Simple Three Party Key Exchange Protocols via Twin Diffie-Hellman Problem

H. K. Pathak[1] and Manju Sanghi[2]
(Corresponding author: H. K. Pathak)
S.o.S in Mathematics, Pt. Ravishanker Shukla University, Raipur (C. G.) 492010, India[1]
Department of Applied Mathematics, Rungta College of Engg.& Tech., Bhilai (C. G.) 492006, India[2]
(Email: manjusanghi13@gmail.com)
(Received Oct. 5, 2011; revised and accepted Dec. 27, 2011)

Abstract

In 2005, Abdalla and Pointcheval suggested a new variation of the computational DH assumption called chosen based computational Diffie Hellman (CCDH) and presented SPAKE-1 and SPAKE-2 simple password based authenticated key exchange protocols. Since then several three party password authenticated key agreement protocols have been proposed based on CCDH assumption but most of them broken. In this paper, we propose two password based simple three party key exchange protocols via twin Diffie-Hellman problem and show that the proposed protocols provide greater security and efficiency than the existing protocols. The protocols are also verified using Automated Validation of Internet Security Protocols and Applications (AVISPA) which is a push button tool for the automated validation of security protocols and the result shows that they do not have any security flaws.

Keywords: AVISPA, password based key exchange protocols, password guessing attacks, SPAN, twin Diffie Hellman problem

1 Introduction

In the secure communication areas authenticated key exchange protocol is one of the most important cryptographic mechanism. In 1992, Bellovin and Merrit [5] proposed the first encrypted key exchange (EKE) family of key exchange protocols, which allow people to use easy to remember passwords without being threatened by dictionary attacks. In the EKE protocol, two communication parties securely share a password in advance, and authentication is achieved after these two parties obtain a common ephemeral session key. However, as the passwords are of low entropy, password based authenticated key exchange protocols are vulnerable to password guessing attacks [11].

Many approaches extended two party EKE protocols into three party EKE protocols, in which a trusted server is used to authenticate the communication parties. But most of them suffer from various attacks. In 1994, Steiner [19] gave the first three party encrypted key exchange (STW3PEKE) protocol with a trusted server S and clients A and B. But the works of Ding and Horster [11] (1995), Sun et al. [20] and Lin et al. [16] (2000) show that Steiner et al's 3PAKE protocol is vulnerable to undetectable on line password guessing attacks and off-line password guessing attacks. In 2005, Abdalla and Pointcheval [4] 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 [17] proposed a simple 3 party authenticated key exchange protocol (S-3PAKE) based on the concept of Abdalla and Pointcheval. But again, from the works of Nam et al., [18] Chung and Ku [8], Chen and Jin [13] and Guo Hua et al. [12] it is found that S-3PAKE protocol also suffers from various password guessing attacks as well as man in the middle attack. In 2009, Kim and Choi [15] proposed fixed STPKE′ protocol using exclusive-or operation as a counter measure for S-3PAKE protocol but recently Tallapally and Padmavathy [21] have proved that Kim and Choi's fixed STPKE′ is also vulnerable to undetectable online password guessing attack. Also, we find that Tallapally and Padmavathy's modified STPKE′ protocol also cannot resist man in the middle attack.

Recently, Cash, Kiltz and Shoup [7] proposed a new computational problem called twin Diffie-Hellman problem which has the following interesting properties:

- The twin Diffie-Hellman problem can easily be employed in many cryptographic constructions where one would usually use the ordinary Diffie-Hellman problem, without imposing a terrible efficiency penalty;
- The twin Diffie-Hellman problem is hard, even given access to a corresponding decision oracle, assuming the ordinary Diffie-Hellman problem (without access

to any oracles) is hard.

The heart of their method is a rapdoor test which can be used to implement an effective decision oracle for the twin Diffie-Hellman problem, without knowing any of the corresponding discrete logarithms. They applied the rapdoor test to many new variant protocols [6, 9, 10, 14] based on the Diffie-Hellman problem.

In this paper, motivated by cash et al's $SPAKE_2^+$ protocol we have proposed 2, three party, password authenticated, secure key exchange protocols based on twin Diffie-Hellman problem. Formal verification is the method for verification of security protocols to get the user confidence. There are many tools available for verification of the protocols. The analysis and verification of the proposed protocols is done using AVISPA (Automated validation of internet security protocols and applications) and SPAN.

