Buletin Teknik Elektro dan Informatika
(Bulletin of Electrical Engineering and Informatics)
Vol.1, No.3, September 2012, pp. 233~244
ISSN: 2089-3191

Efficient Three Party Key Exchange Protocol

H. K. Pathak[1], Manju Sanghi[2]
[1]S.o.S in Mathematics, Pt. Ravishanker Shukla University, Raipur (C.G.) 492010, India
[2]Deptt. of Mathematics, Rungta College of Engg.& Tech., Bhilai (C.G.) 492006, India

Abstract

Key exchange protocols allow two or more parties communicating over a public network to establish a common secret key called a session key. In 1976, Diffie and Hellman proposed the first practical key exchange (DH key exchange) protocol. In 2005, Abdalla and Pointcheval suggested a new variation of the computational DH assumption called chosen based computational Diffie Hellman (CCDH) and presented simple password based authenticated key exchange protocols. Since then several three party password authenticated key agreement protocols have been proposed In 2007, Lu and Cao proposed a simple 3 party authenticated key exchange (S-3PAKE) protocol. Kim and Koi found that this protocol cannot resist undetectable online password guessing attack and gave fixed STPKE' protocol as a countermeasure using exclusive-or operation. Recently, Tallapally and Padmavathy found that STPKE' is still vulnerable to undetectable online password guessing attack and gave a modified STPKE' protocol. Unfortunately, we find that, although modified STPKE' protocol can resist undetectable online password guessing attack but it is vulnerable to man in the middle attack. Also, we propose and analyze an efficient protocol against all the known attacks.

Keywords: *Password based key exchange protocol, password guessing attacks, Man in the middle* attack*, AVISPA tool.*

1. Introduction

With the invention of world wide web and its rapid spread, the need for authentication and secure communication became still more acute. Key exchange protocols allow 2 or more parties communicating over a public network to establish a common secret key called a session key. In 1976, Diffie and Hellman [1] proposed the first practical key exchange protocol. However DH protocol does not provide authentication and hence, suffers from man in the middle attack. To over come this problem, many authenticated key exchange protocols, based on passwords, identities and public keys have been proposed. Since the password based authenticated key exchange protocols, require user to remember only human memorable passwords, without any device to generate or store them, they are quite simple and efficient.

Password based authenticated key exchange protocols, however are vulnerable to pass- word guessing attacks [3] since users usually choose easy-to-remember passwords. In general, the password guessing attacks can be divided into three classes:

Detectable on-line password guessing attacks: an attacker attempts to use a guessed password in an on-line transaction. He/ she verifies the correctness of his/her guess using the response from server. A failed guess can be detected and logged by the server.

Undetectable on-line password guessing attacks: similar to above, an attacker tries to verify a password guess in an online transaction. However, a failed guess cannot be detected and logged by server, as the server is not able to distinguish an honest request.

Off-line password guessing attacks: an attacker guesses a password and verifies his/her guess off-line. No participation of server is required, so the server does not notice the attack from a malicious one.

Due to practical significance many password authenticated key exchange (PAKE) pro- tocols have been proposed. In 1992, Bellovin and Merrit [2] proposed the first, two party password authenticated key exchange protocol (2PAKE) where two communicating parties Alice and Bob share low entropy passwords for establishing a secret session key. In 1994, Steiner [4] extended the concept to 3 party (STW3PEKE) with a trusted server S

Received July 4, 2012; Revised August 7, 2012; Accepted September 3, 2012

and clients A and B. In (1995) Ding and Horster [3] and Sun et al. [5] showed that Steiner et al's 3PAKE protocol is vulnerable to undetectable on line password guessing attacks. In (2000) Lin et al. [6] also showed that STW3PEKE suffers not only undetectable online password guessing attacks but also off-line password guessing attacks. Moreover, they pre- sented a new LSH-3PEKE protocol using server's public key. Lin et al. [7] (2001) presented LSSH-3PEKE resistant to both off-line and undetectable on line password guessing attacks without using server public keys. In (2004) Chang and Chang [8] gave LHL-3PEKE with round efficient version. In the same year Lee et al. [9] presented two enhanced three party encrypted key exchange protocols without using public key techniques. In 2005, Abdalla and Pointcheval [10] suggested a new variation of the computational DH assumption, called chosen-based computational Diffie-Hellman and presented SPAKE-1 and SPAKE-2, simple password based authenticated key exchange protocols.

In 2007, Lu and Cao [11] proposed a simple 3 party authenticated key exchange protocol (S-3PAKE) based on the concept of Abdalla and Pointcheval. They claimed that their protocol is superior to similar protocols with respect to security and efficiency. But unfor- tunately, works of Nam et al., [12] Chung and Ku [13] Chen and Jin [14] and Guo Hua et al. [15] showed that S-3PAKE suffers from various attacks like off-line dictionary attack, man in the middle attack, impersonation of initiator attack and impersonation of responder attack. In 2009, Kim and Choi [16] showed that S-3PAKE cannot resist undetectable online password guessing attack also and presented fixed STPKE' as a countermeasure by using exclusive or operation. Recently, Tallapally and Padmavathy [17] found that fixed STPKE' is still vulnerable to undetectable on-line password guessing attack, if the identity of the client is exposed. They proposed an alternative protocol by modifying STPKE'. Unfortunately, we find that modified STPKE' protocol although is resistant to undetectable online password guessing attack but cannot withstand man in the middle attack. We also propose an efficient key exchange protocol and show that it can withstand all the known attacks. We also validate the proposed protocol using AVISPA tool (Automated validation of internet security protocols and applications) which is a push button tool for the automated validation of security protocols.

