Journal of Information Security, 2015, 6, 69-81
Published Online April 2015 in SciRes. http://www.scirp.org/journal/jis
http://dx.doi.org/10.4236/jis.2015.62008

Scientific Research Publishing

An Enhanced Dragonfly Key Exchange Protocol against Offline Dictionary Attack

Eman Alharbi, Noha Alsulami, Omar Batarfi

Information Technology Department, King Abdul-Aziz University, Jeddah, KSA
Email: alharbi_eman@yahoo.com, naalsulami2@kau.edu.sa, obatarfi@kau.edu.sa

Received 9 February 2015; accepted 7 March 2015; published 12 March 2015

Abstract

Dragonfly is Password Authenticated Key Exchange protocol that uses a shared session key to authenticate parties based on pre-shared secret password. It was claimed that this protocol was secure against off-line dictionary attack, but a new research has proved its vulnerability to off-line dictionary attack and proving step was applied by using "Patched Protocol" which was based on public key validation. Unfortunately, this step caused a raise in the computation cost, which made this protocol less appealing than its competitors. We proposed an alternate enhancement to keep this protocol secure without any extra computation cost that was known as "Enhanced Dragonfly". This solution based on two-pre-shared secret passwords instead of one and the rounds between parties had compressed into two rounds instead of four. We prove that the enhanced-Dragonfly protocol is secure against off-line dictionary attacks by analyzing its security properties using the Scyther tool. A simulation was developed to measure the execution time of the enhanced protocol, which was found to be much less than the execution time of patched Dragonfly. The off-line dictionary attack time is consumed for few days if the dictionary size is 10,000. According to this, the use of the enhanced Dragonfly is more efficient than the patched Dragonfly.

Keywords

Password Authenticated Key Exchange (PAKE), Original Dragonfly, Patched Dragonfly, Enhanced Dragonfly, Two-Pre-Shared Password

1. Introduction

Nowadays, information is increasing at a faster rate and it needs to be secured when exchanging it over insecure networks. The most efficient way to secure this information is Cryptology which is the science of building and analyzing secret code. It consists of cryptography and cryptanalysis. Cryptography is the science that builds se-

How to cite this paper: Alharbi, E., Alsulami, N. and Batarfi, O. (2015) An Enhanced Dragonfly Key Exchange Protocol against Offline Dictionary Attack. *Journal of Information Security*, **6**, 69-81. http://dx.doi.org/10.4236/jis.2015.62008

cret codes and cryptanalysis is the science that analyzes those codes [1]. There are two types of cryptography: Symmetric and Asymmetric key cryptography. Symmetric key cryptography has a shared key that is used by the sender and receiver in encryption and decryption processes, while Asymmetric key cryptography has a public key which is used in encryption processes and private key which is used in decryption processes [2].

One of the symmetric cryptography protocols is key exchange protocol, which allows two parties who share non-secret information to compute a shared key via public communication. Authenticated Key Exchange (AKE) is a symmetric protocol that not only allows parties to compute the shared key but also ensures the identity of the parties where a party can compute a shared key only if he/she is the one who claims to be [3]. Another example of symmetric protocols is the Two-party Password-based Authenticated Key Exchange (2PAKE) protocol that permits two parties to generate a session key based on a pre-shared password and authenticates each other [4]. In general, PAKE protocols are exposed to different types of attacks, mainly passive and active attacks. Passive attacks are difficult to detect and are used to get information that is sent between parties without modifying the information or harming the resources. Active attacks enable the attacker to modify the encrypted message or change the meaning of the decrypted message by creating false streams [5]. However, the most complicated type of attack is the dictionary attack regardless of whether it is offline or online. Offline dictionary attack enables the attacker to record information from successful execution of the protocol by eavesdropping on the communication between parties. Then the attacker goes offline to find the required password by trying all passwords and choosing the correct password depending on the recorded information. The interaction with the server is not required in an offline dictionary attack. Online dictionary attacks enable the attacker to find the required password by trying all passwords during the interaction with the server [6].

2PAKE protocol has various types of protocols. In 1992, the first 2PAKE protocol (called the Encrypted Key Exchange (EKE)) was proposed by Bellovin and Merritt [7]. Since then, many other 2PAKE protocols have been proposed (e.g. [8]-[12]). One of these protocols was Dragonfly which was defined by Dan Harkins who claimed that this protocol had resistance to passive attacks, active attacks and offline dictionary attacks without security proofs [12]. In 2013, Dylan and Feng analyzed this protocol and proved that this protocol had weakness points in its security properties against active attack and offline dictionary attacks. They proposed to patch Dragonfly as a solution to verify its security properties by adding a public key validation [12].

The main problem in the patched dragonfly protocol was that it consumed an increased computation cost during the execution of the protocol when public key was used, which involved a large bit operation. In this paper, we introduced another enhancement for the Dragonfly protocol without using a public key. The proposed solution is based on using two-pre-shared secret passwords instead of one and the number of rounds between parties has been reduced to two instead of four. This step adds more security to the original Dragonfly protocol while the attacker will require more than 1.5 days to perform the off-line attack in case the shared password is extracted from a dictionary with the size 1000. This result has been approved through a simulation project which was developed by the java programming language. The simulation project has been used to find out the required execution time by both parties who are sharing the communication. The execution time is found to be less than the time required in the use of public key validation. We also approved the efficiency of Enhanced-Dragonfly protocol by using the Scyther tool to analyze its security properties and its structure.

The paper is organized into sections: Section 2 presents the related work; Section 3 presents review of the Dragonfly protocol; the proposed solution is presented in Section 4, while discussion is presented in Section 5. Finally, the conclusion is in Section 6. It is focused towards those who are interested in cryptography and cryptanalysis domain.

2. Related Work

Password Authenticated Key Exchange (PAKE) protocols have been playing an essential role in providing secure communications. As is usual in cryptography, with each new release of a protocol, attempts to attack this protocol have also been released in order to measure resistance against these attacks.