The rest of the paper is organized as follows. In Section 2 we discuss twin DH problem. In Section 3 we present our proposed protocols along with their security analysis. Section 4 verifies the protocols using AVISPA and SPAN. Section 5 gives the performance comparison of various protocols and the paper is concluded in Section 6.

2 Password Authenticated Key Exchange Protocol based on Twin DH Problem

Cash, Kiltz and Shoup [7] suggested a new computational problem called twin Diffie-Hellman (DH) problem which is closely related to the usual (computational) DH problem and can be used in many of the same cryptographic constructions that are based on the DH problem. Moreover, the twin DH problem is atleast as hard as the ordinary DH problem. They presented ($SPAKE_2^+$) a very simple and efficient method of securing a password authenticated key exchange protocol of Abdalla and Pointcheval against server compromise, which can be proved secure using their trapdoor test, in the random oracle model, under the DH assumption.

2.1 Twin DH Problem

Let $\mathcal{G}$ be a cyclic group of prime order q with a generator g. Define $dh(X, Y) := Z$ where $X = g^x$, Y = g^y and Z = g^{xy}.

Given random $X, Y \in \mathcal{G}$, the problem of computing $dh(X, Y)$ is the DH problem. The DH assumption asserts that it is hard to compute $dh(X, Y)$ with random choice $X, Y \in \mathcal{G}$.

Define 2dh : $\mathcal{G}^3 \longrightarrow \mathcal{G}^2$
$(X_1, X_2, Y) \longrightarrow (dh(X_1, Y), dh(X_2, Y))$, which is called the twin DH function. The twin DH assumption states that it is hard to compute 2dh (X_1, X_2, Y), given random $(X_1, X_2, Y) \in \mathcal{G}$.

3 Proposed Protocols

In this Section we propose two secure key exchange protocols based on twin Diffie-Hellman assumption. The first proposed protocol (Protocol 1) is a variation of S-3PAKE protocol [17] proposed by Lu and Cao, which makes it immune against password guessing attacks and man in the middle attack. The second proposed protocol (Protocol 2) uses bit-wise exclusive or operation. We first introduce some notations and then describe the complete protocols.

Some notations used in the protocols are listed as follows:

($\mathcal{G}$, g, p): a finite cyclic group $\mathcal{G}$ generated by an element g of prime order p;

A, B: Two communicating parties;

S: Server;

M, N: Two elements in $\mathcal{G}$;

$Pw_A(P_A, P_1)$: Password of A, partially shared with S;

$Pw_B(P_B, P_2)$: Password of B, partially shared with S;

$\oplus$: bit-wise exclusive or operation;

H, H': Two secure one-way hash functions.

In this system assume that two communicating parties A and B wish to agree a common session key. Let Pw_A be the password shared between A and server S which is an arbitrary bit string. Here, A stores (P_A, P_1), while the server stores (P_A, U_A), where $U_A = g^{P_1}$ and $(P_A, P_1) = H(Pw_A, id_A, id_B)$. Similarly Pw_B be the password shared between B and S. Again, B stores (P_B, P_2), while the server stores (P_B, U_B), where $U_B = g^{P_2}$ and $(P_B, P_2) = H(Pw_B, id_A, id_B)$. Clients A and B can derive (P_A, P_1) and (P_B, P_2) from Pw_A and Pw_B respectively.

3.1 Protocol 1

The following are the detailed steps of the protocol as shown in Figure 1.

Step 1a. A chooses a random number $x \in_R \mathbb{Z}_q$ and computes $X \leftarrow g^x.M^{P_A}$, and then sends $id_A \parallel X \parallel id_B$ to B.

Step 1b. B also chooses a random number $y \in_R \mathbb{Z}_q$ and computes $Y \leftarrow g^y.M^{P_B}$, then sends $id_A \parallel X \parallel id_B \parallel Y$ to S.

Step 2a. Upon receiving $id_A \parallel X \parallel id_B \parallel Y$, S uses P_A and P_B to compute $g^x \leftarrow X/M^{P_A}$ and $g^y \leftarrow Y/N^{P_B}$ respectively.

