, n_s^*, y^*)$ and returns paz^* to $\mathscr{A}$.

(4) The adversary $\mathscr{A}$ may continue to ask CreatePuzSoln() queries which the challenger $\mathscr{C}$ answers as before.

(5) Eventually, after τ clock cycles $\mathscr{A}$ outputs a potential solution $soln^* = R^*$. If VerSoln(puz^*, R^*) = true, the adversary $\mathscr{A}$ wins and then the challenger $\mathscr{C}$ outputs 1 and terminates and otherwise outputs 0 and terminates.

Hence,

$$\Pr\left(Exec^{Q,DIFF}_{\mathscr{A},PatPuz}(k) = 1\right) = \Pr(S_0). \tag{8}$$

Game G_1. Now we transform game G_0 into game G_1, replacing the pseudo-random function F with a truly random function $\mathscr{R}$. This change is indistinguishable due to the pseudo-randomness of F, so

$$\left|\Pr(S_0) - \Pr(S_1)\right| \le negl(k). \tag{9}$$

Game G_2. We now transform game G_1 into game G_2, replacing the hash function $\varphi(\cdot)$ with a randomly chosen one, $\varphi^\dagger(\cdot)$: $h \xleftarrow{\$} \{0,1\}^\ell$. We assume that the family function of hash function $\mathscr{H}$ is entropy smoothing. This means that it is hard to distinguish $(k, H_k(\delta))$ from (k, h), where k is a random element of K, δ is a random element of G, and h is a random element of $\{0,1\}^\ell$. Algorithmically, game G_2 looks like this:

(1) The challenger $\mathscr{C}$ first runs the Setup on 1^k and obtains $s \xleftarrow{\$}$ *sSpace*. s is kept secret.

(2) The adversary $\mathscr{A}$ will then start to ask CreatePuzSoln() queries. $\mathscr{C}$ answers to queries as follows:

   (a) CreatePuzSoln(): as in game G_1 except replacing $\varphi^\dagger$ to compute y.

(3) At any time during the game, $\mathscr{A}$ is allowed to issue a Test() query for which $\mathscr{C}$ do the same as G_1 except replacing $\varphi^\dagger$ to compute y.

(4) The adversary $\mathscr{A}$ may continue to ask CreatePuzSoln() queries which the challenger $\mathscr{C}$ answers as before.

(5) Eventually, after τ clock cycles $\mathscr{A}$ outputs a potential solution $soln^* = R^*$. If VerSoln(puz^*, R^*) = true, the adversary $\mathscr{A}$ wins and then the challenger $\mathscr{C}$ outputs 1 and terminates and otherwise outputs 0 and terminates.

In the above game, a truly random function $\mathscr{R}$ is input to the function $\varphi^\dagger$ to compute y. As a result the only way $\mathscr{A}$ could find a correct solution R' would be for $\mathscr{A}$ to invert the function $\varphi^\dagger$. Hence we have

$$\Pr(S_2) \le Adv^{\mathrm{OWF}}_{A,H} \le \frac{1}{2^{n\times\ell\times|z|}}, \tag{10}$$

where S_2 is the event that $\mathscr{A}$ wins the game G_2, n is the number of pattern points, ℓ is the length of pattern points such that $2^\ell = |h(x)|$, and $Adv^{\mathrm{OWF}}_{A,H}$ is defined as below. The second inequality follows the fact that the only way a protected (by our scheme) AP may be attacked is to reveal the pattern. An attacker can reach the correct pattern either through solving the puzzle or guessing the pattern. If he or she chooses the first way, the main goal of our scheme is satisfied. The probability of finding a correct R' is $1/2^{n\times\ell\times|z|}$ in which in case of $n = 6$, $\ell = 7$, and $|z| = 2$ the result would be $1/2^{43}$ which is negligible.