In research [13], analysis of the security weaknesses of SPAKE1 and SPAKE2 protocols has been introduced. It was found that these protocols are vulnerable against password compromises, impersonation and Denial-of-Service (DoS) attacks. Additionally, it was proved that the Hitchcock protocol [14], which uses public key encryptions, is insecure against momentary key compromise masquerades, Key Compromise Impersonation (KCI) attacks and off-line dictionaries. To remove these disadvantages, a new efficient protocol has been designed which relies on two-party PAKE protocols based on symmetric key exchange protocol and multiple hash

function calculations. The public key scheme has been replaced here with the symmetric encryption because the latter requires fewer bit operations. Many theories have been explained to prove that the proposed scheme is secure against various attacks. It provides the mutual authentication and forward secrecy attributes with many other security attributes. However, there is a little increase in computational costs that was caused by the extra hash calculations. This may be deemed negligible when considering the efficiency of the extra security attributes and lower number of rounds.

Moreover, another research [15] introduced the vulnerabilities of SPEKE and EKE protocols and analyzed them due to theoretical practices. In addition, a new PAKE protocol was proposed and was called Password Authenticated Key Exchange by Juggling (J-PAKE). It depends on shared secret-password between two entities and it has two rounds to create a strong shared key. It is based on well-established primitive in cryptography called Zero-Knowledge Proof (ZKP) that allows a person to prove his knowledge of a discrete logarithm without revealing it. The analysis of this protocol proved that it prevents off-line dictionary attacks, limits an active attacker to guess only one password per protocol execution and provides forward secrecy. It has advantages like: protection of users from leaking passwords and it does not require PKI deployments. This protocol is very efficient according to the use of well-established primitives.

Recent researches in the field of PAKE protocols are moving towards using Public Key Infrastructure (PKI) in the authentication process. In recent years, PAKE has been used with RSA encryption method to provide the appropriate secrecy between two parties. In [16] a new proposed protocol named as E3PAKE-RSA, which is based on RSA scheme with the three-party model that enables two entities to only need to remember a human-memorable password to authenticate each other and agree on a large session key with the assistance of the trusted server. The public/private key parameters are selected by the clients rather than a trusted distribution center, which makes the process of on-line authentication obsolete. It was predicted that the computation cost for this protocol would be respectively high according to the two-party RSA-PAKE. However, during the testing phase, which applied to the generic constructions, the computation cost of E3PAKE-RSA protocol was found to be as much as the two-party RSA-PAKE and the number of total rounds are less than this. Actually, this protocol requires more time to execute due to the use of the RSA encryption method and the long bit operations, while the other PAKE protocols (which do not use PKI) work faster.

3. Overview on Dragonfly Protocol

Dragonfly is a PAKE protocol that is used to exchange session keys between two parties with mutual authentication inside mesh networks. It was defined by Dan [12]. Finite cyclic groups are required to implement dragonfly such as Finite Field Cryptography (FFC) or Elliptic Curve Cryptography (ECC) [17]. This paper implements this protocol by using FFC. A finite field is “a field F that contains a finite number of elements which is the order of F” [18]. More details about finite field are presented in [19] [20].

In this section, we will explain the Dragonfly protocol in detail, with the discovered attack on it followed by the solution which is suggested in [12].

3.1. Original Dragonfly Protocol

3.1.1. Notations

The meanings of different notations that occur in this paper are listed in **Table 1**.

3.1.2. Message Exchange

There are two types of message exchanges in Dragonfly which are [12] [17] [21]:

1) Commit Exchange is the message that is sent by both parties who commit to a single guess of the shared password.

2) Confirm Exchange occurs when both the parties confirm knowledge of the shared password.

3.1.3. Operations

There are three operations in Dragonfly protocol which are [12] [17] [21]:

1) Operation: is the modular multiplication that takes two elements as inputs and the output will be a new element, e.g. C = A.B mod p (A, B and C are elements and p is a prime number).

2) Scalar operation takes one element and one scalar as input and the output will be a new element, e.g. C =

Table 1. Notations of original/enhanced dragonfly protocol.

Notation	Definition	Notation	Definition
IDA	Initiator's identifier	EA	Initiator's element operation
IDB	Responder's identifier	EB	Responder's element operation
Q	Finite cyclic group	E'A	Initiator's element operation after hashing
K	Shared session key	E'B	Responder's element operation after hashing
ss	Shared secret code.	A	Initiator's scalar and element operations
N	New password element	B	Responder's scalar and element operations
P	Password element	rA	Random number generated by initiator.
q	the order of Q	rB	Random number generated by responder.
p	The group prime	H()	Hash function
sA	Initiator's scalar operation	mA	2nd random number generated by initiator.
sB	Responder's scalar operation	mB	2nd random number generated by responder.

Ab mod p (for scalar b, elements A, C and p a prime number).

3) Inverse operation: There is an inverse of elements if the result of the multiplication of that element with its inverse is 1, e.g. (A*inverse (A)) mod p = 1 (for elements A and p which is a prime number).

3.1.4. Protocol Execution

Suppose Alice and Bob are parties who want to execute Dragonfly protocol, they should follow some sequential steps in order to implement this protocol as shown in **Figure 1** [12] [17] [21]:

1) Before sending any message, Alice and Bob share a password that is a string which includes their identities and they also choose FFC or ECC to use. Then the password is mapped into an element of Q which is a generator (P) by using an algorithm.

2) Both parties produce scalar and element operations based on choosing two scalars that belong to Q randomly (r, m). Then each party sends his/her scalar and element operations to another party (s, E).

3) After that each party calculates the shared secret depending on element and scalar operations for them ss.

4) Both parties send a hash function that contains scalar and element operations of both parties and ss.

