International Journal on
Advanced Science
Engineering
Information Technology

Vol.9 (2019) No. 1
ISSN: 2088-5334

Dynamic Message Puzzle as Pre-Authentication Scheme in Wireless Sensor Networks

Farah Afianti [#], Wirawan [#], Titiek Suryani [#]

[#]*Department of Electrical Engineering, Faculty of Electrical Technology, Institut Teknologi Sepuluh Nopember, Surabaya, 60111, Indonesia*
E-mail: farah.afianti16@mhs.ee.its.ac.id; wirawan@ee.its..ac.id; titiks@ee.its.ac.id

***Abstract*— Denial of Service (DoS) is a type of attack that has a huge impact on a computer system. This can deplete and shorten the lifetime of wireless sensor networks (WSNs). Signature-based DoS is a kind of DoS attack that exploits the high computation of a public key cryptography based authentication. The adversaries have the opportunity to send a large number of a fake signature to the WSNs. Message Specific Puzzle (MSP) was developed to defend against this type of attack. This scheme utilizes a hash function as an irreversible method to create a puzzle and produce a session key. Furthermore, this has low complexity in the sender and receiver for construction and verification process. However, the sender-side delay occurred. The higher the security expected for the system leads to the more time is needed for the user to send messages. The number of hash iteration in the puzzle construction cannot be controlled. This paper proposes the Dynamic Message Puzzle scheme that uses the power of first quartile (Q1power1) and the exponential of second quartile (Q2exp) threshold functions. These limit the maximum number of hash iterations for each puzzle construction. Consequently, this mechanism can decrease sender-side delay by at least 60%. Besides avoiding zero solution and has a high value of mean absolute deviation, this scheme also increases the adversaries' complexity in attacking the system. The proposed scheme transmits index implicitly. This obscures the portion of each parameter in the transmitted packet.**

***Keywords*—broadcast authentication; pre-authentication; wireless sensor networks; signature based DoS; puzzle scheme.**

I. INTRODUCTION

The commonly used communications between nodes in wireless sensor networks (WSNs) are broadcast [1], [2]. This communication is efficient but lacks protection against denial of service (DoS) attacks. Such attacks can have a huge impact on a computer system by sending large numbers of false messages. This makes the system so busy verifying received packets that authenticated users cannot access the attacked sensor node. Apart from that, it can reduce WSN lifetime because node sensors are exhausted by verifying all the fake messages.

A signature-based DoS attack is an action that sends a large number of fake signatures [3]. The main target of this attack is the high complexity of digital signature verification, which uses more energy than receiving messages [3]. Therefore, additional protection is needed to accompany public key cryptography based authentication.

Several filtering methods have been developed against signature-based DoS attacks [3], [7]–[9], [11]–[13]. In 2000, Aura et al. developed a mechanism to resist DoS attacks in client-server environments [11]. This method, called MSP, was improved to allow for implementation in WSNs [3]. Instead of a puzzle, Du et al. used several key chains to represent each network user [7]. This approach is not scalable, because the number of key chains increases with the number of connected users. Later, Chuchaisri et al. developed key-pool and key-chain schemes [12]. These mechanisms incorporate the Bloom Filter Vector as the process of membership verification. They still have limitations in handling false positive packets.

Furthermore, they use a forwarding key chain that is weak against compromised keys. In 2013, Dong et al. developed three filtering methods, namely a group-based filter, a keychain-based filter, and a hybrid filter [8]. The security performance of the group-based filter was less good due to the number of compromised nodes was high. The keychain-based filter only had better security if the number of legitimate packets was low. Also, the computation overhead for the key-chain based filter was higher than for the group-based filter. The hybrid filter combined both filters in order to reduce security limitations, but its implementation had high complexity. In the same year, Tan et al. added confidentiality and constructed a cipher puzzle mechanism for advertisement packets [9]. It has high security because every packet contains an encrypted message. However, this means it has high complexity, especially for resource-constrained devices. In 2016, Kim et al. built a mechanism that randomly drops received packets based on sensor node capacity, which depends on reservoir sampling as the packet selection algorithm. However, this method could drop legitimate messages as false harmful packets and forward fake messages as false positive packets. Among these filtering methods, MSP has the lowest complexity in the receiver verification process [14]. It only needs two hash operations to verify the puzzle solution and session key. Furthermore, no false positive or false negative packets can be received. Therefore, the Message Specific Puzzle (MSP) [3] is one of the most promising pre-authentication methods for unencrypted messages.

MSP can be used in any application, especially the Internet of Things applications that exchange plaintext messages, to avoid signature-based DoS attacks [4]–[6]. This mechanism acts as a filter of the main digital signature. It only uses two hash function operations in the verification process. It has low computation complexity and is appropriate for the characteristics of sensor nodes. However, a drawback is a sender-side delay [7], [8]. The more security expected for the system, the higher the number of hash iterations needed to produce a puzzle solution. The higher number of hash iterations increases the delay or time needed for processing on the sender side.

Furthermore, the pattern content and puzzle strength for each packet are fixed [7], [9], [10] so that adversaries can use copied packets as the sender. In order to resolve this limitation, this paper proposes a Dynamic Message Puzzle, which uses a threshold function. The pattern content is zero, but the length of the pattern or the puzzle strength is dynamic for each transmitted packet. The objective of this method is to control the sender-side delay by decreasing the puzzle strength if it exceeds a threshold value. We constructed a tag that consists of an implicit value for the index, the current and the previous puzzle strength. Its length is dynamic. The process of tagging obscures the index and puzzle strength. Also, it increases the attacker's computing complexity in finding the puzzle solution.

The following are two of the main contributions in this paper:

- The development of dynamic puzzle strength in the Message Specific Puzzle (MSP) to decrease sender-side delay while ensuring that there is a solution to each constructed puzzle.
- It is obscuring packet transmission. The scheme does not send the index explicitly. It increases the computing complexity of the adversary by making it difficult to distinguish between the indexes and puzzle strength values.

This paper is organized as follows: Section II briefly discusses the preliminary research and the proposed methods. The preliminary reviews the keychain that is used as a session key generator and MSP mechanism. Section III explains the design of the experiment, the result and its discussion including the security analysis of the proposed method. The concluding section summarizes this paper's contribution.

II. Material and Method

This section provides a discussion of the MSP as the preliminary studies and the DMP as the proposed methods.