Step 2b. Then, S chooses a random number $z \in_R \mathbb{Z}_q$ to compute $L \leftarrow g^z$, $g^{xz} \leftarrow (g^x)^z$, $g^{yz} \leftarrow (g^y)^z$. Now, S computes $X' \leftarrow g^{yz}.H(P_A, id_A, id_B, g^x, (U_A)^z)$ and $Y' \leftarrow g^{xz}.H(P_B, id_A, id_B, g^y, (U_B)^z)$ and sends $(X' \parallel L)$ and $(Y' \parallel L)$ to B.

Step 3a. B, on receiving the message, uses P_2 to compute $(U_B)^z \leftarrow L^{P_2}$ and $g^{xz} \leftarrow Y'/H(P_B, id_A, id_B, g^y, (U_B)^z)$ and authenticates S. Here, P_2 is the secret known only to B. Now, B uses y to compute $g^{xyz} \leftarrow (g^{xz})^y$ and $\alpha \leftarrow H(id_A, id_B, g^{xyz})$ and forwards $X' \parallel L, \alpha$ to A.

Step 3b. A, on receiving $(X' \parallel L, \alpha)$, uses P_1 to compute $(U_A)^z \leftarrow L^{P_1}$ and $g^{yz} \leftarrow X'/H(P_A, id_A, id_B, g^x, (U_A)^z)$ and authenticates the server. Here P_1 is the secret known only to A. Then A uses x to compute $g^{xyz} \leftarrow (g^{yz})^x$ and $\alpha \leftarrow H(id_A, id_B, g^{xyz})$ and verifies whether computed α is equal to the received α. If, both are equal, then A authenticates B, finds $\beta \leftarrow H(id_A, id_B, g^{xyz})$ and the session key $SK_A \leftarrow H'(id_A, id_B, g^{xyz})$ and forwards β to B.

Step 3c. Upon receiving β, B computes $\beta \leftarrow H(id_A, id_B, g^{xyz})$ and verifies whether computed β is equal to the received β. If both are equal then B authenticates A and computes the session key $SK_B \leftarrow H'(id_A, id_B, g^{xyz})$.

3.2 Security Analysis

The security of Protocol 1 mainly relies on the difficulty of twin Diffie-Hellman problem. Unlike, in other protocols the passwords of the clients are only partially shared with the server. This prevents most of the attacks.

1) Trivial attacks: An attacker may directly try to compute the passwords and/or the session key $SK = H'(id_A, id_B, g^{xyz})$ from the transmitted messages $(X, Y, X', Y', \alpha, \beta)$. But due to the difficulties of discrete logarithm problem, twin Diffie-Hellman problem and one-way ness of hash function, the trivial attack is not possible in our proposed protocol.

2) Online password guessing attacks: In online guessing attacks, an attacker tries to confirm a guessed password in an online transaction. In our proposed Protocol 1, the passwords of the clients are only partially and not completely shared with the server. The part P_1 and P_2 of the passwords are kept secret with the users A and B respectively and are not transmitted through any messages. Even if an attacker or a malicious user B tries to guess A's password he can only guess P_A get $X' = M^{P'_A}$ and send it in online transaction. But to verify the correctness of his guessed password he has to compute $H(P_A, id_A, id_B, g^x, (U_A)^z)$ which is impossible since he requires the value of P_1 for getting the value of $(U_A)^z$. Therefore online guessing attack is not possible on our proposed protocol.

3) Off-line password guessing attack: Assume that an attacker tries to mount off-line password guessing attack to guess the password. He intercepts the messages X and X' or Y and Y' but still he cannot verify his guessed password due to the difficulty of getting the values of P_1 or P_2 and one-way ness of hash function. Hence off-line password guessing attack is impossible in our scheme.

4) Man in the middle attack: The attacker in the middle attack involves interrupting a message and substituting it with his own message such that the communicating parties, without detecting the attacker compute the wrong session key. However authentication straight forwardly prevents this attack. For, suppose an attacker C tries to impersonate A and communicate with B, he computes $C = M^{Pw_c}$ and send $id_A \parallel C$ impersonating as A, in the subsequent steps to compute $g^{yz} \leftarrow X'/H(P_A, id_A, id_B, g^x, (U_A)^z)$ he needs the value of P_1 for calculating $(U_A)^z$ which is impossible to find. Hence there is no scope for man in the middle attack in our protocol.