5) Each party verifies the hash; if it is valid then the authentication succeeds. Then they create a shared key K = H(ss|EA·EB|(sA + sB) mod q), but if it is invalid then the authentication fails and all messages must be destroyed.

3.2. Attack on Original Dragonfly Protocol

An attacker was able to impersonate the identity of Bob by following specific algorithm which has been defined in [12] which makes him accessible to the pre-shared password that is P. There are two cases when the attacker can do such attack:

1) If the algorithm takes a long time to run and Alice terminates the communication, then the attacker can get P and he can impersonate the identity of Alice or Bob in the next run of the Dragonfly protocol.

2) If the algorithm is completed quickly, then the attacker impersonates the identity of Bob during the run of protocol by sending his hash function and sends it to Alice to compute the shared key K.

3.3. Patched Dragonfly Protocol

This solution was used to secure the Dragonfly protocol against the off-line attack. This was developed by [12], which relies on using public key validation. However, the execution time, which was examined through the introduced simulated project, was relatively high. Generally, the use of public key validation causes an increase in the computation cost to the execution of the protocol because a large number of bit operations will be required.

This research is conducted to find a better solution for this protocol without consuming more time in order to keep it efficient and compete with other PAKE protocols. This will be discussed in the following sections.

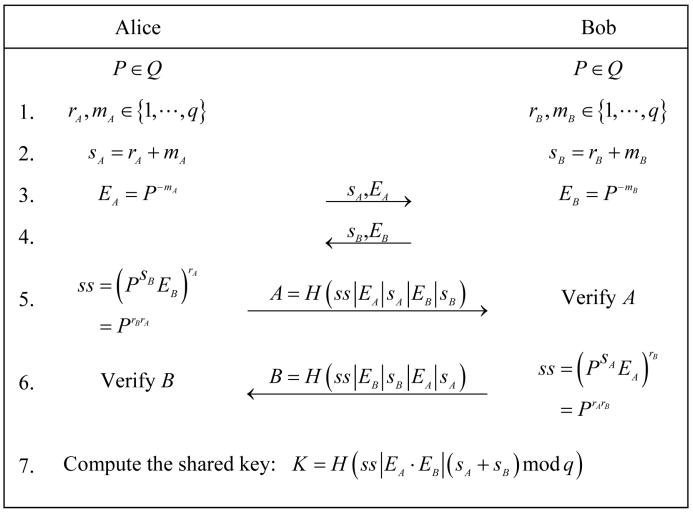

Figure 1. Steps of dragonfly protocol.

4. The Proposed Solution

This paper proposes a solution to secure the original Dragonfly protocol without the addition of public key in order to keep the execution time as fast as possible. It relies on using two pre-shared passwords instead of one. Here the attacker must search for each N, P in the dictionary. The first password P will be generated in the same way as in the original Dragonfly protocol. This is done by extracting a string from the dictionary and using the identities of both parties. It has a length of 1024 bits. The generation of the second password N is the same as P but with two minor changes; which are based on the extraction of another word from the same dictionary and the use of the identity of the respondent and it has a length of 160 bits instead of 1024.

The enhanced Dragonfly protocol will work according to the following steps:

1) Two passwords will be generated, which are P and N, and will be shared between both parties.

2) Both parties produce scalar and element operations based on choosing one scalar that belongs to Q randomly (r). Then each party will compute his/her scalar $(A,B) = r(A,B) + N$ and element operations $E = P - N$.

3) Both parties will compute the hash value of E which is $E' = H(E)$.

4) The first party will compute the hash value A for the parameters E' and s_A. Then A will be sent to the second party with the scalar value of the first party. The second party will verify the hash value A to ensure the authentication process. Here if the hash value is verified then the second party will compute the hash value B and send it to the first party with his/her scalar value S_B, otherwise it will be declined. The first party in turn will verify the hash value B. If it is valid then the authentication succeeds.

5) After that each party will calculate the shared secret which is $ss = (P^{sB} E)^{rA}$

6) Then they create a shared secret key which is $K = H(ss|E|(s_A + s_B) \bmod q)$.

The benefits of this solution are: firstly, the attacker will never be able to guess the hash value which includes the hash of the element value that uses the exponent of two-secret values. Secondly, the authentication process occurs in two rounds rather than four rounds as in the original protocol. This elimination makes the execution process go faster. The steps of enhanced Dragonfly protocol have been displayed in **Figure 2**.

4.1. Methodology over Protocol Testing

A simulation project has been developed to test the time needed to execute the protocol compared to the patched dragonfly protocol. It was written in Java and maintained in Net Beans. This project has been handled by using 64 bits Windows 8.1 pro-operating system and a processor with the speed 2.38 GHz. This project includes three various classes:

1) Dragonfly: is the main class that has objects from the other two classes and it also has some local variables such as:

- ID_A, ID_B are identities of initiator and respondent. The identity is the Mac address. For example ID_A = 001CB3098515, ID_B = 001AB3008600.

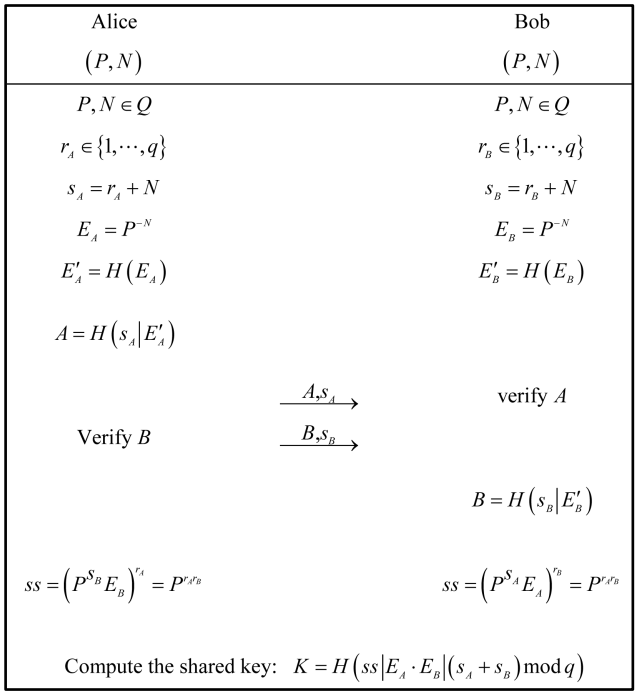

Figure 2. The enhanced dragonfly protocol.