Definition 7. For an adversary $\mathscr{A}$ we define its advantage against a function $\psi : \mathscr{X} \mapsto \mathscr{Y}$, where $\mathscr{X}$ is fixed and finite, in terms of OWF as

$$Adv^{\mathrm{OWF}}_{\mathscr{A},\psi} = \Pr x\left[x \xleftarrow{\$} \mathscr{X}; y \longleftarrow \psi(x); \left(\tilde{x} \longleftarrow \mathscr{A}(y) \wedge \psi(\tilde{x}) = y\right)\right]. \tag{11}$$

Let $\varepsilon_i : \mathbb{N} \mapsto [0,1]$ be a monotonically increasing function. Then the function ψ is an $\varepsilon_i(\cdot)$-OWF if for all adversaries $\mathscr{A}$ it holds that $Adv^{\mathrm{OWF}}_{\mathscr{A},\psi} \le \varepsilon_i(\tau)$.

We also claim that

$$\left|\Pr(S_1) - \Pr(S_2)\right| = \epsilon_h(\tau), \tag{12}$$

where ϵ_h is the h-advantage of some efficient algorithm (which is negligible assuming $\mathscr{H}$ is entropy smoothing).

To prove this claim, any difference between $\Pr(S_1)$ and $\Pr(S_2)$ can be parlayed into a corresponding h-advantage. The following algorithm D interpolates between game G_1 and game G_2 and so has h-advantage equal to $|\Pr(S_1) - \Pr(S_2)|$.

Algorithm D (k, h)

```
(1) $\mathscr{C}$ first runs the Setup on $1^k$: $s \overset{\$}{\leftarrow} sSpace$
(2) Run $\mathscr{A}$ with oracle access to CreatePuzSoln() and Test(), which are answered as follows:
    (a) CreatePuzSoln():
        (i) $n_s \overset{\$}{\leftarrow} \{0,1\}^k$.
        (ii) $R \overset{\$}{\leftarrow} \{0,1\}^*$
        (iii) $x \leftarrow \{0,1\}^*_Q$
        (iv) $compute\ y$
        (v) $R' \leftarrow P_{x,z}(R)$
        (vi) Return $(puz, soln) \leftarrow (y, R')$
    (b) Test(): $puz^* \leftarrow y^*$ and return $puz^*$
(3) $\mathscr{A}$ outputs a potential solution $soln^*$
(4) $\mathscr{A}$ may continue to ask CreatePuzSoln() queries which are answered as before.
(5) Output 1 if VerSoln($puz^*, soln^*$) = true and 0 otherwise.
```

Based on this indistinguishability assumption, $|\Pr(S_1) - \Pr(S_2)|$ is negligible.

Combining (8) through (12) yields the desired result. □

6.3. Security Properties. In [70, 71], some general criteria are listed to specify the properties a puzzle must meet to be considered as an effective and efficient anti-DoS approach. Simply put, these criteria prove how secure and powerful the proposed client puzzle protocol is. Here we review these properties with an emphasis on PatPuz puzzle.

Computation Guarantee. Since the hash function is considered to resist preimage and collision attack, the only way to solve a puzzle is to use a brute force method. Hence, STAs have to look up a range of 2^k possible solutions to find out the right pattern. Even though this range may be reduced to $2^{k/2}$ possible solutions due to a birthday attack [72], the client (and also attacker) still has to spend enough time to find the puzzle's solution.

Adjustability of Difficulty. In our scheme, the AP adjusts the difficulty degree by increasing or decreasing Q. Note that these variations have no effect on the time spent to generate or verify the puzzle.

Efficiency. Since the puzzle verification is done only by looking for a correct pattern in a puzzle solution R'—a significantly low computational process—the proposed client puzzle protocol resists a puzzle verification attack where an attacker forwards too many bogus puzzle solutions. In addition, a protected AP is required to store only a long-term secret value s, to verify the received solutions. To make the verification phase more efficient, a short-term value, which is a Q-bit length, can be stored by AP.

Correlation Free. The PatPuz puzzle is correlation-free. That means knowing all previous puzzle solutions does not help to solve the current puzzle in any way. This property is provided by the unpredictability feature of the random numbers generated in puzzle generation stage (s) and puzzle solving (R).

Stateless. The proposed client puzzle protocol does not require storing any client's or puzzle related information, except s, and x. Moreover, the memory allocated to x is cleared after changing the puzzle, meaning that the algorithm utilizes a fixed-size memory to handle the puzzle. Hence, a protected AP in our scheme will face no memory shortage in a relatively short time.