A. MSP overview

MSP was developed as a weak algorithm that acts as an addition to the main signature in order to counter DoS attacks [3]. This scheme is implemented in WSN environments. It consists of three main steps: one-way keychain as a method to generate a session key, digital signature creation, and puzzle solution construction.

1) One-Way Keychain

The one-way keychain is utilized to produce the session key in each transmitted packet. This mechanism blocks a fake message from the adversaries [3]. If the session key is not valid, then the packet will be dropped. Therefore the next steps cannot be continued. The limitation of this mechanism is the finite value of the key number that must be declared in the initialization phase [14].

A hash function is utilized as the basic algorithm for a one-way keychain. This starts with generating random value that is used to fill the last keychain. The rest of the key set is calculated using Equation (1) [7]:

$$K_n = F_H(K_{n+1}) \qquad (1)$$

The calculation is stopped while the expected number of key is fulfilled. The overall session key that utilized by the system is formulized by Equation (2).

$$Keychain = \bigcup_{i=1}^{N_{key}-1} F_H{}^{i}(x) \bigcup x \; ; x \in random \qquad (2)$$

Commitment key is the name for initial session key that is transmitted to the receiver. This parameter is utilized for the verification process.

2) Digital Signature

The high computation of digital signature generation provides challenge especially for the sensor nodes. This drawback is minimized by limiting the number of signature or modifies the main algorithm. The successful implementation of the public key cryptography (PKC) based authentication in the resource constraint device was presented [15]–[18].

MSP utilizes the Elliptic Curve Digital Signature Algorithm (ECDSA) as the main signature generation [3]. As a part of Elliptic-Curve Cryptography (ECC) [19], this method is suitable for sensor nodes environments [20]. It is because of the efficient length of the key. The ECDSA's key length is 40 bytes. This can produce signature that has length 40 bytes while RSA requires 128 bytes [21]. The key length of RSA is longer than ECDSA.

3) Puzzle Solution

The last step before transmitting the packet is puzzle generation. The aim is looking for the hash result that has *L* bits consecutive zeroes value. It begins with a concatenation process, which aggregates several sent parameters and then hashes them. The output of the concatenation process is compared with the pattern. The MSP pattern has zero value and has a length of L bits. It is expressed in Equation (3):

$$F_H\left(index \,\|\, M \,\|\, Sig \,\|\, K_{idx} \,\|\, P_{idx}\right) = \underbrace{000...}_{L} XXX \ldots \qquad (3)$$

B. *Dynamic Message Puzzle*

This paper proposes a Dynamic Message Puzzle (DMP), which uses a threshold function. It can control the number of hash iterations so that sender-side delay can be reduced. Furthermore, dynamic puzzle strength increases the attacker's complexity. The detailed of the system is presented as a block diagram that is shown in Fig. **1**.

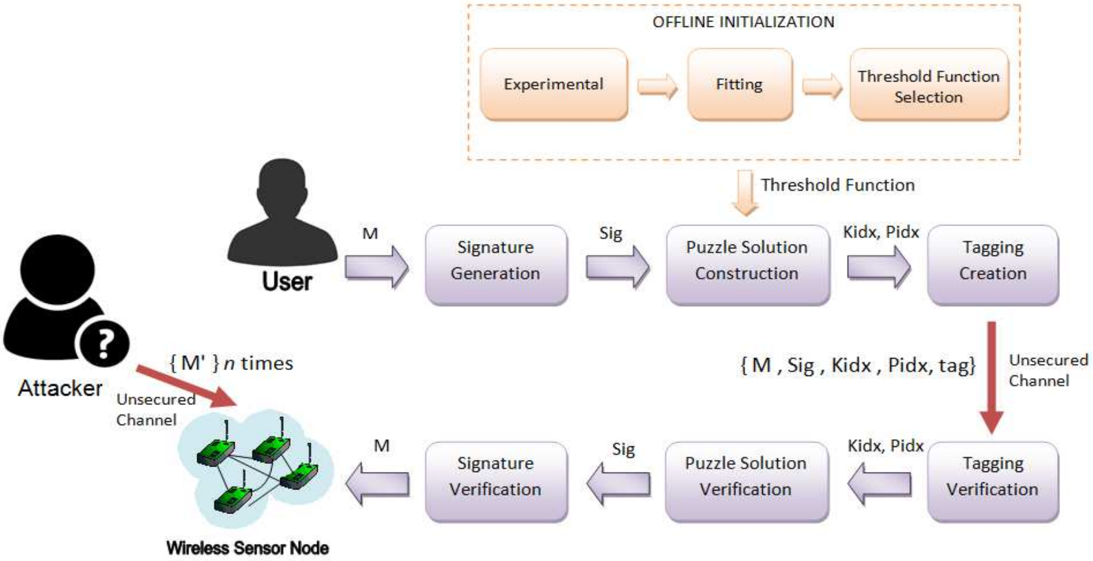


Fig. 1 Block diagram of the proposed method.

The step starts with the offline initialization. This aims to generate the threshold function that is used in the real system. This step starts with an experimental process. Then, the value from its result is used as a starting point in the fitting process. The function from the fitting process is selected from some candidates based on certain parameters.

Through an experimental process, values from the MSP puzzle generation activity were gathered. This process was repeated until a small sample was gathered for each different condition. The number of repetitions was set to 60 as a representation of the expected overall system behavior. An illustration of this mechanism is shown in Fig. **2**.