- p_modular with the length of 1024 bits.
- Q is the order of subgroup whose length is 160 bits.
- secret_password: it is the first shared word from the dictionary (English word dictionary) that is used to generate P, while secret_password_2 is the second shared string word from the dictionary that is used to generate N. for example Password 1 = aardvark, Password 2 = aback, and the result of the generation process is located in **Table 2**.

2) Password: is the class to convert two shared string words that are taken from the dictionary into elements in Q subgroup and using two Mac addresses as identities for Alice and Bob to generate P and N. It has three methods which are:

- Password_Generator: the constructor that takes p_modular and q as parameters.
- pass1_ToGenerator: which takes the first word and generates the first shared password with the identities of Alice and Bob, which is P. The generated password will be 1024 bits long.
- pass2_ToElement: which takes the second word and generates the second shared password with the identity of Bob only, which is N. The generated password will be 160 bits long.

3) EnhancedDragonfly: is the class that implements the proposed solution. It works as the initiator agent, which is Alice. The respondent agent which is Bob will follow the same steps that are provided by this class. First, a random value rA will be selected from {1,…,q} while q is a Subgroup Order as shown in **Table 2**. The passed P and N will be used with rA to calculate Scalar sA, Element E and hash value A. Then these values will be used to calculate the shared secret key. We use MD5 for hash function.

This class includes seven methods:

1) Enhanced Dragonfly is the constructor that takes p_modularandq as parameters.

2) scalar_operation: it is calculating scalar operation that is sA = rA + N

3) element_operation: it is calculating element operation that is E = P N

4) sending_operation: it is used to calculate and send the hash value A

5) verifying_operation: It returns true or false according to the verification of the hash value of the other party.

6) sharedsecret_computation: used to calculate the shared secret key ss, which will be used later in the next equation.

7) sharedkey_computation: returns the secret key in the form of hash value.

Table 2. Values of important parameters of enhanced dragonfly protocol simulation.

Parameter	Value
p (prime modulus) Length = 1024 bit	158504237350117218257694040407858462891200963393591516141634530019500073576668911484320236691230272700829696627835300528268812589945710892915312387871915477639981299699199508802054202638712563938694798320890359871758714295993316429790666166949637203291430199423252838320875858890351875476170726372911749163531
P (shared password 1) Generator of subgroup Length = 1024 bit	245836050749495025790716955721356594922029074157543413879265529608146406103090427746649685991947950279676336777509950329927150039213331945321969566477919750646586599496131181689792341953230133744753934853335811527838271893831704049295570586189433061342003202390396402570049947445088999117217876457221586107 4
q (subgroup order) Length = 160 bit	1067173368745944614404811917041803739479862559381
N (shared password 2) Length = 160 bit	1000000000067898284897661991820474501459056843147
r_A Length = 160 bit	731591560221122268400653835555982580348221317561
Shared key in hash value with length 128 bits	84377010012050211717876910330791257 1

The class diagram of this project is displayed in **Figure 3**, while **Figure 4** presents the input and output of Enhanced Dragonfly protocol simulation. The values of important parameters of Enhanced Dragonfly protocol simulation are listed in **Table 2**.

By applying this simulation, the execution time for one participant was found to be 19 minutes only. This experiment was applied 15 times in order to measure the total execution time of the enhanced Dragonfly protocol and the average of the total time is almost 38 minutes. This result works as a proof for the efficiency of the proposed solution, and the use of two-shared passwords will not affect the time of exchanging the secret keys.

4.2. Testing and Analysis of the Enhanced Dragonfly Protocol

Any protocol should be analyzed in order to see if there are any possible attacks that can occur and find how secure it is. The automatic analysis is the most recent approach which is used in analysis of security protocols and it is more efficient than the manual one. The required time to write the protocol description depends on the learning of its language. All possible attacks will be tested by the used tool. The automatic analysis will be used in this research.

Various tools are used to analyze the security protocols and to check the possibility of any attack can happen when protocol is put to use. An example for this is the Scyther tool which supports the evaluation of protocols with respect to these corruption models which have led to the automatic detection of many attacks that previously could only be found by manual analysis [22].

4.2.1. Description of Scyther Tool and SPDL

Scyther [23] is one of the most efficient analyzing tools compared with other tools [24]. It was found that this tool can discover attacks more efficiently than others and it can also verify protocols with an unlimited number of sessions. Its run time is also less than others. There are many protocols that have been analyzed by this tool such as IKEv1, IKEv2 protocol suits, ISO/IEC 9798 family of authentication protocols and MQV family of protocols.

The Scyther tool takes the protocol description as input, and outputs a summary report stating whether the claim was true or false, whether any attack was found or not, and optionally, visual graph descriptions in the dot language from the GraphVizlibrary or representations of trace patterns in XML which describe the trace pattern [25]. Additionally, there are a set of Python bindings for the Scyther command-line tool, which provide a convenient means to script Scyther experiments. **Figure 5** displays how this tool works.