The rest of the paper is organized as follows. In Section 2 we review S-3PAKE, fixed STPKE' and modified STPKE' protocols. In Section 3 we discuss man in the middle attack on modified STPKE' protocol. Section 4 gives the proposed protocol. The security analysis of the protocol along with its formal verification and validation is discussed in Sections 5 & 6. Finally the paper is concluded in Section 7

2. Review of password based protocols

2.1. Review of Lu and Cao's (S-3PAKE) protocol.

In 2007, Lu and Cao proposed a simple three-party key exchange (S-3PAKE) protocol based on the chosen-basis computational Diffie-Hellman (CCDH) assumption. They claimed that their protocol can resist various attacks and is superior to similar protocols with respect to efficiency.

The notations used in the protocol are described as follows:
(G, g, p): A finite cyclic group G generated by an element g with prime order p.
M, N: Two elements in G.
S: A trusted server.
A, B: Two clients.
P w_A : The password shared between A and S.
P w_B : The password shared between B and S.
H, H′ : Two secure one way hash functions.
x, y, z: Random numbers selected by A, B and S.

The steps of the protocol can be briefly described as shown in figure 1.

Step 1: A chooses a random number x ∈ Zp and computes U = gx.M PwA and sends A|| U to B.

Step 2: B also chooses a random number y ∈ Zp and computes V = gy .NPwB and sends

A || U|| B || V to S.

Step 3: Upon receiving A|| U || B || V , S uses PwA and PwB to compute gx = U/M PwA and gy = V/N PwB . Then S chooses another random number z ∈ Zp and computes gxz = (gx)z and gyz = (gy)z . Finally S computes U′ = gyz .H (A, S, gx)PwA and V′ = gxz .H (B, S, gy)PwB

Step 4: B, on receiving U′ || V′, uses PwB to compute gxz = V ′ / H (B, S, gy)PwB and uses the random number y to compute gxyz = (gxz)y and α= H (B, A, gxyz). B forwards U′|| α to A.

Step 5: A computes gyz = U′ /H (A, S, gx)Pw_A and uses x to compute gxyz =(gyz) x and verifies α. If the verification fails, A terminates the protocol, otherwise A computes the session key SKA = H′ (A, B, gxyz) and sends β = H ′ (A, B, gxyz) to B.

Step 6: B verifies β. If it holds, B computes the session key SKB = H′ (A, B, gxyz).

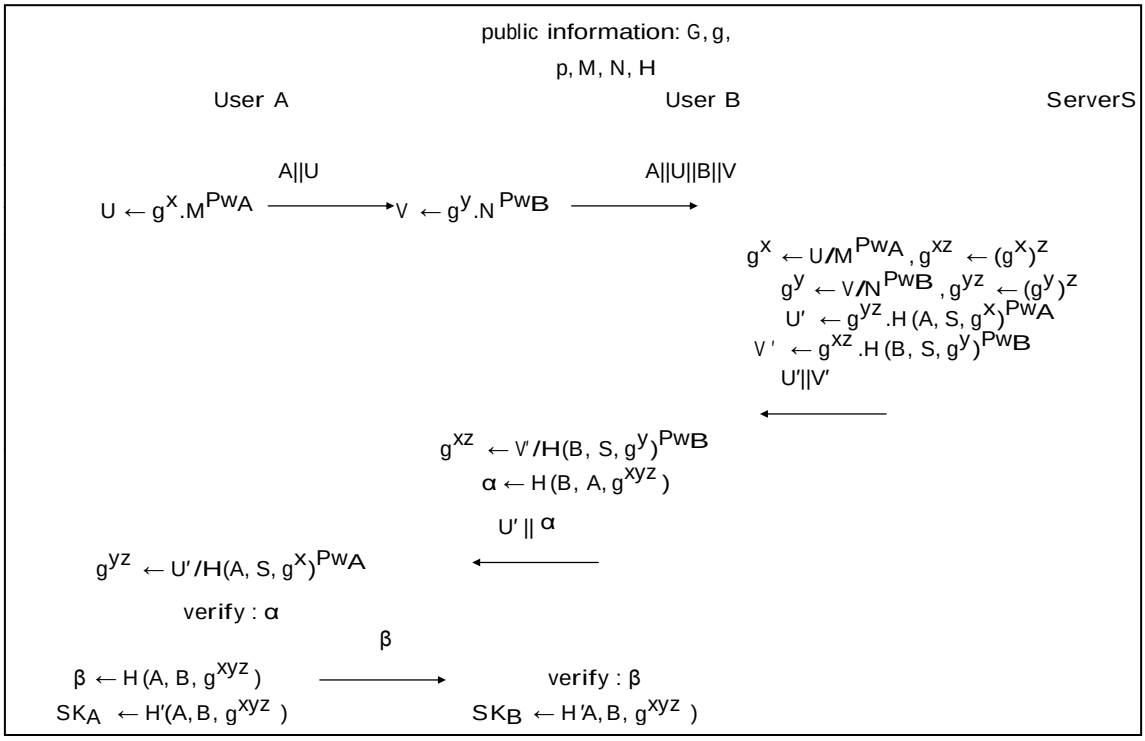

Figure1. Lu and Cao's S-3PAKE protocol

2.2 Review of Kim and Choi's STPKE' protocol.

Kim an Choi found that, S-3PAKE protocol is vulnerable to undetectable online password guessing attack by using formal description and proposed an alternative, fixed simple three party key exchange (STPKE') protocol by using an extra random number and exclusive OR operation. The following are the steps of the protocol:

Step 1a: A chooses 2 random numbers x, a ∈ Zp and computes U= (gx ⊕ ga).M PwA and
A′ = A.ga and sends A′ || U to
B.

Step 1b: B chooses random numbers y, b ∈ Zp and computes V= (gy ⊕ gb).N PwB , and B′ =
B.gb and sends A′ || U || B′ V to S.

Step 2a: Upon receiving A′ || U || B′ V , S finds ga = A′ /A, gb = B′ /B, gx ⊕ ga = U/M PwA , gx= (gx ⊕ ga) ⊕ ga , gy ⊕ gb = V /N PwB , gy = (gy ⊕ gb) ⊕ gb , chooses a random

number $z \in Zp$ and computes $gxz = (gx)z$ and $gyz = (gy)z$.