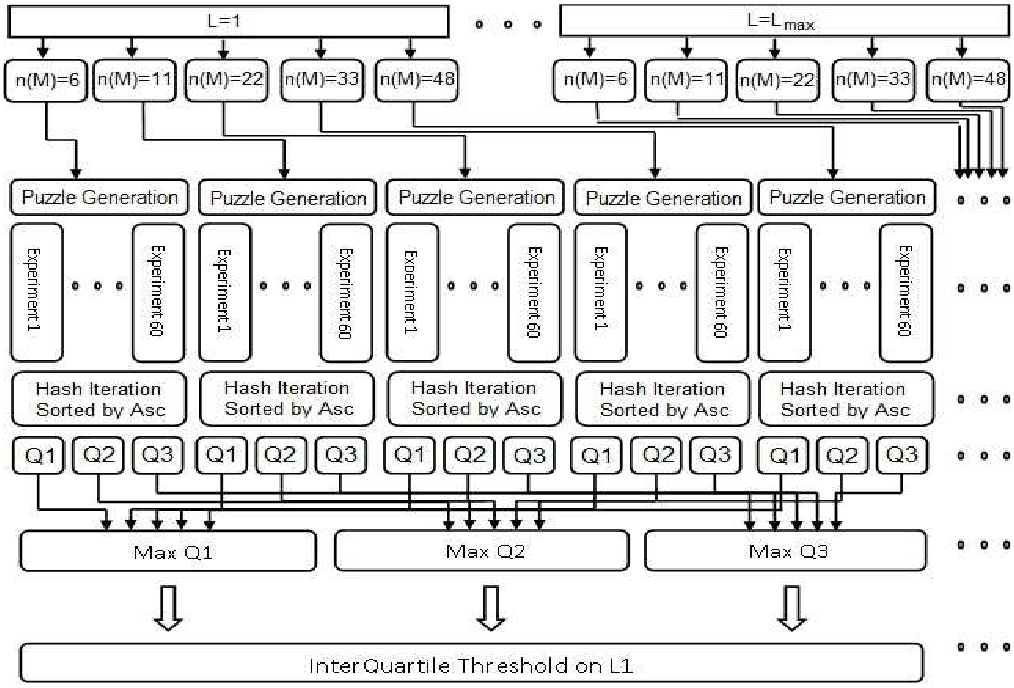


Fig. 2 Diagram of Experimental process.

The statistic information on the characteristics of MSP is required to be analyzed. More specifically, the detailed information about the number of hash iterations required for each puzzle strength (*L*) variant. The average value of the number of hash iterations in each puzzle strength is 2^L [3]. However, this paper aims to reduce this value because therefore sender-side delay can be decreased. The quartiles are utilized instead of the mean value. There are three quartiles, each of which represents part of the data.

Another sent parameter that has variable length is Message. We observed five message length variants, i.e., 6, 11, 22, 33 and 48 bits and chose to represent 2 variant blocks in the hash function. MSP uses SHA1. Its blocks are multiplications of 512 bits. The particular positions for each message length variant can be seen in Fig. 3.

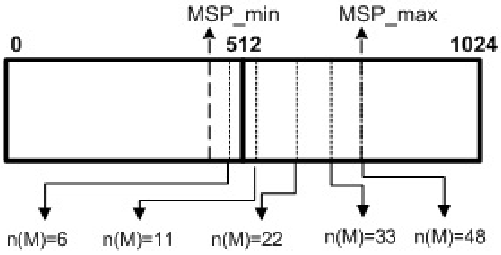


Fig. 3 Padding process on SHA1.

Hence, the possible value of the first quartile threshold value for each puzzle strength (L) was five, as the number of message variants. We had to select one of them to represent all message variants. In this work, we chose the maximum value because this could cover all message length variants. The same condition occurred in the second and third quartile. Furthermore, the value of the quartile threshold in each puzzle strength can be collected from the experimental process.

The second step of the initialization process is fitting. This is aimed at creating a mathematical model in the form of a function from the experimental values. The values collected for each puzzle strength are fitted in a curve equation. A detailed illustration is shown in Fig. 4.

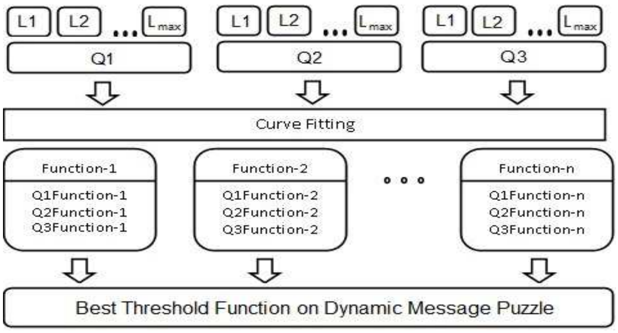


Fig. 4 Fitting process.

For simplicity, the curve was fitted using MATLAB 2015a. The function consists of an independent variable, i.e., puzzle strength $\{L : 1 \le L \le L_{\max}, L \in Z\}$, and a dependent variable, i.e., the number of hash iterations. Moreover, there are several functions that approximate experimental values.

Several parameters can be used to calculate the function that is appropriate to represent the threshold value from the experiment. We selected two parameters. First, the coefficient of determination, or R-squared, in a range from zero to one. This compares the variants based on the original and the estimated values. The high R-squared values reflect the estimated values that are close to their original or actual value. It means that the highest value of R-squared, one, indicates that the predicted value is equal to the original value. Moreover, this parameter has been tested to measure the exponential family regression model [22]. This is formulated in Equation (4), the most general form of R-squared:

$$R^2 \equiv 1 - \frac{\sum_{i=1}^{n} (Y_{[i]} - X_{[i]})^2}{\sum_{i=1}^{n} (Y_{[i]} - \bar{Y})^2} \quad (4)$$

Where: $X_{[i]}$ denotes actual value in i th trial, $Y_{[i]}$ means estimated value in i th trial and $\bar{Y}$ describes the average of the estimated value.

Secondly, the root means square error (RMSE) is calculated. This value is often used to measure the deviation from original and estimated values from a population, as formulated in Equation (5):

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (X_{[i]} - Y_{[i]})^2}{n}} \quad (5)$$

Based on these two parameters, there were several functions as candidate threshold functions that fit well enough to the values from the experimental process. These functions cannot be used directly by the system. It is important to find the best threshold function that is appropriate for MSP puzzle generation. Therefore, three parameters were added to analyze threshold functions behavior. The details of these parameters are given in TABLE I.