Security Protocol Description Language (SPDL) is a language that is used in Scyther for writing a protocol [26]. This language is just expressive enough to capture the most relevant security protocols in an abstract way. It allows us to develop high-level reasoning methods and tools, which can be applied to a large class of protocols. SPDL is used to create a definition for existing protocols that use symmetric or asymmetric encryption. It includes three scopes; global scope (the largest scope), protocol scope (smaller than global) and role scope (the

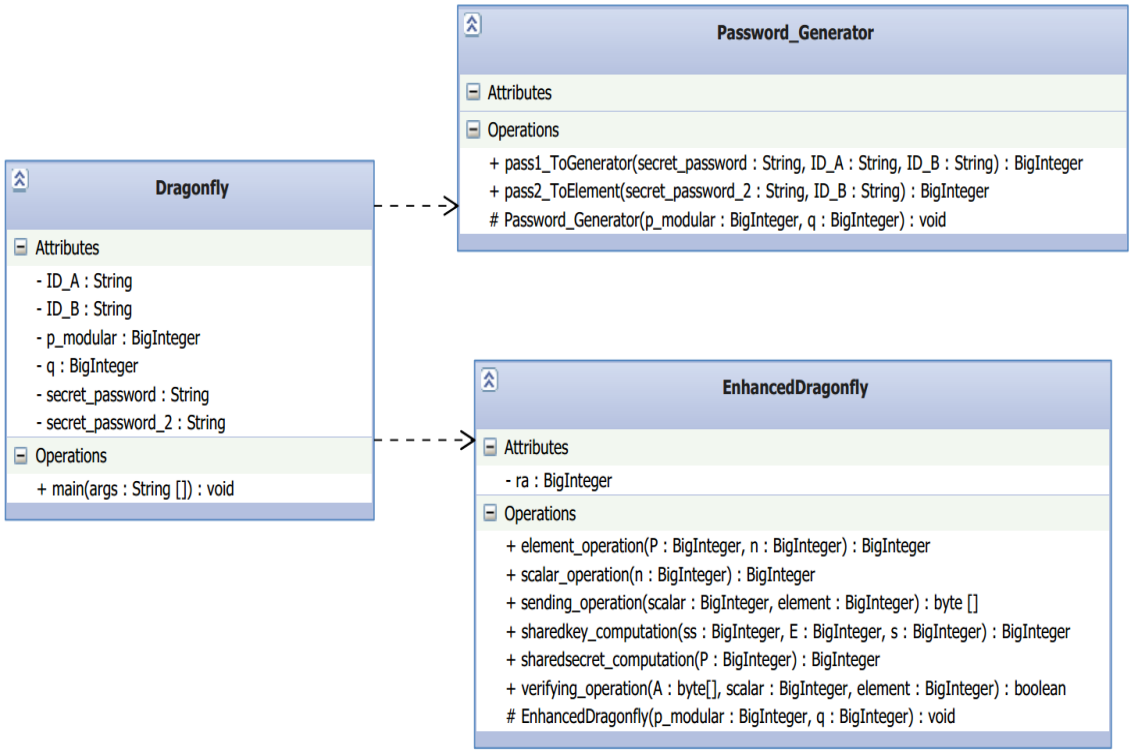

Figure 3. Enhanced dragonfly class diagram.

```
run:
p=
15850423735011721825769404040785846289120096339359151614163453001950007357666891148432023669123027270082969662783530052826881258994571089291531238787191547763998129969919950880205420263871256393869479832089035987175871429599331642979066616694963720329143019942325283832087585889035187547617072637291174916
3531
q= 106717336874594461440481191704180373947986255938 1
Enter a secret password:
aardvark
Enter The Identity of the Initiator:
001CB3098515
Enter The Identity of the Respondent:
001AB3008600
P=
24583605074949502579071695572135659492202907415754341387926552960814640610309042774664968599194795027967633677750995032992715003921333194532196956647791975064658659949613118168979234195323013374475393485333581152783827189383170404929557058618943306134200320239039640257004994744508899911721787645722158610
74
Enter a second secret password:
aback
Enter The Identity of the Respondent:
001AB3008600
n= 10000000000678982848976619918204745014590568431 47
ra = 731591560221122268400653835555982580348221317561
hash value of shared key
84 - 37 70 100 120 50 2 117 - 17 87 69 - 103 - 30 79 125 - 71
Execution time of protocol in millisecond= 11
```

Figure 4. Input and output of the enhanced dragonfly simulation project.

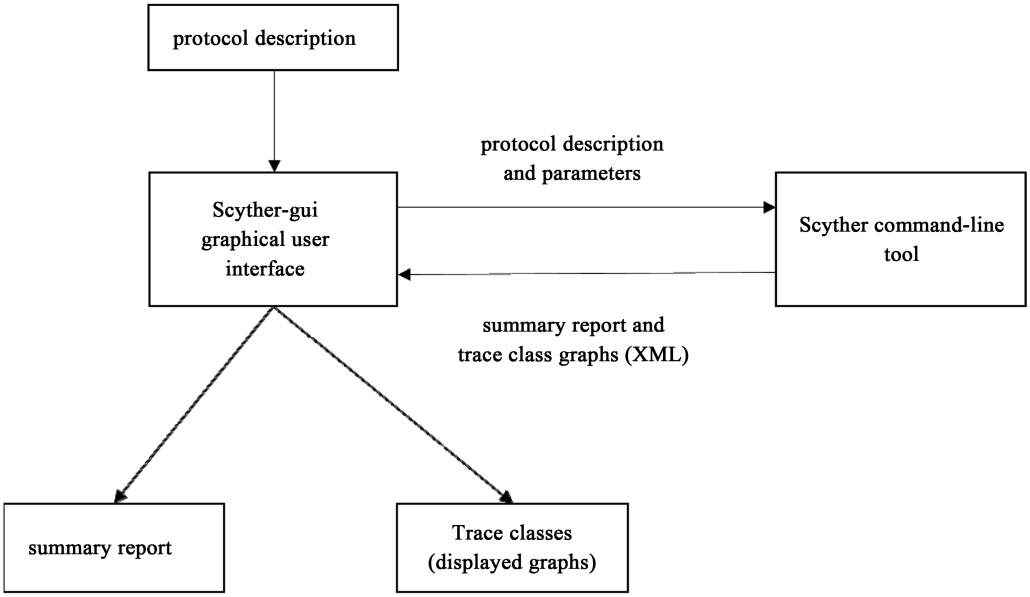

Figure 5. The syther tool.

smallest one). Furthermore, it includes the initial knowledge required to execute a role (each agent's role includes sequence of events, such as sending or receiving a message), the declaration of functions, global constants and variables.