5) Replay attack: In a replay attack, an attacker may want to pretend to be A by replaying X to B. However, as he does not know the password of user A and x, y, z are randomly chosen in each session, the attacker has no ability to derive $g^{yz} \leftarrow X'/H(P_A, id_A, id_B, g^x, (U_A)^z)$ and produce a valid session key $SK_A = H'(id_A, id_B, K)$ where $K = g^{xyz}$. Similarly the attacker also cannot pretend to be B. The replay attack will hence fail.

6) Forgery attacks: If a masked server tries to deceive the requesting users A and B he has to obtain the validity of messages $g^x, g^y, H(P_A, id_A, id_B, g^x, (U_A)^z)$, $H(P_B, id_A, id_B, g^y, (U_B)^z)$ in Step 2 of our protocol where P_A and P_B are the passwords of A and B respectively. However, without knowing the users passwords P_A, P_B and also $(U_A)^z and (U_B)^z$, it is not easy for a masked server to compute $H(P_A, id_A, id_B, g^x, (U_A)^z)$ and $H(P_B, id_A, id_B, g^y, (U_B)^z)$ exactly so that users A and B can construct the common session key. Also, in the case of untrusted server, since the password of the user is not completely shared with the server, to get the value of P_1 from g^{P_1}, an attacker has to face the difficulty of discrete logarithm problem.

7) Perfect forward secrecy: Even if the passwords P_A and P_B of the users are compromised, the attacker cannot calculate the session key as P_1 and P_2 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.

3.3 Protocol 2

Figure 2 illustrates our proposed Protocol 2. This protocol uses $\oplus$ a bit-wise exclusive or operation. The following are the steps of the protocol.

Step 1a. A chooses a random number $x \in_R \mathbb{Z}_q$ and computes $N_A \leftarrow g^x \oplus H(P_A, id_A, id_B)$ and sends $(N_A, id_A)$to B.