TABLE I
PARAMETER SELECTION FOR THE BEST THRESHOLD FUNCTION

Parameter	Description
nH	Information about delay on the sender side
MAD_L	Mean of the distance in a set of L to the maximum value (L_{max})
Puzzle Solution Probability	Information whether puzzle scheme has a solution

The number of hash iterations (*nH*) represents the delay on the sender side. The hash function has low complexity. This is different when the function is repeated for about 2^L times, in which case there will be a delay. Furthermore, the mean absolute deviation of L (MAD_L) is used to count the distance from L to the maximum puzzle strength (L_{max}). Although it is dynamic, the puzzle strength value is expected to approach the maximum puzzle strength. It measures the mean of the distance between other points to the central point. In our case, the central point is the maximum puzzle strength. For this goal Equation (6) is used:

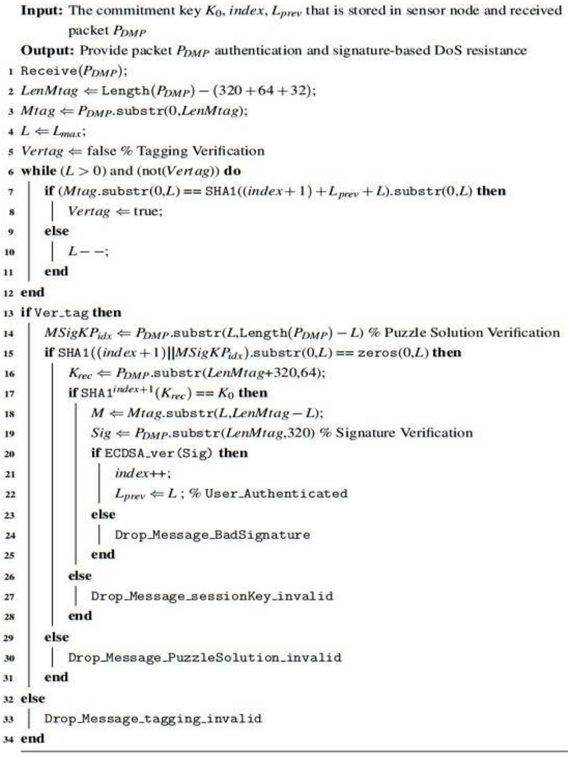
$$MAD_L = \frac{\sum_{i=1}^{n} |L_i - L_{max}|}{n} \quad (6)$$

If the hash iteration is confined, the zero solution may occur. Therefore, the puzzle solution probability must be measured. The probability of finding a solution in *nH* iterations for puzzle strength L is expressed in Equation (7) [3]:

$$P_{(n_H,L)} = 1 - \left(1 - \left(\frac{1}{2}\right)^{L}\right)^{n_H} \tag{7}$$

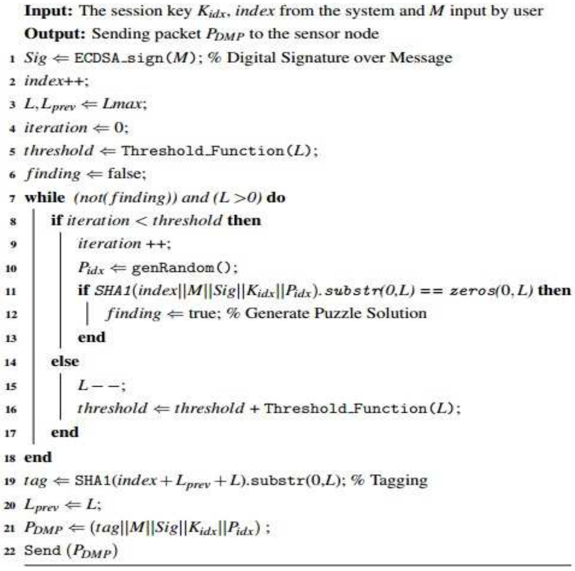
The communication from the user to the sensor node can be started after getting the best threshold function. The detailed process of real communication in Figure 1 is expressed by Algorithm 1 and Algorithm 2 that is implemented in the sender and receiver side.

Algorithm 1 DMP Verification on the Receiver Side



```
Input: The commitment key $K_0$, $index$, $L_{prev}$ that is stored in sensor node and received
       packet $P_{DMP}$
Output: Provide packet $P_{DMP}$ authentication and signature-based DoS resistance
1  Receive($P_{DMP}$);
2  $LenMtag \Leftarrow$ Length($P_{DMP}$) $-(320+64+32)$;
3  $Mtag \Leftarrow P_{DMP}$.substr(0,$LenMtag$);
4  $L \Leftarrow L_{max}$;
5  $Vertag \Leftarrow$ false % Tagging Verification
6  while ($L > 0$) and (not($Vertag$)) do
7      if ($Mtag$.substr(0,$L$) == SHA1(($index+1$) $+ L_{prev} + L$).substr(0,$L$) then
8          $Vertag \Leftarrow$ true;
9      else
10         $L--$;
11     end
12 end
13 if Ver_tag then
14     $MSigKP_{idx} \Leftarrow P_{DMP}$.substr($L$,Length($P_{DMP}$) $- L$) % Puzzle Solution Verification
15     if SHA1(($index+1$)||$MSigKP_{idx}$).substr(0,$L$) == zeros(0,$L$) then
16         $K_{rec} \Leftarrow P_{DMP}$.substr($LenMtag$+320,64);
17         if SHA1$^{index+1}$($K_{rec}$) == $K_0$ then
18             $M \Leftarrow Mtag$.substr($L$,$LenMtag - L$);
19             $Sig \Leftarrow P_{DMP}$.substr($LenMtag$,320) % Signature Verification
20             if ECDSA_ver(Sig) then
21                 $index$++;
22                 $L_{prev} \Leftarrow L$ ; % User_Authenticated
23             else
24                 Drop_Message_BadSignature
25             end
26         else
27             Drop_Message_sessionKey_invalid
28         end
29     else
30         Drop_Message_PuzzleSolution_invalid
31     end
32 else
33     Drop_Message_tagging_invalid
34 end
```

Algorithm 2 DMP Constructor on the Sender Side



```
Input: The session key $K_{idx}$, $index$ from the system and $M$ input by user
Output: Sending packet $P_{DMP}$ to the sensor node
1  $Sig \Leftarrow$ ECDSA_sign($M$); % Digital Signature over Message
2  $index$++;
3  $L, L_{prev} \Leftarrow Lmax$;
4  $iteration \Leftarrow 0$;
5  $threshold \Leftarrow$ Threshold_Function($L$);
6  $finding \Leftarrow$ false;
7  while (not(finding)) and ($L$ >0) do
8      if $iteration < threshold$ then
9          $iteration$ ++;
10         $P_{idx} \Leftarrow$ genRandom();
11         if $SHA1(index||M||Sig||K_{idx}||P_{idx}).substr(0,L) == zeros(0,L)$ then
12             $finding \Leftarrow$ true; % Generate Puzzle Solution
13         end
14     else
15         $L--$;
16         $threshold \Leftarrow threshold +$ Threshold_Function($L$);
17     end
18 end
19 $tag \Leftarrow$ SHA1($index + L_{prev} + L$).substr(0,$L$); % Tagging
20 $L_{prev} \Leftarrow L$;
21 $P_{DMP} \Leftarrow (tag||M||Sig||K_{idx}||P_{idx})$ ;
22 Send ($P_{DMP}$)
```

The activity on the sender side starts with signing the message using ECDSA (line 1 of Algorithm 1). The default value of the current and the previous puzzle strength (if the index is equal to 1 or the first packet is sent) is the maximum puzzle strength (line 3 of Algorithm 1). This value is set to 22 bits. This is an acceptable number [3] regarding the amount of delay. The threshold variable contains the highest number of hash iterations for L-bit puzzle strength (line 5 of Algorithm 1).