Step 2b: Finally, S computes $U' = gyz .H (A', B', S, gx)PwA$ and $V' = gxz .H (B', A', S, gy)$ PwB and sends $U' \| V'$ to B.

Step 3a: Upon receiving $U' \| V'$ from S, B computes $gxz = V' / H (B', A', S, gy)PwB$ and uses y to compute $gxyz = (gxz)y$. Then B computes $\alpha = H (B', A', gxyz)$ and forwards $U' \| \alpha$ to A.

Step 3b: Upon receiving $U' \| \alpha$ from B, A computes $gyz = U' / H (A', B', S, gx)PwA$ and uses x to compute $gxyz = (gyz)x$. A verifies $\alpha = H (B', A', gxyz)$. If the verification holds, A computes $\beta = H (A', B', gxyz)$ and the session key $SKA = H' (A', B', gxyz)$ and forwards β to B.

Step 3c: B computes $H (A', B', gxyz)$ and verifies β. If the verification holds B computes the session key $SKB = H' (A', B', gxyz)$. Figure 2. illustrates STPKE' protocol.

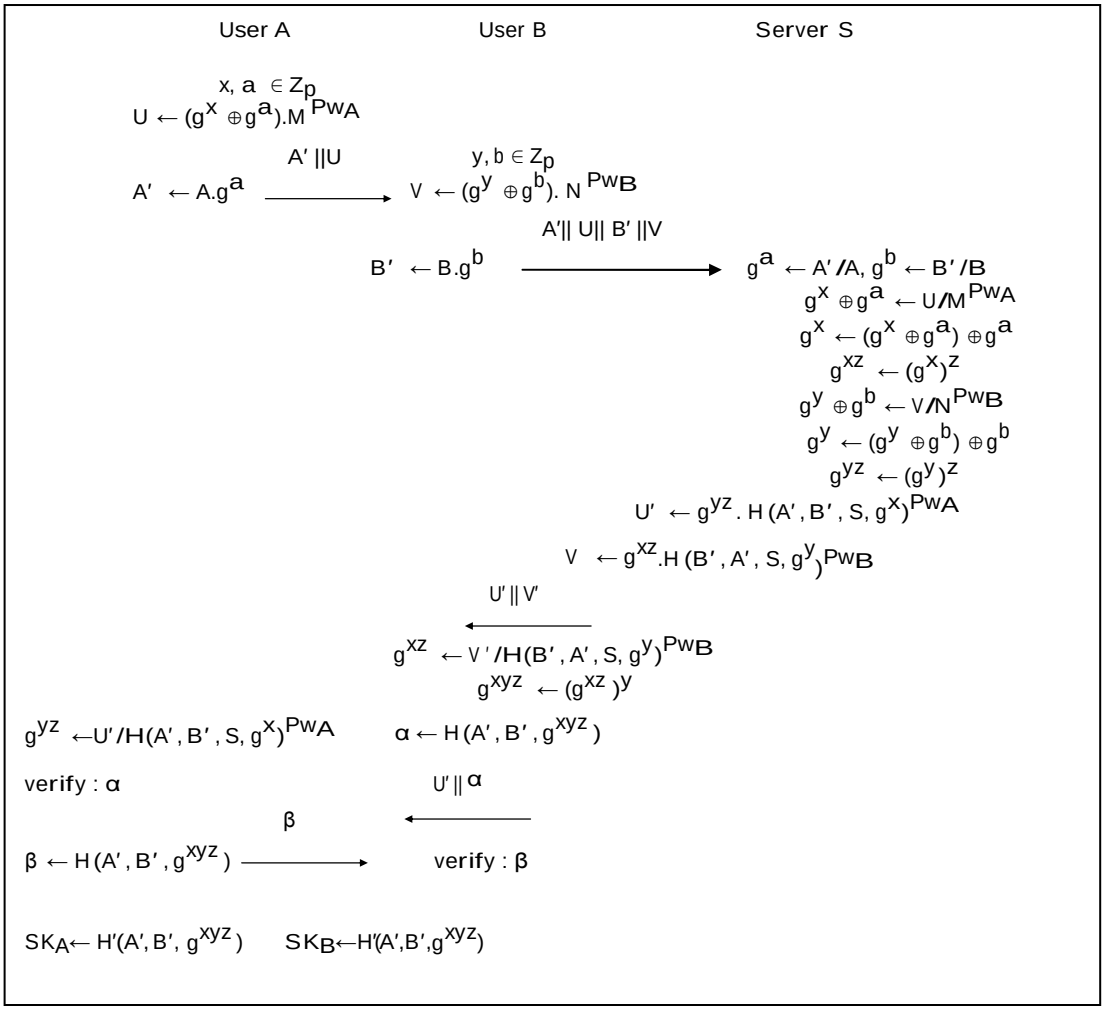

Figure 2. Kim and Choi's fixed STPKE' protocol

2.3. Review of modified STPKE' protocol.

Recently Tallapally and Padmavathy proposed a modified simple three party key exchange protocol (modified STPKE') to overcome the undetectable password guessing attack on Kim and Choi's (STPKE') protocol and S-3PAKE protocol proposed by Lu and Cao. They made the values of M and N secret between client and server as against the public information in S-3PAKE protocol and used parallel sessions for message transmission. The following are the detailed steps of the protocol.

Step 1a: A chooses a random number x ∈ Zp and computes U= gx.MPwA and sends A|| U to S.

Step 1b: B also chooses a random number y ∈ Zp and computes V= gy.NPwB and sends B|| V to S.

Step 2a: Upon receiving A|| U and B|| V , S uses PwA and PwB to compute gx = U/M PwA and gy = V/N PwB .

Step 2b: Then S chooses a random number z ∈ Zp and computes gxz = (gx)z and gyz = (gy)z Finally S computes U ′ = gyz .H (A, S, gx)PwA and V ′ = gxz .H (B, S, gy)PwB and sends U′ to A and V ′ to B.