4.2.2. SPDL of Enhanced Dragonfly Protocol and Analysis Result

In this sub-section, the SPDL representation of Enhanced Dragonfly protocol will be explained in detail. After that, the findings from this analysis will be discussed. The SPDL code of the enhanced dragonfly is displayed in **Figure 6**. First of all, a global hash function H1 & H2 has been defined (outside the protocol definition) to make it available to be accessed by all rolls in the protocol. Also, the functions of multiplication mult, addition add, exponential exp and the inverse inv were defined globally. Function is a special type which defines a function term that can take a list of variables. By default, it behaves like a hash function: given the term H1(x) where H1 is of type Function, it is impossible to derive x. These functions are defined in an abstract way in SPDL, which doesn't support any real equations.

Next, the protocol description starts with the keyword protocol which takes the role names as parameters. There are three rolls, namely; the key generator center (role KGC), the initiator (role A) and the respondent (role B):

- **Key generator center role KGC** [27]: Used here to pass the shared secret password P and N to role A and B. The way of generation, which is used to generate these shared passwords, doesn't take a place here in the tool because the tool depends on the structure of the protocol only. The shared secret passwords P&N have been identified as constants and their scope is within the protocol because they should be used inside all roles.
- **Initiator role A:**

❖ This role represents the start of the communication to compute the shared secret key. The selection of random value r_A, which is the first step in the protocol structure (see **Figure 2**) by Alice (initiator), has been clarified by using the fresh declaration which is used by SPDL to get random values. Nonce is a standard data type that is often used in Scyther tool.

❖ The calculation of S_A (Scalar value), E (element value), and A1 (hash value) occurs by using the predefined functions, and then S_A and A1 will be sent to role B.

❖ Note: labels in recv and send need to disambiguate similar occurrences of events in a protocol specification. In addition, they are used to link each send event with its recv one [28].

❖ In the received event of role A, the hash value of B and the Scalar value S_B will be saved in a Ticket variable, which can be substituted by any term.

- ❖ Role A will verify if these values are from Role B or not by using the match function. The definition of match [29] ensures the new variable assignment equivalent to the old one and it returns true or false. If the returned value is true then the next steps computed, otherwise Role A will conclude that this value is not from the honest agent, and it closes the session.
- ❖ The calculation of the secret key needs to be kept in secrecy, so that a claim with the type secret is used. The main idea behind claim events is locality [28] [29]: that a term is not in the adversary knowledge, or any certain role in run. Secrecy means that if an agent communicates with non-compromised agents, the term should be secret in every trace of the protocol. Secrecy expresses that certain information is not revealed to an adversary, even though this data is communicated over an un-trusted network. Here, claim event has been used to keep session's key secret and is known to role A only. When the tool runs, it tries to find the corresponding attack to each claim.

- **Respondent Role B:**

- ❖ Generates the random value r_B by fresh declaration.
- ❖ Calculates the scalar S_B, Element E and hash value B1 and sends it to Role A.
- ❖ Verifies the received hash value from Role A and compares it by using match function.

If the output of matching is true, then B will calculate the secret session key and claim it with secret type.

The SPDL of the protocol is inserted into the Scyther tool to see if any attack could occur or if there is any structural weak point in this proposed protocol. The result of this analysis brings out that there is NO attack found. All roles are verified, which means that no impersonation can occur during the connection and the characteristics of the protocol structure are verified. This means that the way of exchanging the data is totally secure. This result is printed in **Figure 7** and **Figure 8**.

```
// Hash functions
hashfunction H1,H2;
// Addition, multiplication, exponent, inverse
function mult,add,exp,inv;
// The protocol description
protocol dragonfly(A,B,KGC)
{
const N,P;
role KGC // Key Generation Center
{
send_1(KGC,A,P,N); // Publish public parameters
send_2(KGC,B,P,N);
}
role A
{
fresh ra:Nonce;
var B1,SB:Ticket;
recv_1(KGC,A,P,N);
send_3(A,B,add(ra,N), H1(exp(P,inv(N)),add(ra,N))); // Send SA and A1

recv_4(B,A,SB,B1);
match(H1(add(ra,N),H1(exp(P,inv(N)))),B1);
claim_A1(A,Secret,(H2(exp(exp(P,SB), mult (exp(P,inv(N))),ra),H1(exp(P,inv(N))), add(ra,N))));
};
role B
{
fresh rb:Nonce;
var A1,SA:Ticket;
recv_2(KGC,B,P,N);
recv_3(A,B,SA,A1);
match(A1,H1(SA,H1(exp(P,inv(N)))));
send_4(B,A,H1(add(rb,N),H1(exp(P,inv(N)))),add(rb,N));// Send SB and B1
claim_B1(B,Secret,(H2(exp(exp(P,SA), mult(exp(P,inv(N))),rb),H1(exp(P,inv(N))), add(SA,add(rb,N)))));
};
}
```

Figure 6. SPDL of enhanced dragonfly protocol.

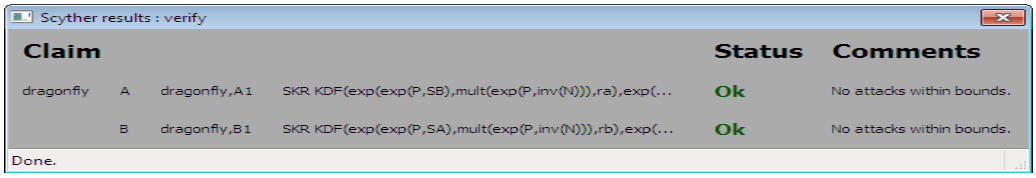

Figure 7. Scyther result of the enhanced dragonfly.