Meanwhile, the number of hash iterations for finding the puzzle's solution is compared with the threshold value (line 8 of Algorithm 1). The puzzle strength will be decreased, and the threshold value recomputed only if the hash iterations surpass the threshold value. This will stop when the puzzle is found, or zero solution (no solution) is reached.

The last activity in the Dynamic Message Puzzle scheme is tagging. The first step is the summation of the index, and the previous and current puzzle strength (line 19 of Algorithm 1). The tag is the output of the hashing function for the summation and is cut into the first L bits. The last activity in Algorithm 1 is sending the packet, which consists of Message, Signature, Session Key, Puzzle Solution, and Tag. The length of each value is shown in TABLE.

TABLE II
PARAMETERS OF PACKET

Parameter	Length (bits)
Tag	1, 2, 3, ..., *L*
M	1, 2, 3,...., 384
Sig	320
K_{idx}	64
P_{idx}	32

The total length of the sent packet is 102 bytes referring to the IEEE standard for 802.15.4 [23]. Sending the index explicitly is avoided. Instead, the tag is used, which contains index value implicitly. This also reduces communication overhead and increases the attacker's complexity by obscuring the values of the index and the current and previous puzzle strength by using a hash function.

The base station as the first receiver broadcasts the packet to the node sensor as the actual receiver. The activity at the receiver's side starts with tag verification (line 6 of Algorithm 2). The sensor node as the receiver tries the possible puzzle strength value based on the stored index and puzzle strength. The highest number of trials for each received packet is equal to L_{max}. First, it tries the maximum puzzle strength as the current puzzle strength. If this fails, then the puzzle strength is decreased by one. The result of this process is that either the current puzzle strength is larger than zero (valid tag) or the current puzzle strength is zero (invalid tag). The process is continued with puzzle solution verification (line 14 of Algorithm 2) only if the tag is verified. This next process verifies whether the hash function result is matched with L bits of zeroes. The input parameter of the hash function is the concatenation of the stored index and other parameters from the received packet (line 15 of Algorithm 2). If this is the case, then the puzzle solution is valid, and the algorithm continues to session key verification (line 17 of Algorithm 2). This mechanism is part of the puzzle solution verification. It aims to avoid fake puzzle with the same consecutive zero patterns. The sensor node repeats the hash function ($index$+1) times on the received session key. After that, it has to check whether the result is the same as the commitment key it has stored. If it is the same, then the algorithm continues to the last verification step, i.e., ECDSA verification (line 20 of Algorithm 2). This process needs information about the public key of the sender, message, and signature it has received. The details of this process can be found in [19].

III. Results And Discussion

This section contains a discussion of the experimental design including its result and security analysis.

A. *Experimental Design*

We implemented our proposed scheme using Network Simulator-3 (NS-3), version 3.20 with a WSN module. The detailed network construction for our scheme is displayed in Fig. **5**

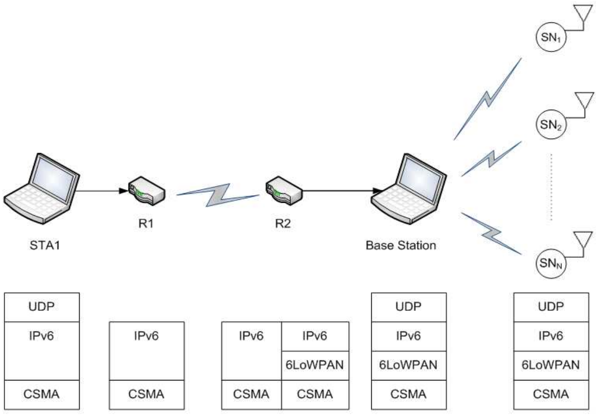


Fig. 5 Network topology for simulation

The sender-side consists of Administrator, namely STA1, and the R1 router. The receiver-side consists of the R2 router, base station, and some sensor nodes. There is no fixed number for the number of sensor nodes; it can be increased to fulfill the requirement. Furthermore, the data rate from the sender's router to the base station is 5 Mbps because it is operated through the cloud. In contrast, the data rate from the base station to the wireless sensor nodes is 250 kbps as it is a resource-constrained device.

The testing scenario for communication between the sender and the wireless sensor network was divided into five types based on the contents of the message. Details of the contents of each message can be seen in TABLE III.

TABLE III
MESSAGE CONTENTS

Message Sample	**Length (bits)**
silent	6
so shutdown	11
save key delete repair	22
recovery share send add max power	33
wireless sensor network stop-start shutdown play	48

B. *Performance Analysis*

From the offline initialization, we got the experimental values as shown in Fig. 6. These values are the maximum numbers for each variant of message length. The higher the puzzle strength, the higher the number of hash iterations in each quartile.

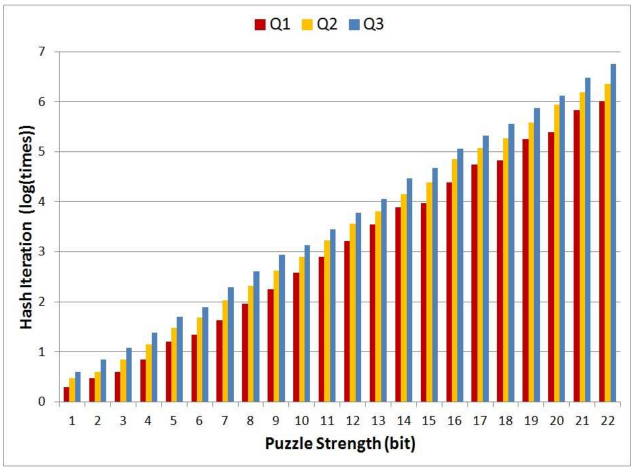


Fig. 6 Quartiles of hash iteration values in MSP from experimental

Based on the quartile value from the experimental process, we got candidate mathematical models for each quartile. The functions consisted of an independent variable, i.e., puzzle strength ranging from 1 to 22, and a dependent variable, i.e., the number of hash iterations. Two types of functions are appropriate in this case, i.e., power functions and exponential functions. Furthermore, we used two power functions to differentiate the puzzle strength used. The first power function used the constant value as the base and the puzzle strength variable as the exponent. The second power function used the puzzle strength variable as the base and the constant value as the exponent. The last function was an exponential function followed by constant multiplication. TABLE shows the particular functions for each quartile and the type of function, followed by the measurement to find the best approximate function.