Tamper-Resistance. No STA (attacker) is able to learn x by examining the other STAs' solutions. Every STA, in our scheme, is required to apply the produced pattern over its own R which is basically random.

7. Conclusion and Future Work

The main consideration for implementing any security protocol in 802.11 based networks is how much cost the proposed algorithm imposes on both the CPU and memory to complete its tasks. This paper proposes a novel puzzle to meet the AP's constraints and protect wireless network against CRF DoS attacks.

The following items are satisfied by the proposed scheme:

(i) *Low-cost generation*. The CPU load in our proposed scheme is very low during initial setup and puzzle generation. The simulation output shows that this phase takes less than 6 milliseconds.

(ii) *Low-cost verification*. Like the generation phase, our proposed verification load is very low on the AP's CPU. Hence, the simulation output shows that the time an AP has to spend to accomplish the verification phase is less than 0.4 milliseconds.

(iii) *Antireassign DoS attack*. Since both generation and verification in our scheme are very cost-effective, the proposed scheme eliminates a second DoS attack on the AP which can be posed by attack-prone security puzzles.

(iv) The memory usage of PatPaz puzzle is fixed and very small, so much so as to be almost negligible. Therefore, the proposed puzzle will never suffer from memory exhaustion.

(v) The proposed scheme also defines $T_{\exp}$ to limit the puzzle's solution life. Before $T_{\exp}$ expires, all received puzzle solutions are discarded. Hence, launching an effective DoS attack becomes more challenging.

7.1. Future Work. This paper proposes a lightweight method based on a cryptographic client puzzle. Cryptographic puzzles are very low-cost in puzzle generation and verification; however, they pose some problems. First of all, they are naturally solved within a probabilistic time. Secondly, the puzzle may be solved through parallelization. Thirdly, varying the difficulty level in this approach is too coarse. That means the difficulty of solving an $n - 1$ bit puzzle is two times less than solving an n bit puzzle. Thus it is very troublesome to design and implement an appropriate difficulty degree which increases efficiency as much as possible.

Future work can be focused on designing a new puzzle which mitigates (or even eliminates) the aforementioned problems of cryptographic client puzzles. A future study can also focus on finding a smarter mechanism to perceive and apprehend DoS attacks, in order to adjust the puzzle difficulty efficiently.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors would like to express greatest appreciation to Universiti Teknologi Malaysia (UTM) for financial support of the research done in Advanced Informatics School (AIS).

References

[1] M. Nasreldin, H. Aslan, M. El-Hennawy, and A. El-Hennawy, "WiMax security," in *Proceedings of the 22nd International Conference on Advanced Information Networking and Applications Workshops/Symposia (AINA '08)*, pp. 1335–1340, March 2008.

[2] P. H. Yu and U. W. Pooch, "A secure dynamic cryptographic and encryption protocol for wireless networks," in *Proceedings of the EUROCON 2009 (EUROCON '09)*, IEEE, St.-Petersburg, Russia, 2009.

[3] K. Bicakci and B. Tavli, "Denial-of-Service attacks and countermeasures in IEEE 802.11 wireless networks," *Computer Standards & Interfaces*, vol. 31, no. 5, pp. 931–941, 2009.

[4] K. Bicakci and B. Tavli, "Denial-of-Service attacks and countermeasures in IEEE 802.11 wireless networks," *Computer Standards and Interfaces*, vol. 31, no. 5, pp. 931–941, 2009.

[5] C. Dwork and M. Naor, *Pricing via Processing or Combatting Junk Mail*, Springer, Berlin, Germany, 1992.

[6] L. Chen, P. Morrissey, and N. Smart, "Security notions and generic constructions for client puzzles," in *Proceedings of the 15th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology (ASIACRYPT '09)*, 2009.

[7] D. Stebila, L. Kuppusamy, J. Rangasamy, C. Boyd, and J. Gonzalez Nieto, "Stronger difficulty notions for client puzzles and denial-of-service-resistant protocols," in *Topics in cryptology—CT-RSA 2011*, vol. 6558 of *Lecture Notes in Computer Science*, pp. 284–301, Springer, Heidelberg, Germany, 2011.