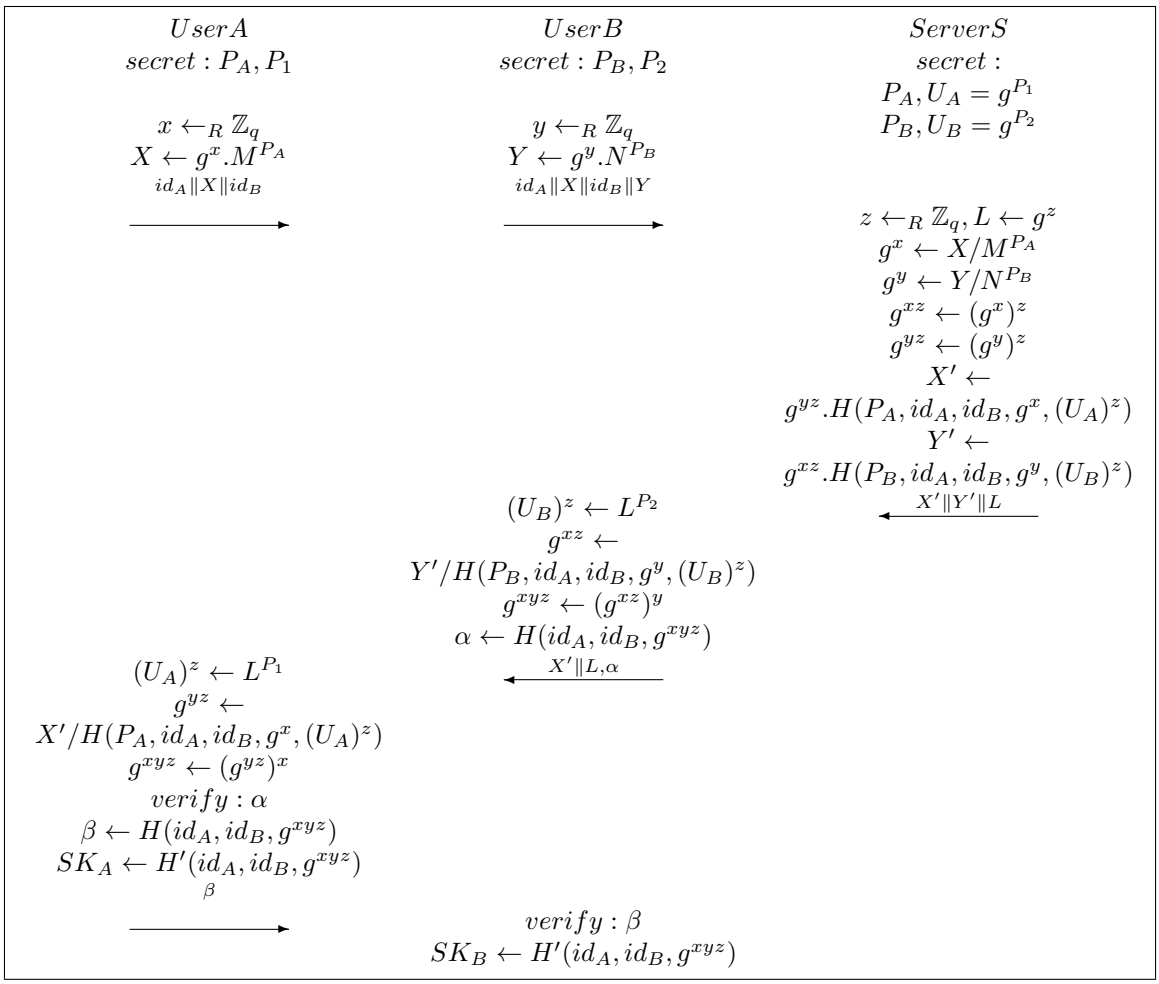

Figure 1: Proposed Protocol 1

Step 1b. B also chooses a random number $y \in_R \mathbb{Z}_q$ and computes $N_B \leftarrow g^y \oplus H(P_B, id_A, id_B)$ and sends (N_A, id_A),(N_B, id_B) to S.

Step 2a. Upon receiving (N_A, id_A) and(N_B, id_B), S uses P_A and P_B to compute $g^x \leftarrow N_A \oplus H(P_A, id_A, id_B)$ and $g^y \leftarrow N_B \oplus H(P_B, id_A, id_B)$ respectively.

Step 2b. Then S chooses a random number $z \in_R \mathbb{Z}_q$ to compute $L \leftarrow g^z$,$(U_A)^z$, $(U_B)^z$, $a \leftarrow g^{xz} \leftarrow (g^x)^z$, $b \leftarrow g^{yz} \leftarrow (g^y)^z$. Then S computes $Z_A \leftarrow b \oplus H(P_A, id_A, id_B, g^x, (U_A)^z)$ and $Z_B \leftarrow a \oplus H(P_B, id_A, id_B, g^y, (U_B)^z)$ and sends $(Z_A, L), (Z_B, L)$ to B.

Step 3a. B, on receiving the message uses P_2 to compute $(U_B)^z \leftarrow L^{P_2}$ and $a \leftarrow Z_B \oplus H(P_B, id_A, id_B, g^y, (U_B)^z)$ and authenticates S. Now, B uses y to compute $K \leftarrow g^{xyz} \leftarrow a^y$, $\alpha \leftarrow H(id_A, id_B, K)$ and forwards $(Z_A, L), \alpha$ to A.

Step 3b. A, on receiving (Z_A, L, α) uses P_1 to compute $(U_A)^z \leftarrow L^{P_1}$ and $b \leftarrow Z_A \oplus H(P_A, id_A, id_B, g^x, (U_A)^z)$ and authenticates the server. Then A uses x to compute $K \leftarrow g^{xyz} \leftarrow b^x$ and checks whether $\alpha \leftarrow H(id_A, id_B, K)$ 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 $\beta \leftarrow H(id_A, id_B, K)$ and forwards it to B. Also, A computes the session Key $SK_A \leftarrow H'(id_A, id_B, K)$.

Step 3c. Upon receiving β, B computes $\beta \leftarrow H(id_A, id_B, K)$ and verifies whether computed β is equal to the received β. If both are equal then B authenticates A and computes the session key $SK_B \leftarrow H'(id_A, id_B, K)$.