Step 3a: B on receiving V ′,uses PwB to compute gxz = V′ / H (B, S, gy)PwB and uses the random number y to compute gxyz = (gxz)y and α= H (B, A, gxyz) and forwards α to A.

Step 3b: Upon receiving U′ , A computes gyz = U ′ /H (A, S, gx)PwA and uses x to compute gxyz = (gyz)x and verifies α. If the verification fails, A terminates the protocol, otherwise A computes the session key SKA = H′ (A, B, gxyz) and sends β = H (A, B, gxyz) to B.

Step 3c: B verifies β. If it holds, B computes the session key SKB = H′ (A, B, gxyz). Figure 3. illustrates modified STPKE' protocol.

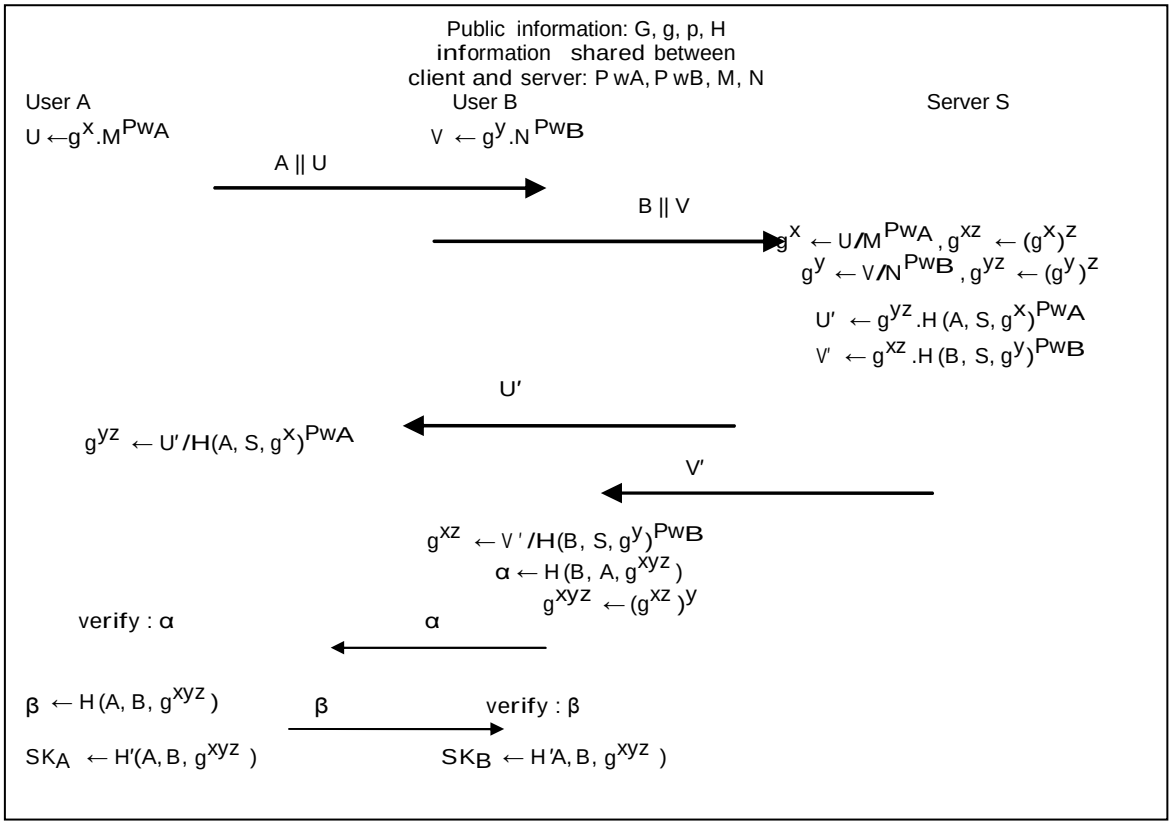

Figure3. Modified STPKE' protocol

3. Man in the middle attack on Modified STPKE' protocol

Padmavathy is vulnerable to man in the middle attack. We assume that an attacker C is registered with the trusted server S, shares password P WC and an element Q ∈ G with S and thus is able to set up normal protocol sessions with other clients.

Step 1a: A chooses a random number x ∈ Zp , and computes U= gx.M PwA and sends A|| U toS.

Step1b: The attacker C records the message A || U ,choose a random number I∈ Zp and

computes W = gl .QpwC and sends C|| W to S. (Q ∈ G) is shared between C and S.

Step1c: B chooses a random number y ∈ Zp ,computes V = gy .N PwB and sends B|| V to S.
Step 2a: Upon receiving C|| W and B|| V, S computes gl= W/QPwC and gy = V /N PwB .

Step 2b: Then S chooses a random number z ∈ Zp to compute glz = (gl)z and gyz = (gy) z and then S computes W′ = gyz .H (C, S, gl)PwC and V′= glz .H (B, S, gy) PwB and sends W′ to C and V ′ to B.
Step 3a: Upon receiving V′, B uses PwB to compute glz = V ′/H (B, S, gy)PwB and then uses y to compute glzy = (glz)y . Now, B sends α= H (B, A, glyz) to A to establish secret session key.
Step 3b: The attacker C receives α, uses PwC to compute gyz = W′ /H (C, S, gl)PwC and then uses l to compute glzy = (gyz)l , computes α from H (B, A, glyz) (C gets the identity of B by attacking B || Y sent by user B to S) and verifies the received and computed values. If they are same, computes β = H (A, B, glyz) and sends it to B impersonating as A. C also calculates the session key SK_A = H′(A, B, glyz).