[8] M. Alizadeh, W. H. Hassan, M. Zamani, S. Karamizadeh, and E. Ghazizadeh, "Implementation and evaluation of lightweight encryption algorithms suitable for RFID," *Journal of Next Generation Information Technology*, vol. 4, no. 1, pp. 65–77, 2013.

[9] M. Alizadeh, J. Shayan, M. Zamani, and T. Khodadadi, "Code analysis of lightweight encryption algorithms using in RFID systems to improve cipher performance," in *Proceedings of the IEEE Conference on Open Systems (ICOS '12)*, Kuala Lumpur, Malaysia, October 2012.

[10] E. Amiri, M. Alizadeh, H. Keshavarz, M. Zamani, and T. Khodadadi, "Energy efficient routing in wireless sensor networks based on fuzzy ant colony optimization," *International Journal of Distributed Sensor Networks*, vol. 2014, Article ID 768936, 17 pages, 2014.

[11] M. Gharooni, M. Zamani, M. Mansourizadeh, and S. Abdullah, "A confidential RFID model to prevent unauthorized access," in *Proceedings of the 3rd International Conference on Information Science and Engineering*, Yangzhou, China, 2011.

[12] E. Ghazizadeh, M. Zamani, J.-L. Ab Manan, and M. Alizadeh, "Trusted computing strengthens cloud authentication," *The Scientific World Journal*, vol. 2014, Article ID 260187, 17 pages, 2014.

[13] E. Ghazizadeh, M. Zamani, J.-L. Ab Manan, and A. Pashang, "A survey on security issues of federated identity in the cloud computing," in *Proceedings of the 4th IEEE International Conference on Cloud Computing Technology and Science (CloudCom '12)*, pp. 562–565, Taipei, Taiwan, December 2012.

[14] H. Shohreh, M. Zamani, and H. Roza, "Dynamic monitoring in ad hoc network," *Applied Mechanics and Materials*, vol. 229-231, pp. 1481–1486, 2012.

[15] M. Janbeglou, M. Zamani, and S. Ibrahim, "Redirecting network traffic toward a fake DNS server on a LAN," in *Proceedings of the 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT '10)*, pp. 429–433, IEEE, Chengdu, China, July 2010.

[16] M. Janbeglou, M. Zamani, and S. Ibrahim, "Redirecting outgoing DNS requests toward a fake DNS server in a LAN," in *Proceedings of the IEEE International Conference on Software Engineering and Service Sciences (ICSESS '10)*, pp. 29–32, Beijing, China, July 2010.

[17] M. Janbeglou, M. Zamani, and S. Ibrahim, "Improving the security of protected wireless internet access from insider attacks," *Advances in Information Sciences and Service Sciences*, vol. 4, no. 12, pp. 170–181, 2012.

[18] F. Kiani, E. Amiri, M. Zamani, T. Khodadadi, and A. A. Manaf, "Efficient intelligent energy routing protocol in wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 2014, Article ID 618072, 13 pages, 2014.

[19] T. K. Araghi, M. Zamani, and A. B. T. Abdul Mnaf, "Performance analysis in reactive routing protocols in wireless mobile Ad Hoc networks using DSR, AODV and AOMDV," in *Proceedings of the International Conference on Informatics and Creative Multimedia (ICICM '13)*, pp. 81–84, Kuala Lumpur, Malaysia, September 2013.

[20] T. Koohpayeh Araghi, M. Zamani, A. Abdul Manaf, and S. Kohpayeh Araghi, "An access control framework in an Ad Hoc network infrastructure," in *Proceedings of the 1st International Conference on Communication and Computer Engineering*, Malacca, Malaysia, 2014.

[21] K. Mohebbi, S. Ibrahim, M. Zamani, and M. Khezrian, "UltiMatch-NL: a Web service matchmaker based on multiple semantic filters," *PLoS ONE*, vol. 9, no. 8, pp. 1–21, 2014.