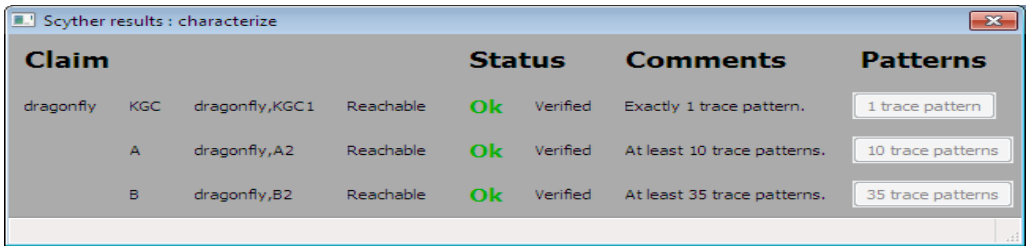

Figure 8. Verification of the enhanced dragonfly structure.

5. Discussion

This study has revealed that the enhanced Dragonfly protocol is secure against off-line dictionary attacks and it's approved for efficiency due to its execution speed. In this section, the off-line attack will be developed within the simulation project in order to analyze the time which is required for the attacker to perform his/her search against the two-shared passwords. Then, a comparison between the efficiency of the enhanced Dragonfly protocol and the original/patched Dragonfly protocol is introduced.

5.1. Off-Line Attack Simulation

Attack class has been applied to previous simulation projects. This class works by measuring the time taken to check all possible passwords, as dictionary size varies from 100, 1000 to 100,000 words. Each experiment was performed 5 times to find the search time by attack class and the average value was taken. As a result, the average time of this search within the enhanced protocol increased linearly as with increase in the dictionary size. The average time of off-line search when the dictionary length is 10,000 is more than 1.5 days, and with the original Dragonfly protocol [12] the off-line search algorithm took an average of 25 seconds to find the correct password with the same dictionary length. This is because the enhanced protocol has an exponential relationship between the two-secret passwords in the computation of the element variable. Therefore, the attacker will be forced to search for two words instead of one. As a comparison between the time that is required for searching one password and the time required for searching two passwords, this experiment found that the time increased exponentially. The search of such an attack on the patched Dragonfly protocol requires few days to be successful. However, here the execution time will have an impact on the efficiency of Key exchange technique. **Table 3** shows a linear relationship between dictionary size and the time taken to try all passwords.

5.2. Comparison between Enhanced Dragonfly and Patched Dragonfly Protocols

The enhanced Dragonfly protocol is very similar to the patched Dragonfly protocol except for two minor changes. Firstly, the patched protocol uses only one shared password and a public key, while the enhanced protocol uses two-shared passwords. Second, the rounds between the initiator and the respondent are only two rounds in the enhanced protocol while it is four rounds in the patched protocol. However, the patched dragonfly involves some computational cost because it is uses a public key validation which creates large bit operations due to its large size. This can decrease the protocol efficiency and make it less appealing than its competitors [12]. Otherwise, in the enhanced dragonfly protocol, the execution of the protocol will take less time. In order to measure the execution time of the enhanced Dragonfly protocol we used the simulation project to compute the time required by the key generation center that uses dictionary size 10,000 plus the time which required by the initiator party when execute the required computation of the protocol. The cost of both patched and enhanced Dragonfly protocol for each participant has been placed into **Table 4**, and based on these time measurements we

Table 3. Time average of off-line search on enhance dragonfly protocol.

Dictionary size	Average of the first attempt to find P	Average of the first attempt to find P & N	The time of the dictionary attack
100	20 ms	40 ms	5.1 minutes
1000	135 ms	269 ms	4.8 hours
10,000	141 ms	282 ms	More than 1.5 days

Table 4. Computation cost for dragonfly and enhanced protocol.

Protocol	Execution time/participant
Public key Dragonfly	64.0 ms
Two-shared password Dragonfly	19.0 ms

have observed that the efficiency of using two-shared passwords is more than using the public key.

The limitation of this research is that these results are based on a normal speed machine with the 64 bits Windows 8.1 pro-operating system and a processor with the speed of 2.38 GHz which is not the case in the real time attack. Furthermore, the attacker is likely to significantly shorten the time of exhaustive search by distributing the calculation over several high performance machines.

6. Conclusion

In this research we proved that the Dragonfly protocol can be enhanced in a way, other than the public key validation, to make it secure against off-line dictionary attacks. The proposed solution that has been introduced to the original Dragonfly protocol relies on the use of two-shared secret passwords P and N. This solution works by the use of an exponential relationship between these passwords and performing the authentication process in two rounds only instead of four, which positively affects its efficiency. The use of public key validation, which is proposed in the patched protocol, is causing a computational cost to its execution that makes it less efficient. However, the execution time of the enhanced Dragonfly protocol and the time of an off-line dictionary attack were measured in the developed simulation project. As a result of this simulation, the required time to execute the enhanced protocol for one of the parties was 19 ms, while it was 64 ms in the patched protocol. Respectively, the time to perform an off-line dictionary attack was very close to the time to attack the patched protocol for the dictionary size 10,000, which meant that the resistance of the enhanced protocol against off-line dictionary attack was proved. The enhanced protocol has been analyzed by using Scyther analysis tool and it was found to be secure and did not have any structural failure. We can say that, the enhanced dragonfly protocol can be executed in a very short time while the off-line dictionary attack is a time consuming search for the attacker and it needs a few days to complete it.

References

[1] Stamp, M. (2007) Classic Ciphers in Applied Cryptanalysis: Breaking Ciphers in the Real World. Wiley-IEEE Press, Canada, 25. http://dx.doi.org/10.1002/9780470148778.ch2

[2] Kendhe, A.K. and Agrawal, H. (2013) A Survey Report on Various Cryptanalysis Techniques. *International Journal of Soft Computing and Engineering* (*IJSCE*), **3**, 287-293.