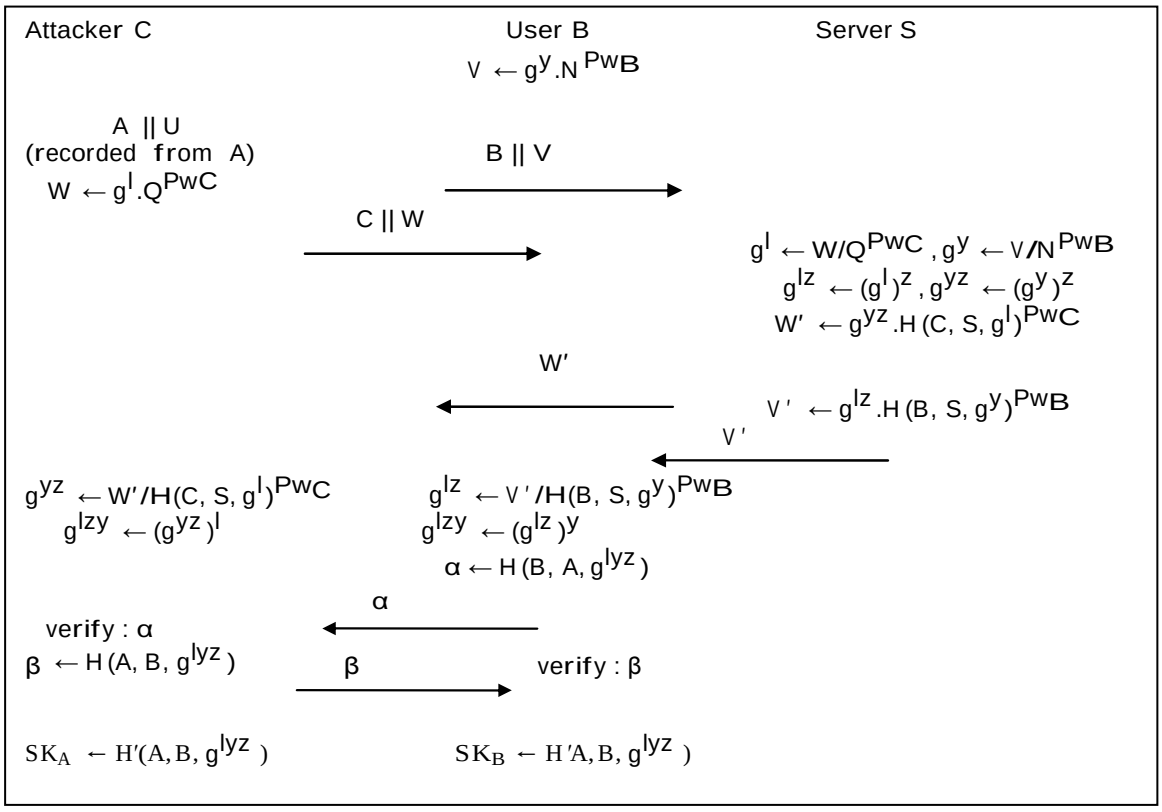

Figure 4. Man in the middle attack on Modified STPKE'protocol

Step 3c: B verifies β and authenticates A. B also computes the session key SKB = H′(A, B, glyz).

Thus, an attacker C can successfully impersonate user A and establish secret session key with B as shown in Figure 4. This session key can be used for all communications with B.

4. Proposed protocol

In this Section we propose an efficient three party key exchange protocol which can resist all the known attacks. For this purpose we break the password into two parts. Assume that two communicating parties A and B wish to agree a common session key. Let PwA be the

password shared between A and server S which is an arbitrary bit string. Here, A stores (PA , P1), while the server stores (PA , NA), where NA = gP1 and (PA , P1) = H (PwA , A, B). Similarly PwB be the password shared between B and S. Again, B stores (PB , P2), while the server stores (PB , NB), where NB = gP2 and (PB , P2) = H (PwB , A, B). Clients A and B can derive (PA , P1) and (PB , P2) from PwA and PwB respectively. The following are the steps of the protocol.

Step 1: A chooses a random number $x \in RZq$ and computes $U \leftarrow gx \oplus H(PA, A, B)$ and sends (U|| A)to B.

Step 2: B also chooses a random number $y \in RZq$ and computes $V \leftarrow gy \oplus H(PB, B, A)$ and sends (U || A), (V || B) to S.

Step 3: Upon receiving (U || A) and(V || B), S uses PA and PB to compute $gx \leftarrow U \oplus H(PA, A, B)$ and $gy \leftarrow V \oplus H(PB, B, A)$ respectively. Then S chooses a random number $z \in RZq$ to compute $gxz \leftarrow (gx)z$, $gyz \leftarrow (gy)z$ (NA)z , (NB)z .Finally, S computes $U'' \leftarrow gyz \oplus H(PA, A, B, gx, (NA)z)$ and $V' \leftarrow gxz \oplus H(PB, B, A, gy, (NB)z)$ and sends (U′ || gz) (V′ || gz) to B.

Step 4: B, on receiving the message uses P2 to calculate $(NB)z \leftarrow (gz)P2$ and computes $gxz \leftarrow V' \oplus H(PB, B, A, gy, (NB)z)$ and authenticates S. Now, B uses y to compute $K \leftarrow gxyz$, $\alpha \leftarrow H(B, A, K)$ and forwards U′ || gz || α to A.

Step 5: A, on receiving the message from B, uses P1 to compute $(NA)z \leftarrow (gz)P1$ and $gyz \leftarrow U' \oplus H(PA, A, B, gx, (NA)z)$ and authenticates the server. Then A uses x to compute $K \leftarrow gxyz$ and checks whether $\alpha \leftarrow H(B, A, K)$ holds or not. If it does not hold, A terminates the protocol, otherwise A is convinced that K is the valid session key. Then A computes $\beta \leftarrow H(A, B, K)$ and forwards it to B. Also, A computes the session Key $SKA \leftarrow H'(A, B, K)$.

Step 6: Upon receiving β, B computes $\beta \leftarrow H(A, B, K)$ and verifies whether computed β is equal to the received β. If both are equal then B authenticates A and computes the session key $SKB \leftarrow H'(A, B, K)$

5. Security Analysis

In this Section we analyze the security and efficiency of the proposed protocol.