[22] S. Nikbakhsh, A. B. A. Manaf, M. Zamani, and M. Janbeglou, "A novel approach for rogue access point detection on the client-side," in *Proceedings of the 26th IEEE International Conference on Advanced Information Networking and Applications*, pp. 684–687, Fukuoka, Japan, March 2012.

[23] H. R. Zeidanloo, A. B. Manaf, P. Vahdani, F. Tabatabaei, and M. Zamani, "Botnet detection based on traffic monitoring," in *Proceedings of the International Conference on Networking and Information Technology (ICNIT '10)*, pp. 97–101, Manila, Philippines, June 2010.

[24] H. R. Zeidanloo, M. J. Zadeh, P. V. Amoli, M. Safari, and M. Zamani, "A taxonomy of Botnet detection techniques," in *Proceedings of the 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT '10)*, pp. 158–162, Chengdu, China, July 2010.

[25] A. Sadeghian and M. Zamani, " Detecting and preventing DDOS attacks in Botnets by the help of self triggered black holes," in *Proceedings of the Asia-Pacific Conference on Computer Aided System Engineering*, Bali, Indonesia, 2014.

[26] A. Sadeghian, M. Zamani, and A. Abdul Manaf, "A taxonomy of SQL injection detection and prevention techniques," in *Proceedings of the International Conference on Informatics and Creative Multimedia (ICICM '13)*, pp. 53–56, Kuala Lumpur, Malaysia, September 2013.

[27] A. Sadeghian, M. Zamani, and S. M. Abdullah, "A taxonomy of SQL injection attacks," in *Proceedings of the International Conference on Informatics and Creative Multimedia (ICICM '13)*, pp. 269–273, Kuala Lumpur, Malaysia, September 2013.

[28] A. Sadeghian, M. Zamani, and S. Ibrahim, "SQL injection is still alive: a study on SQL injection signature evasion techniques," in *Proceedings of the International Conference on Informatics and Creative Multimedia*, pp. 265–268, Kuala Lumpur, Malaysia, September 2013.

[29] A. Sadeghian, M. Zamani, and A. A. Manaf, "SQL injection vulnerability general patch using header sanitization," in *Proceedings of the International Conference on Computer, Communication, and Control Technology*, Langkawi, Malaysia, 2014.

[30] A. Sadeghian, M. Zamani, and B. Shanmugam, "Security threats in online social networks," in *Proceedings of the International Conference on Informatics and Creative Multimedia (ICICM '13)*, pp. 254–258, Kuala Lumpur, Malaysia, September 2013.

[31] A. Hematian, S. Chuprat, A. A. Manaf, and N. Parsazadeh, "Zero-delay FPGA-based odd-even sorting network," in *Proceedings of the IEEE Symposium on Computers and Informatics (ISCI '13)*, pp. 128–131, April 2013.

[32] Z. A. Davani and A. A. Manaf, "Enhancing key management of ZigBee network by steganography method," in *Proceedings of the 2nd International Conference on Informatics and Applications (ICIA '13)*, pp. 77–81, IEEE, Łódź, Poland, September 2013.

[33] Y. Zhang, S. Wang, G. Ji, and Z. Dong, "Genetic pattern search and its application to brain image classification," *Mathematical Problems in Engineering*, vol. 2013, Article ID 580876, 8 pages, 2013.

[34] Y. Zhang, L. Wu, N. Neggaz, S. Wang, and G. Wei, "Remote-sensing image classification based on an improved probabilistic neural network," *Sensors*, vol. 9, no. 9, pp. 7516–7539, 2009.

[35] Y. Zhang, L. Wu, Y. Huoc, and S. Wang, "A novel global optimization method- Genetic pattern search," *Applied Mechanics and Materials*, vol. 44–47, pp. 3240–3244, 2011.

[36] Y. Zhang and L. Wu, "Pattern recognition via PCNN and Tsallis entropy," *Sensors*, vol. 8, no. 11, pp. 7518–7529, 2008.

[37] M. Gast, *802.11 Wireless Networks the Definitive Guide*, O'Reilly, Sebastopol, Calif, USA, 2005.