TABLE IV
THRESHOLD FUNCTIONS

Threshold Function Name	R-squared	RMSE
Q1power1 = $\lceil 1.8786^{L} + 1726 \rceil$	0.9874	2.8085e+04
Q1power2 = $\lceil L^{12.4} \times (2.412e - 11) \rceil$	0.9892	2.5977e+04
Q1exp = $\lceil e^{(L\times 0.5997)} \times 1.966 \rceil$	0.9881	2.7198e+04
Q2power1 = $\lceil 1.95^{L} + (2.373e + 04) \rceil$	0.9760	8.7320e+04
Q2power2 = $\lceil L^{10.78} \times (7.754e - 09) \rceil$	0.9939	4.4053e+04
Q2exp = $\lceil e^{(L\times 0.5282)} \times 20.86 \rceil$	0.9912	5.3044e+04
Q3power1 = $\lceil 2.018^{L} + (6.448e + 04)$	0.9861	1.5488e+05
Q3power2 = $\lceil L^{14.35} \times (3.147e - 13) \rceil$	0.9987	4.6620e+04
Q3exp= $\lceil e^{(L\times 0.688)} \times (1.533) \rceil$	0.9989	4.4073e+04

It can be seen that the second power function had the best value for both parameters. It had the highest R-squared approximating one and the lowest RMSE. Furthermore, the first power function had the lowest value of R-Squared and the highest RMSE among the three types of threshold functions. The three-second power functions were the best three fitted functions. However, two parameters are not enough to justify the assumption that the best-fitted function is also the best threshold function. This needs more analysis based on other aspects because the deviation between the lowest (L = 1) and the highest (L = 22) puzzle strength in our case was very high. Especially the second power function tended to miss lower values of puzzle strength to be fitted compared to higher values, so that it had the highest estimated value under the original value among the three functions, even though a lower estimated value leads to zero solution.

For further analysis, the average of hash iterations for each function is computed. The average of hash iterations value for each length message variant can be found in Fig. **7**, which shows a comparison between the maximum puzzle strengths of MSP (MSP_L=22) and DMP with threshold function variants.

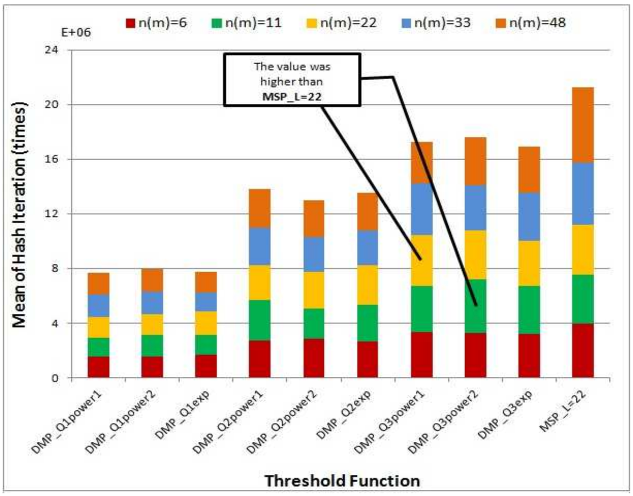


Fig. 7 The average of hash iterations in MSP and DMP.

A higher number of hash iterations mean a more sender-side delay in sending messages. The higher the quartile number, the higher the average value of hash iterations. DMP_Q1power1 had the lowest average value of hash iterations, followed by DMP_Q1exp and DMP_Q1power2 respectively. The average value of hash iterations for 22-bit puzzle strength was 2^{22}, or equal to 4194304. The highest average value of hash iterations in the first quartile (Q1) of DMP was about 1707555 for 33-bit messages of DMP_Q1power2, which is a decrease of about 60%. The DMP_Q2power2 had the lowest average value of hash iterations in the second quartile of DMP threshold functions, followed by DMP_Q2exp and DMP_Q2power1 respectively. Also, the overall value of DMP_Q3 was lower than the MSP_L=22. However, the mean numbers of hash iterations of DMP_Q3power2 for 11-bit messages and DMP_Q3power1 for 22-bit messages were higher than for MSP_L=22. This shows that the delay in the third quartile approached the original MSP. This is the reason the third quartile is not recommended for selection as the threshold value. The sender-side delay for those values is still high and could even exceed the mean number of hash iterations of the original MSP.

Another consideration for choosing the best threshold function is the dynamic puzzle strength. We need to analyze whether the threshold function leads to zero solution. We utilized Equation (7) and the sum of hash iterations from the output of the threshold function to calculate the probability of finding the puzzle solution, as presented in Fig. 8.

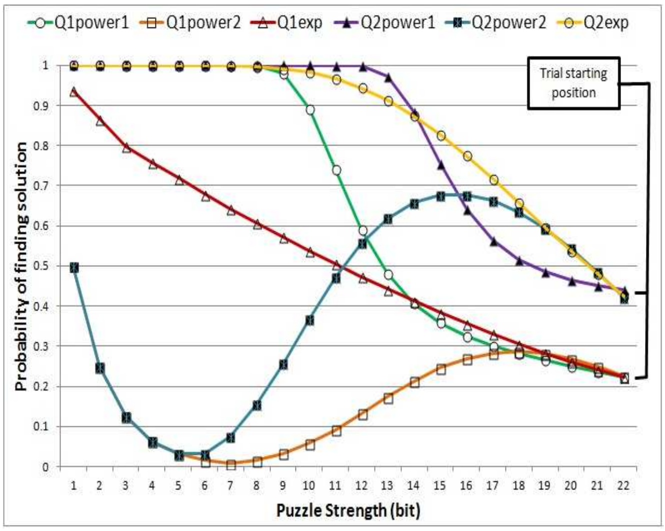


Fig. 8 Probability of finding puzzle solution with DMP.