1. Trivial attacks: Computing the session key from the transmitted messages α or β, is impossible due to the one-wayness of hash function. Also, for computing it from other transmitted messages U or V an attacker has to face the difficulty of discrete logarithm problem. So, our protocol is resistant to trivial attack.

2. Password guessing attacks: Suppose an attacker or a malicious user B try to guess A's password as P'_A , generates $g^{x'} \leftarrow U \oplus H(P'_A, A, B)$ and sends it to the server S in online transaction in step 2 of our protocol. To verify the correctness of his guessed password he needs to compute $gyz \leftarrow U' \oplus H(PA, A, B, gx, (NA)z)$ and $gxz \leftarrow V' \oplus H(PB, B, A, gy, (NB)z)$ which is impossible as
he needs the values of P1 and P2 for computing (NA)z and (NB)z Similarly remaining off-line also, using the transferred messages U, V, U′ , V′ , gz , an attacker cannot verify the correctness of his guessed password.

3. Man in the middle attack: In step 3 of our protocol, S authenticates the 2 communicating parties A and B from the messages $U \leftarrow gx \oplus H(PA, A, B)$ and $V \leftarrow gy \oplus H(PB, B, A)$ sent by B. A and B authenticate S, from $U' \leftarrow gyz \oplus H(PA, A, B, gx, (NA)z)$ and $V' \leftarrow gxz \oplus H(PB, B, A, gy, (NB)z)$ as PA , PB are known only to S. Finally, A authenticates B from $\alpha \leftarrow H(A, B, K)$. Thus, in each step of our protocol each party authenticates the other communicating party and hence there is no scope for man in the middle attack.

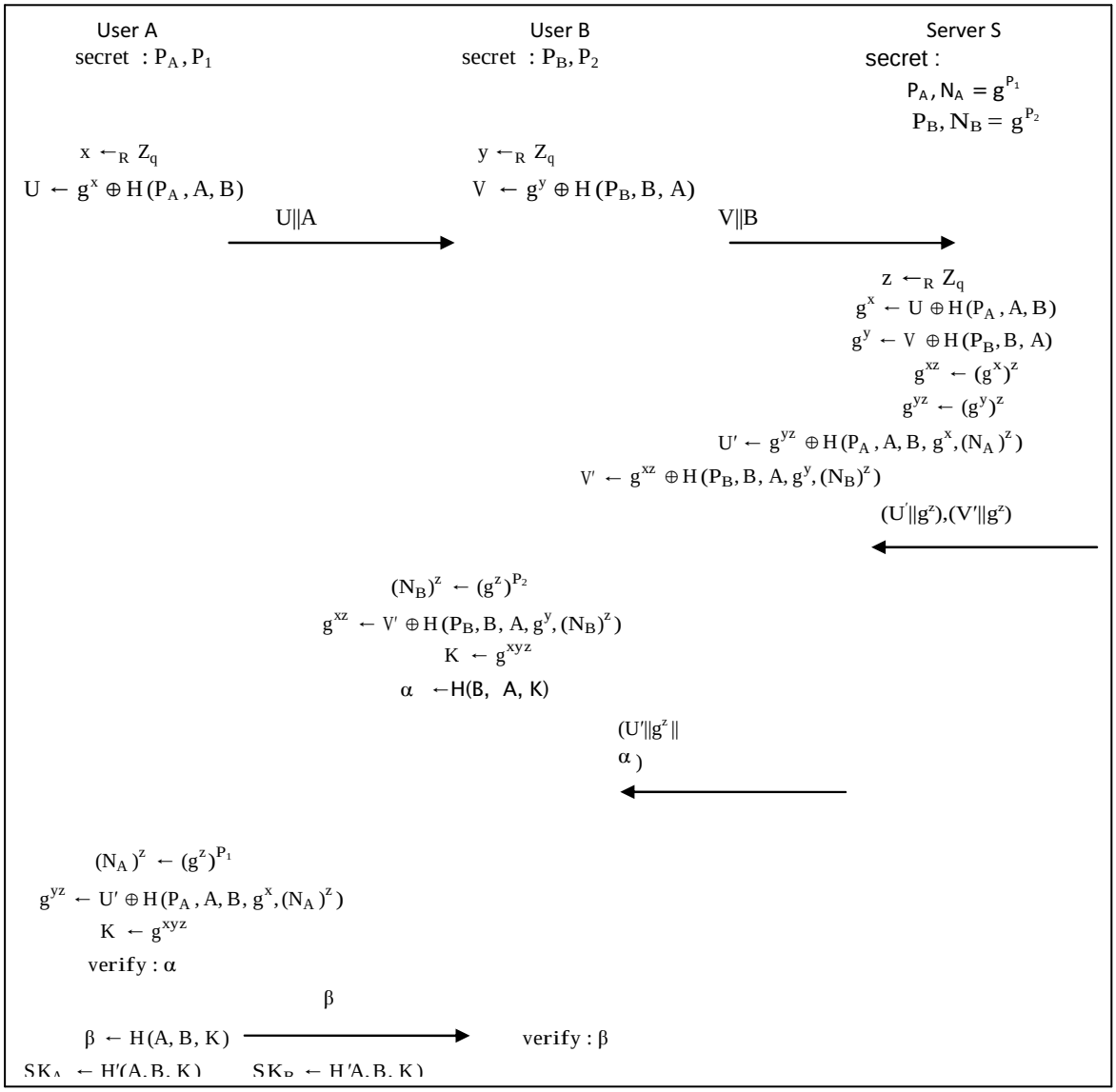

Figure 5. Proposed protocol

4. Replay attack: Since one way hash function is used, our proposed protocol is invulnerable to this attack.

5. Forgery attacks: In case the server is compromised, the attacker is required to compute gx ←U⊕ H (PA , A, B) and gy ← V ⊕ H (PB , B, A) where PA and PB are the passwords of A and B respectively. However it is not possible to compute these values without the knowledge of the passwords and hence A and B cannot construct the common session key.