[38] H. Ordi, B. Mousavi, B. Shanmugam, M. R. Abbasy, and M. R. N. Torkaman, "A novel proof of work model based on pattern matching to prevent DoS attack," in *Communications in Computer and Information Science*, vol. 166, pp. 508–520, Springer, Berlin, Germany, 2011.

[39] J. Bellardo and S. Savage, "802.11 denial-of-service attacks: real vulnerabilities and practical solutions," in *Proceedings of the 12th Conference on USENIX Security Symposium (SSYM '03)*, Washington, DC, USA, 2003.

[40] C.-H. Liu and Y.-Z. Huang, "The analysis for DoS and DDoS attacks of WLAN," in *Proceedings of the 2nd International Conference on MultiMedia and Information Technology (MMIT '10)*, pp. 108–111, April 2010.

[41] B. Thapa, *Robust Wireless Communication in Adversarial Settings [doctoral dissertation]*, Northeastern University, Boston, Mass, USA, 2011.

[42] Q. Dong, L. Gao, and X. Li, "A new client-puzzle based DoS-resistant scheme of IEEE 802.11i wireless authentication protocol," in *Proceedings of the 3rd International Conference on BioMedical Engineering and Informatics (BMEI '10)*, pp. 2712–2716, October 2010.

[43] L. Chibiao, X. Chugui, Q. Jinming, and L. Changjing, "Experimental and theoretical study of authentication request flooding attack on 802.11 WLAN," *Energy Procedia*, vol. 13, pp. 6800–6808, 2011.

[44] R. C. Merkle, "Secure communications over insecure channels," *Communications of the ACM*, vol. 21, no. 4, pp. 294–299, 1978.

[45] G. O. Karame and S. Čapkun, *Low-Cost Client Puzzles Based on Modular Exponentiation*, vol. 6345 of *Computer Security*, Springer, 2010.

[46] B. Waters, A. Juels, J. A. Halderman, and E. W. Felten, "New client puzzle outsourcing techniques for DoS resistance," in *Proceedings of the 11th ACM Conference on Computer and Communications Security (CCS '04)*, pp. 246–256, Washington, DC, USA, October 2004.

[47] Back, "Hashcash–a denial of service counter-measure," Tech. Rep., 2002.

[48] T. Aura, P. Nikander, and J. Leiwo, "DOS-resistant authentication with client puzzles," in *Security Protocols*, vol. 2133 of *Lecture Notes in Computer Science*, pp. 170–177, Springer, Berlin, Germany, 2001.

[49] D. S. H. Rosenthal, "On the cost distribution of a memory bound function," *Computing Research Repository (CoRR)*, vol. cs.CR/0311005, pp. 1–7, 2003.

[50] C. Dwork, A. Goldberg, and M. Naor, "On memory-bound functions for fighting spam," in *Advances in Cryptology—CRYPTO 2003*, vol. 2729 of *Lecture Notes in Computer Science*, pp. 426–444, Springer, Santa Barbara, Calif, USA, 2003.

[51] S. Doshi, F. Monrose, and A. D. Rubin, "Efficient memory bound puzzles using pattern databases," *Applied Cryptography and Network Security*, vol. 3989, pp. 98–113, 2006.

[52] S. Tritilanunt, C. Boyd, E. Foo, and J. M. G. Nieto, "Toward non-parallelizable client puzzles," in *Cryptology and Network Security*, vol. 4856 of *Lecture Notes in Computer Science*, pp. 247–264, Springer, Berlin, Germany, 2007.

[53] I. Martinovic, F. A. Zdarsky, M. Wilhelm, C. Wegmann, and J. B. Schmitt, "Wireless client puzzles in IEEE 802.11 networks: security by wireless," in *Proceedings of the 1st ACM Conference on Wireless Network Security (WiSec '08)*, pp. 36–45, April 2008.

[54] Y. I. Jerschow, B. Scheuermann, and M. Mauve, "Counter-flooding: DoS protection for public key handshakes in LANs," in *Proceedings of the 5th International Conference on Networking and Services (ICNS '09)*, pp. 376–382, April 2009.