Based on their quartiles, it can be seen that 22-bit puzzle strength is grouped into two. The higher value of the quartile means the higher value of the finding solution probability at 22-bit puzzle strength. The probability for the first quartile at 22-bit puzzle strength was about 0.22, while the probability for the second quartile was about 0.43. The threshold function with the highest probability of finding a solution was Q2exp, followed by Q2power1 and Q2power2 respectively.

In the first quartile, the second power threshold function (Q1power2) had a high probability of leading to zero solution when approaching 8-bit puzzle strength. This was proved by the result of the dynamic puzzle strength experiment, as shown in Fig. **10**.

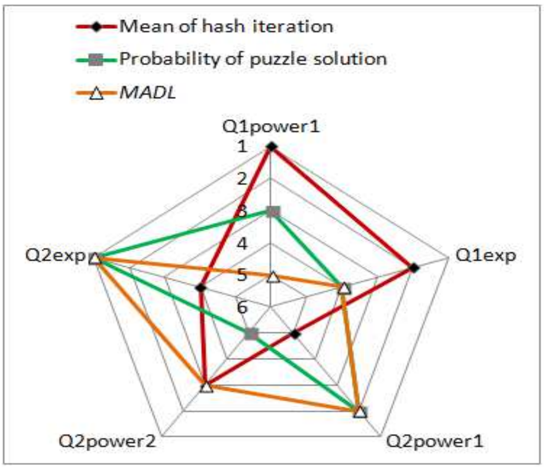


Fig. 9 Threshold function summary.

Almost all of the message length variants using DMP_Q1power2 approached zero solution. The use of the threshold function became insignificant when the Q1power2 threshold function was used, so it is not recommended for selection as the threshold value.

There were five functions left for threshold function consideration. We analyzed the distance between puzzle strength and maximum puzzle strength using MAD_L. The result is rounded up and shown in Fig. **11**.

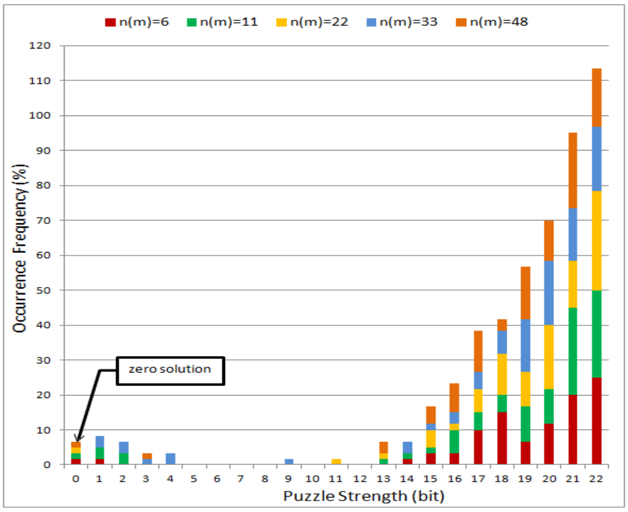


Fig. 10 Frequency of puzzle strength occurrence for DMP_Q1power2.

Q2exp had the lowest MAD_L value, which means that the puzzle strength of Q2exp was close to the maximum puzzle strength (L = 22). It was followed by Q2power2, Q2power1, and Q1exp respectively. Q1power1 reached the highest value of MADL. The candidate threshold functions' performances are summarized in the spider chart in Fig. **9**.

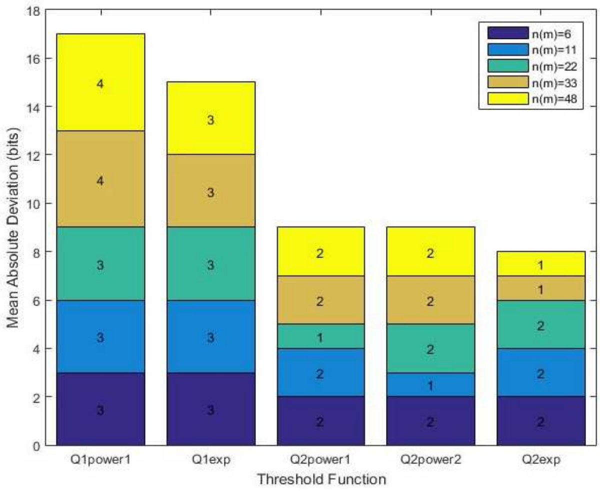


Fig. 11 MAD of dynamic puzzle strength in DMP.

Q1power1 had better performance than Q1exp for two parameters that are the average value of hash iterations and the puzzle solution probability. Also, Q2exp had better performance than the other functions for the puzzle solution probability and MAD_L. Two threshold functions are recommended. If time is critical, then DMP using Q1power1 is preferred. However, if security is a more important aspect, then DMP using Q2exp is selected.

The proposed method increases the overhead for both parties. On the sender side, there is no big impact because it only needs threshold function calculation and comparison between the threshold function and the hash iterations. However, the mechanism of puzzle strength transmission increases the storage and computation overhead on the receiver side. The index value (16 bits) and previous puzzle strength (4 bits) need to be stored, and there is the additional computation of the L hash and the ($3.L$ + 1) sum operation. This operation is needed to check whether hashing the sum of the index, previous and current puzzle strength values matches the received tag. The node sensor has to check this starting from maximum puzzle strength down to one. The computation and storage addition is still acceptable.

C. *Security Analysis*

Static puzzle strength makes it easy for adversaries to send fake puzzle solutions. Even, the puzzle strength value can be equal to the system. This is called a probability attack [7], which is designed for puzzle systems with a fixed pattern content and length, such as MSP. Since our proposed scheme uses dynamic pattern length, the attacker can flood the system with fake puzzle solutions at low length. However, the attacker has to guess the values of the index, previous puzzle strength and session key in order to break the system verification. As mentioned above, the values for index, previous and current puzzle strength are sent implicitly. That information hides behind the value of the hash function, which is sent partially.

Furthermore, the attacker has to guess the next session key from the keychain. Using the brute force, the attacker has to try 2^L times to know the strength of the puzzle solution and 2^{64} times to know the session key. As discussed above, the average value of hash iterations required for puzzle creation in DMP is under 2^L divided by two. That value is under the trials number that is needed by the attacker, i.e., $2^{(64+L)}$.