6. Perfect forward secrecy: In case, the passwords PA and PB of the users A and B are compromised,
the attacker cannot calculate the session key as P1 and P2 are unknown. These values remain unknown even to the server and so there is no chance of any compromise. Also the session key is independent in each session and x, y, z are randomly chosen.

6. Formal Verification and Validation of Proposed Protocols

6.1. AVISPA.

We have validated the security of the proposed protocol using AVISPA tool.

Automated validation of internet security protocols and applications (AVISPA) [18]is a push button tool for the automated validation of security protocols. A modular and expressive formal language called HLPSL (High level protocols specification language) [19] is used by AVISPA to specify the security protocol and their properties. HLPSL is a role- based language, meaning that we first specify the sequence of actions of each kind of protocol participant in a module, which is called a basic role. This specification can later be instantiated by one or more agents playing the given role, and we further specify how the resulting participants interact with one another by combining multiple basic roles together into a composed role. HLPSL specification is translated into the Intermediate Format (IF), using hlpsl2if. The IF specification is then processed by model-checkers to analyze if the security goals are violated. There are four different verification back end tools use to analyze the IF specification namely, OFMC (On-the-Fly Model-Checker), CL-AtSe (Constraint-Logic-based Attack Searcher), SATMC (SAT-based Model-Checker), TA4SP (Tree Automata-based Protocol Analyser). Possible flaws in a protocol can be identified using these back end tools. As, exponential and XOR operations are supported by CL-AtSe and OFMC back ends we use OFMC back end tool with AVISPA and SPAN (Animation tool for AVISPA) to analyze the proposed protocols.

6.2. Specification and Verification of Proposed Protocol.

We verified the security of proposed protocol using AVISPA. For this we define three basic roles played by Alice (A), Bob (B) and Server (S). P WA and P WB are the passwords of A and B where P WA = (PA,P1) and P WB = (PB, P2). PA and PB are shared with S and hence represent the symmetric keys. P1 and P2 remain secret with A and B as their private keys. S shares NA = exp(G,P1) with A and NB = exp(G,P2) with B. Hence NA and NB are the public keys whose inverse is known only to A and B respectively. We then define the composed roles describing the sessions of the protocol and finally the top level role " environment role". For analyzing the protocol using AVISPA tool the following notations have been used.

g → G(value of g is stored in G)
x → X, y → Y
z → Z, I DA → A
I DB → B
The HLPSL specification for the proposed protocol is given below.

```
%% PROTOCOL:      E3PKE protocol%%
ALICE BOB SERVER:
%% Macros:
%% FM1: H(PA, A, B, exp(G,X), exp(NA,Z)),exp(G,Z)
%% FM2: H(PB, A, B, exp(G,Y), exp(NB,Z)), exp(G,Z)
%% Key: exp(exp(GY,Z),X) = exp(exp(GX,Z),Y)
%% GX : exp(G,X)
%% GY: exp(G,Y)
%% U: xor(exp(G,X),H(PA, A, B))
%% V: xor(exp(G,Y),H(PB, B, A))
%% U′ : xor(exp(GY, Z ), F M 1)
%% V′ : xor(exp(GX, Z ), F M 2)
The following are the message transactions:
%%1.A → B : U || A
%%2.B → S : (U || A), (V || B)
%%3.S → B : (U′ || exp(G, Z )), (V′ || exp(G, Z ))
%%4.B → A : U′ || exp(G, Z ) || α
%%5.A → B : β

%% HLPSL:
  role alice( A,B,S : agent, SND,RCV :
              channel(dy), H : hash func,
              PA : symmetric_key,
              G : text)
  played by A
  def=
  local State : nat, X,Z
              : text,
              NA : public_key, GY,
              Key : message,
              const sec_m Key : protocol id init
```

```
State := 0
transition
1. State = 0∧RCV(start)= | >
    State':= 1 ∧ X' := new()
                ∧ SND( xor(exp(G,X'),H(PA.A.B)))
2. State = 1 ∧ RCV(xor(exp(GY',Z'),H(PA.A.B.exp(G,X).exp(NA',Z')))).exp(G,Z')= | >
    State':= 2 ∧ Key' := exp(exp(GY',Z'),X)
                ∧ SND(H(A.B. Key'))
                ∧ witness(A,B,key1,Key')
3. State = 2 ∧ RCV(A.B.Key) = | >
    State':= 3 ∧ request(A,B,key,Key)
                ∧ secret(Key,sec_m,{A,B})
end role
role bob ( A,B,S : agent,
                SND,RCV : channel(dy), H :
                hash func,
                PA,PB : symmetric key, G
                :text)
played by B
def=
local State : nat,
                X,Y,Z : text, GX,GY
                : message, NB
                :public key, Key
                :message,
                FM1: hash(symmetric-key.agent.agent.message.message).message, FM2
                :hash(symmetric-key.agent.agent.message.message).message const sec v Key :
                protocol id
init State := 0
transition
1. State = 0 ∧ RCV( xor(exp(G,X'),H(PA.A.B)))= | >
    State':= 1 ∧ Y' := new()
                ∧ SND(xor(exp(G,X'),H(PA.A.B)).xor(exp(G,Y'),H(PB.A.B)))
2. State = 1 ∧ RCV(xor(exp(GY,Z'),FM1').xor(exp(GX',Z'),FM2'))= | >
        State':= 2 ∧ SND( xor(exp( GY,Z'),FM1'))
3. State = 2 ∧ RCV(H(A.B.exp(exp(GX',Z'),Y)))= | >
    State':= 3 ∧ Key' := exp(exp(GX',Z'),Y)
                ∧ SND(H(A.B. Key'))
                ∧ request(B,A,key1,Key)
                ∧ secret(Key,sec_v,{B,A})
                ∧ witness(B,A,key,Key')
end role
role server ( A,B,S : agent,
                SND,RCV : channel(dy), H :
                hash func,
                PA,PB : symmetric key, G :
                text)
played by S
def=
local State : nat,
                X,Y,Z : text,
                NA,NB : public key,
                GX,GY : message
init State := 0
transition
1. State = 0 ∧ RCV( xor(exp(G,X'),H(PA.A.B)).xor(exp(G,Y'),H(PB.A.B)))= | >
    State':= 1 ∧ Z' := new()
                ∧ NA' := new()
                ∧ NB' := new()
                ∧ GY' := new()
                ∧ GX' := new()
                ∧ SND(xor(exp(GY',Z'),H(PA.A.B.exp(G,X').exp(NA',Z')).exp(G,Z')).
                xor(exp(GX',Z'),H(PB.A.B.exp(G,Y').exp(NB',Z')).exp(G,Z')))
```

```
end role
role session( A,B,S : agent, H :
                    hash  func,
                    PA,PB : symmetric key,
                    NA,NB :public key,
                    G : text)
def=
local SND,RCV  : channel  (dy)
composition
alice(A,B,S,SND,RCV,H,PA,G)
∧ bob(A,B,S,SND,RCV,H,PA,PB,G)
∧ server(A,B,S,SND,RCV,H,PA,PB,G)
end role
role environment()
def=
consta,b,s  : agent,
        h : hash  func,
        key,key1 : protocol  id,
        pa,pb,pi  :symmetric key,
        na,nb,ni  :public key,
        g : text
intruder knowledge = a,b,s,g,h,pi,na,nb,ni
composition
session(b,a,s,h,pa,pb,na,nb,g)
∧ session(i,b,s,h,pi,pb,ni,nb,g)
∧ session(a,i,s,h,pa,pi,na,ni,g)
end role
goal
authentication  on key
authentication  on key1
secrecy of sec m Key, sec v Key end
goal
environment()
Running  the AVISPA tool on the proposed  protocol  returns the following output.


% OFMC
% Version of 2006/02/13
SUMMAR
    Y
    SAFE
DETAILS
    BOUNDED  NUMBER  OF SESSIONS
PROTOCOL
    C:\progra~1\testsuite\results\Etpke.if
GOAL
    as specified
BACKEND
    OFMC
COMMENTS  STATISTICS
    parseTime:  0.00s
    searchTime:  0.37s
    visitedNodes:  48 nodes
    depth:  7 plies
```