[3] Lauter, K. and Mityagin, A. (2006) Security Analysis of KEA Authenticated Key Exchange Protocol. International Association for Cryptologic Research.

[4] Saeedetal, M. (2012) An Enhanced Password Authenticated Key Exchange Protocol without Server Public Keys. ICTC, 2012.

[5] Ali, R., Kumar, A. and Narayan, E. (2013) Encryptioan/Decryption Tool with Cryptanalysis. *International Journal of Computer Science & Engineering Technology* (*IJCSET*), **4**, 1043-1050.

[6] Goyal, V., *et al*. (2005) CompChall: Addressing Password Guessing Attacks. ITCC 1, IEEE Computer Society.

[7] Bellovin, S.M. and Merritt, M. (1992) Encrypted Key Exchange: Password-Based Protocols Secure against Dictionary Attacks. *Proceedings of IEEE Symposium on Security and Privacy*, Oakland, 4-6 May 1992, 72-84.

http://dx.doi.org/10.1109/RISP.1992.213269

[8] Kobara, K. and Imai, H. (2002) Pretty-Simple Password-Authenticated Key Exchange under Standard Assumptions. *IEICE Transactions*, E85-A(10), 2229-2237.

[9] Bellare, M., Pointcheval, D. and Rogaway, P. (2000) Authenticated Key Exchange Secure against Dictionary Attacks. Proceedings of the 2000 Advances in Cryptology (EUROCRYPT'2000). Springer-Verlag, Berlin, 139-155. http://dx.doi.org/10.1007/3-540-45539-6_11

[10] Bresson, E., Chevassut, O. and Pointcheval, D. (2004) New Security Results on Encrypted Key Exchange. In: *Proc. PKC* 2004, *Lecture Notes in Computer Science*, Vol. 2947, Springer-Verlag, Berlin, 145-158.

[11] Abdalla, M. and Pointcheval, D. (2005) Simple Password-Based Encrypted Key Exchange Protocols. *Proceedings of Topics in Cryptology—CT-RSA, Lecture Notes in Computer Science*, Vol. 3376, Springer-Verlag, Berlin, 191-208.

[12] Clarke, D. and Hao, F. (2013) Cryptanalysis of the Dragonfly Key Exchange Protocol.

[13] Saeed, M., Shahriar Shahhoseini, H. and Mackvandi, A. (2011) An Improved Two-Party Password Authenticated Key Exchange Protocol without Server's Public Key. *IEEE* 3*rd International Conference on Communication Software and Networks*, Xi'an, 27-29 May 2011, 90-95.

[14] Hitchcock, Y., Tin, Y.S.T., Boyd, C., Nieto, J.M.G. and Montague, P. (2003) A Password-Based Authenticator: Security Proof and Applications. *INDOCRYPT*'03, *Lecture Notes in Computer Science*, Vol. 2904, Springer-Verlag, Berlin, 388-401.

[15] Hao, F. and Ryan, P. (2010) J-PAKE: Authenticated Key Exchange without PKI. In: *Transactions on Computational Science XI*, Springer, Berlin, 192-206.

[16] Ma, C.G., Wei, F.S. and Gao, F.X. (2013) Efficient Client-to-Client Password Authenticated Key Exchange Based on RSA. *IEEE* 5*th International Conference on Intelligent Networking and Collaborative Systems*, Xi'an, 9-11 September 2013, 233-238.

[17] Harkins, D. (2008) Simultaneous Authentication of Equals: A Secure, Password-Based Key Exchange for Mesh Networks. 2*nd International Conference on Sensor Technologies and Applications* (*SENSORCOMM*), Cap Esterel, 25-31 August 2008, 839-844.

[18] http://cacr.uwaterloo.ca/hac/about/chap14.pdf

[19] Stalling, W. (2006) Chapter 4: Finite Fields. In: *Cryptography and Network Security*, 4th Edition, Pearson Education International, Upper Saddle River, 97-109.

[20] Saeed, M., Mackvandi, A., Naddafiun, M. and Karimnejad, H. (2012) An Enhanced Password Authenticated Key Exchange Protocol without Server Public Keys. 2012 *International Conference on ICT Convergence* (*ICTC*), Jeju Island, 15-17 October 2012, 87-91.

[21] Harkins, D. (2012) Dragonfly Key Exchange. Internet Research Taskforce Internet Draft Version 00. http://tools.ietf.org/html/draft-irtf-cfrg-dragonfly-00

[22] Basin, D., Cremers, C. and Meadows, C. (2009) LTL Model Checking Security Protocols. *Journal of Applied Non-Classical Logics*, **194**, 403-429.

[23] http://www.cs.ox.ac.uk/people/cas.cremers/scyther/index.html

[24] Cremers, C.J.F., Lafourcade, P. and Nadeau, P. (2009) Comparing State Spaces in Automatic Security Protocol Analysis.

[25] Franciscus, C.J. (2006) Scyther—Semantics and Verification of Security Protocols. Eindhoven University of Technology, Eindhoven.

[26] Dalal, N., Shah, J., Hisaria, K. and Jinwala, D. (2010) A Comparative Analysis of Tools for Verification of Security Protocols.

[27] Farouk, A., Fouad, M. and Abdelhafez, A. (2014) Analysis and Improvement of Pairing-Free Certificate-Less Two-Party Authenticated Key Agreement Protocol for Grid Computing. *International Journal of Security, Privacy and Trust Management, IJSPTM*, **3**, 23-36.

[28] Cremers, C. and Mauw, S. (2012) Chapter 4: Security Properties. In: *Operational Semantics and Verification of Security Protocols, Information Security and Cryptography*, Springer-Verlag, Berlin, 37-65.

[29] Cremers, C. (2014) Scyther User Manual. 18 February 2014.