IV. Conclusions

Dynamic Message Puzzle (DMP) scheme using the Q1power1 and Q2exp threshold functions was proposed. We built a tagging mechanism to transmit index and puzzle strength implicitly. This approach can decrease sender-side delay by reducing the average of the average value of hash iterations by about 60%. Furthermore, this increases the complexity of the attacker to guess the value behind the hash function that impacts on the obscurity of the transmitted packet.

The complexity and additional storage on the receiver side are increased but still acceptable. Developing a scheme for multiple sender systems that are close to real system implementation becomes the next challenge for future research.

Nomenclature

F_H	hash function	-
K_{idx}	a session key for index *idx*	byte
L	puzzle strength	bit
nH	hash iteration	times
$\overline{nH}$	average value of hash iteration	times

N_{key}	Number of keys	-
$n(M)$	length of message	byte
P_{idx}	puzzle solution for index idx	byte
Sig	signature	byte
Subscripts		
$index$	Number of packet transmission	-

Acknowledgment

The authors would like to thank the Indonesian Endowment Fund for Education (*Lembaga Pengelola Dana Pendidikan / LPDP*), Ministry of Finance, and the Republic of Indonesia for providing a scholarship.

References

[1] A. Mahmood, H. Yiğitler, R. Virrankoski, and R. Jäntti, "Recursive clock skew estimation for wireless sensor networks using reference broadcasts," *IET Wirel. Sens. Syst.*, vol. 2, no. 4, pp. 338–350, 2012.

[2] D. R. Wijaya, R. Sarno, E. Zulaika, and S. I. Sabila, "Development of mobile electronic nose for beef quality monitoring," in *Procedia Computer Science*, 2017, vol. 124, pp. 728–735.

[3] P. Ning and A. N. Liu, "Mitigating DoS Attacks against Broadcast Authentication in Wireless Sensor Networks," *ACM Trans. Sens. Networks*, vol. 4, no. 1, pp. 1–35, 2008.

[4] S. Hyun and P. Ning, "Seluge: Secure and dos-resistant code dissemination in wireless sensor networks," in *In Information Processing in Sensor Networks, 2008. IPSN'08*, 2008, pp. 445–456.

[5] R. Zhang, J. Zhang, Y. Zhang, J. Sun, and G. Yan, "Privacy-preserving profile matching for proximity-based mobile social networking," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 9, pp. 656–668, 2013.

[6] T. Kothmayr, C. Schmitt, W. Hu, M. Brünig, and G. Carle, "DTLS based security and two-way authentication for the Internet of Things," *Ad Hoc Networks*, vol. 11, no. 8, pp. 2710–2723, 2013.

[7] X. Du and H. Chen, "Defending DoS Attacks on Broadcast Authentication in Wireless Sensor Networks," in *2008 IEEE International Conference on Communications*, 2008, pp. 1653–1657.

[8] Q. Dong, D. Liu, and P. Ning, "Providing DoS resistance for signature-based broadcast authentication in sensor networks," *ACM Trans. Embed. Comput. Syst.*, vol. 12, no. 3, pp. 1–26, 2013.

[9] H. Tan, D. Ostry, J. Zic, and S. Jha, "A confidential and DoS-resistant multi-hop code dissemination protocol for wireless sensor networks," *Comput. Secur.*, vol. 32, pp. 36–55, 2013.

[10] D. He, S. Chan, and M. Guizani, "Cyber Security Analysis and Protection of Wireless Sensor Networks for Smart Grid Monitoring," *IEEE Wireless. Commun.*, vol. PP, no. 99, pp. 2–7, 2017.

[11] T. Aura, P. Nikander, and J. Leiwo, "DOS-resistant authentication with client puzzles," in *In International workshop on security protocols*, 2000, pp. 170–177.

[12] P. Chuchaisri and R. Newman, "Fast response PKC-based broadcast authentication in wireless sensor networks," *Mob. Networks Appl.*, vol. 17, no. 4, pp. 508–525, 2012.

[13] D. Kim, S. Member, and S. An, "PKC-based DoS Attacks-Resistant Scheme in Wireless Sensor Networks," *IEEE Sens. J.*, vol. 16, no. 8, pp. 2217–2218, 2016.

[14] F. Afianti, Wirawan, and T. Suryani, "Filtering methods for broadcast authentication against PKC-based denial of service in WSN: a survey," in *Fifth International Conference on Wireless and Optical Communications*, 2017, vol. 10465, p. 1046503.

[15] A. Liu and P. Ning, "TinyECC : A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks," in *Proceedings of the 7th international conference on Information processing in sensor networks*, 2008, pp. 245–256.

[16] G. De Meulenaer, F. Gosset, F. X. Standaert, and O. Pereira, "On the Energy Cost of Communication and Cryptography in Wireless Sensor Networks," in *WIMOB'08 IEEE International Conference on Wireless and Mobile Computing*, 2008, pp. 580–585.

[17] M. Sethi, J. Arkko, and A. Keranen, "End-to-end Security for Sleepy Smart Object Networks," in *IEEE 37th Conference on Local Computer Networks Workshops (LCN Workshops)*, 2012, pp. 964–972.

[18] A. Xu, M. Li, J. Cai, N. Xue, J. Zhang, D. Liu, P. Craig, and X. Huang, "Improving Efficiency of Authenticated OpenFlow Handshake using Coprocessors," in *IEEE 8th International Conference on Information Technology in Medicine and Education (ITME)*, 2016, pp. 576–580.

[19] D. Johnson, A. Menezes, and S. Vanstone, "The Elliptic Curve Digital Signature Algorithm (ECDSA)," *Int. J. Inf. Secur.*, vol. 1, no. 1, pp. 36–63, 2001.

[20] X. Cao, W. Kou, L. Dang, and B. Zhao, "IMBAS: Identity-based multi-user broadcast authentication in wireless sensor networks," *Comput. Commun.*, vol. 31, no. 4, pp. 659–667, 2008.

[21] Y. Liu, J. Li, and M. Guizani, "PKC based broadcast authentication using signature amortization for WSNs," *IEEE Trans. Wirel. Commun.*, vol. 11, no. 6, pp. 2106–2115, 2012.

[22] A. C. Cameron and A. G. F. Windmeijer, "An R-squared measure of goodness of fit for some common nonlinear regression models," *J. Econom.*, vol. 77, no. I, pp. 329–342, 1997.

[23] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks," No. RFC 4944, 2007.