7. Conclusion

In this paper we have demonstrated man in the middle attack on modified STPKE' protocol proposed by Tallapally and Padmavathy. We have also suggested a countermeasure, an efficient three party key exchange protocol and showed that it is more secure and efficient than the existing protocols and can resist all the known attacks. Finally, we have also validated the proposed protocol using AVISPA, an automated tool for the verification of security protocols.

References

[1] Diffie, W. Hellman M. New directions in cryptography. IEEE Transactions on Information Theory. 1976; 22 : 644-654.
[2] Bellovin SM., Merritt M. Encrypted key exchange: password-based protocols secure against dictionary attacks. Proceedings of the 1992 IEEE symposium on research in security and privacy. 1992; 72-84.
[3] Ding Y., Horster P. Undetectable on-line password guessing attacks. ACM Operating Systems Review. 1995; 29 (4): 77-86.
[4] Steiner M., Tsudik G., Waidner M. Refinement and extension of encrypted key exchange. ACM Operating Systems Review. 1995; 29 (3): 22-30.
[5] Sun H.M., Chen B.C., Hwang T. Secure key agreement protocols for three-party against guessing attacks. The Journal of Systems and Software. 2005; 75 (12): 63-68.
[6] Lin CL, Sun HM, Hwang T. Three party-encrypted key exchange: attacks and a solution, ACM Operating Systems Review. 2000; 34 (4): 12-20.
[7] Lin C.L., Sun H.M., Steiner M., Hwang T. Three-party encrypted key exchange without server public- keys. IEEE Communications Letters. 2001; 5 (12): 497-499.
[8] Chang C.C, Chang YF. A novel three-party encrypted key exchange protocol. Computer Standards and Interfaces. 2004; 26 (5): 471-476.
[9] Lee, TF., Hwang T., Lin CL. Enhanced three-party encrypted key exchange without server public keys. Computers & Security. 2004; 23 (7): 571-577.
[10] Abdalla M., Pointcheval D. Simple password-based encrypted key exchange protocols. Proceedings of the CT-RSA05. 2005; LNCS, 3376: 191-208.
[11] Lu R., Cao Z. Simple three-party key exchange protocol. Computers Security. 2007; 26 (1): 94–97.
[12] Nam J., Paik J., Kang H.K., Kim U.M., Won D. An off-line dictionary attack on a simple three-party key exchange protocol. IEEE Communications Letters. 2009; 13 (3): 205-207.
[13] Chung H.R., Ku W.C. Three weaknesses in a simple three-party key exchange protocol. Information Sciences. 2008; 178 (1): 220-229.
[14] Debiao He, Jianhua Chen, Jin Hu Cryptanalysis of a simple three party key exchange Protocol. Informatica. 2010; 34 : 337–339.
[15] Hua Guo, Zhoujun Li, Yi Mu, Xiyong Zhang. Cryptanalysis of simple three party key exchange protocol. computers security. 2008; 27: 16–21.
[16] Kim H.S., Choi J.Y. Enhanced password-based simple three-party key exchange protocol. Computers and Electrical Engineering. 2009; 35 (1): 107-114.
[17] Tallapally S., Padmavathy R. Cryptanalysis on a three party key exchange protocol- STPKE'. Journal of Information processing systems. 2010; 06 (1: 43-52.
[18] Avispa-a tool for Automated Validation of Internet Security Protocols.http://www.avispa-project.org
[19] D6.2:Specificationof the Problems in the High-Level Specification Language. http://www.avispa-project.org