[55] M. Walfish, M. Vutukuru, H. Balakrishnan, and D. Karger, "DDoS defense by offense," in *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, New York, NY, USA, 2006.

[56] L. von Ahn, M. Blum, N. J. Hopper, and J. Langford, "CAPTCHA: using hard AI problems for security," in *Advances in Cryptology—EUROCRYPT 2003*, vol. 2656 of *Lecture Notes in Computer Science*, pp. 294–311, Springer, Berlin, Germany, 2003.

[57] Juels and J. Brainard, "Client Puzzle," EMC, 1999, http://www.rsa.com/rsalabs/staff/bios/ajuels/publications/client-puzzles/Client_puzzles.ppt.

[58] Q. Dong, L. Gao, and X. Li, "A new client-puzzle based DoS-resistant scheme of IEEE 802.11i wireless authentication protocol," in *Proceedings of the 3rd International Conference on Biomedical Engineering and Informatics (BMEI '10)*, pp. 2712–2716, October 2010.

[59] Q. Tang and A. Jeckmans, *On Non-Parallelizable Deterministic Client Puzzle Scheme with Batch Verification Modes*, Centre for Telematics and Information Technology, University of Twente, 2010.

[60] T.-J. Shi and J.-F. Ma, "Design and analysis of a wireless authentication protocol against DoS attacks based on Hash function," in *Aerospace Electronics Information Engineering and Control*, 2006.

[61] Q. Dong, L. Li, and X. Li, "Quadratic residue based client puzzle distributed by beacon frame in dos-resistant wireless access authentication," *Advances in Information Sciences and Service Sciences*, vol. 3, no. 11, pp. 79–86, 2011.

[62] W.-C. Feng, E. Kaiser, and A. Luu, "The design and implementation of network puzzles," in *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '05)*, pp. 2372–2382, Miami, Fla, USA, March 2005.

[63] Y. Lei, S. Pierre, and A. Quintero, "Client puzzles based on quasi partial collisions against DoS attacks in UMTS," in *Proceedings of the IEEE 64th Vehicular Technology Conference (VTC '06)*, pp. 1–5, IEEE, Montreal, Canada, September 2006.

[64] R. L. Rivest, A. Shamir, and D. A. Wagner, "Time-lock puzzles and timed-release crypto," Tech. Rep. MIT/LCS/TR-684, MIT Laboratory for Computer Science, Cambridge, Mass, USA, 1996.

[65] D. Hofheinz and D. Unruh, "Simulatable security and polynomially bounded concurrent composability," in *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 169–182, Berkeley/Oakland, Calif, USA, May 2006.

[66] Y. I. Jerschow and M. Mauve, "Secure client puzzles based on random beacons," in *NETWORKING 2012*, vol. 7290 of *Lecture Notes in Computer Science*, pp. 184–197, Springer, Berlin, Germany, 2012.

[67] J. Rangasamy, D. Stebila, C. Boyd, and J. G. Nieto, "Efficient modular exponentiation-based puzzles for denial-of-service protection," in *Proceedings of the International Conference on Information Security and Cryptology (ICISC '11)*, 2011.

[68] L. Kuppusamy, J. Rangasamy, D. Stebila, C. Boyd, and J. G. Nieto, "Practical client puzzles in the standard model," in *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security (ASIACCS '12)*, pp. 42–43, May 2012.

[69] V. Shoup, "Sequence of games: a tool for taming complexity in security proofs," Tech. Rep. 2004/332, IACR Cryptology ePrint Archive, 2004.

[70] M. Abliz and T. Znati, "A guided tour puzzle for denial of service prevention," in *Proceedings of the 25th Annual Computer Conference Security Applications (ACSAC '09)*, pp. 279–288, December 2009.

[71] B. Groza and B. Warinschi, "Cryptographic puzzles and DoS resilience, revisited," *Designs, Codes and Cryptography*, vol. 73, no. 1, pp. 177–207, 2014.

[72] J. Patarin and A. Montreuil, "Benes and Butterfly schemes revisited," in *Proceedings of the 9th International Conference on Information Security and Cryptology (ICISC '05)*, vol. 3935, pp. 92–116, 2006.