3.4 Security Analysis

1) Trivial attacks: Computing the session key from the transmitted messages α or β, is impossible due to the one-way ness of hash function. Also, for computing it from other transmitted messages Z_A or Z_B 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

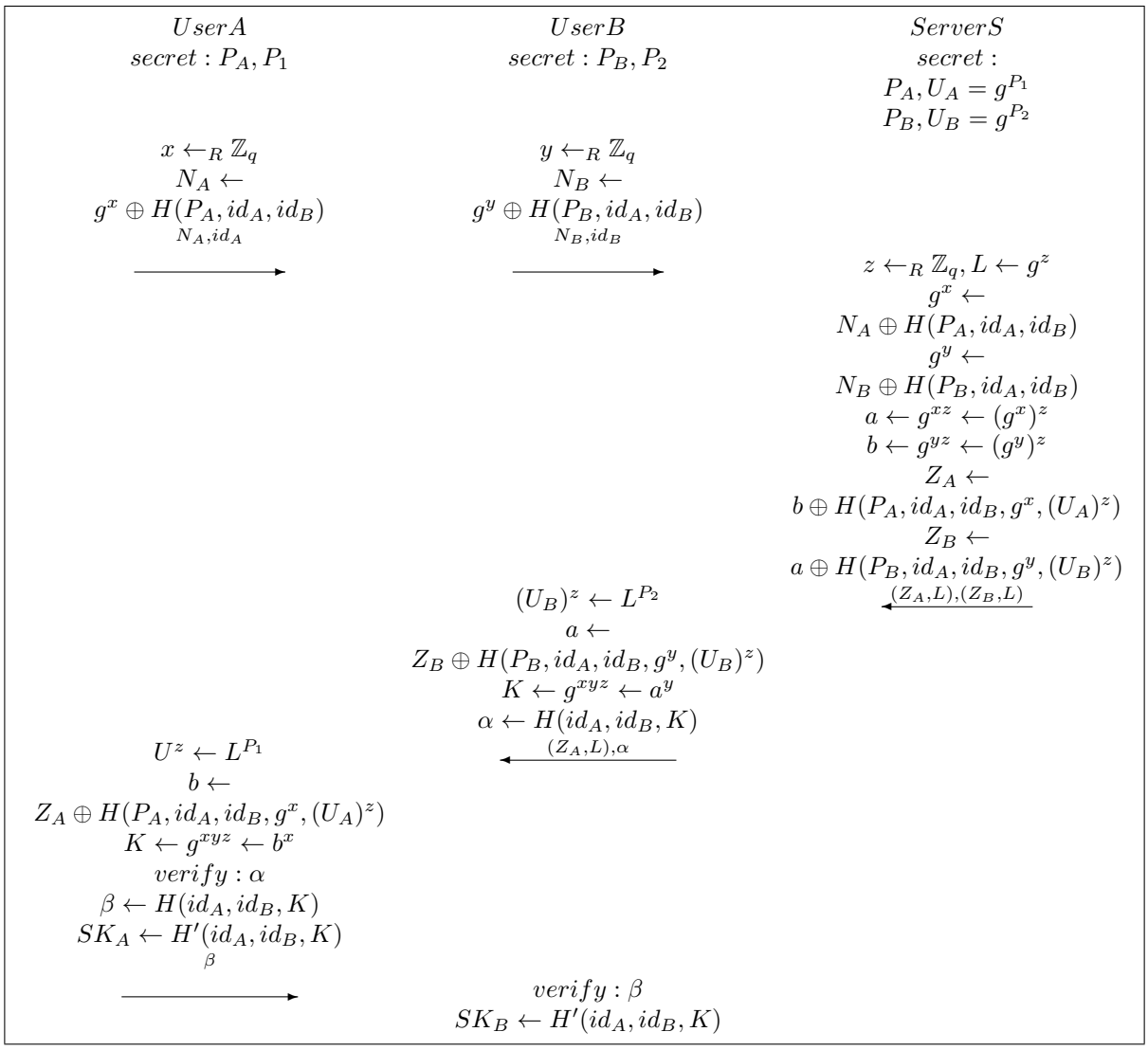

Figure 2: Proposed Protocol 2

P'_A, generates $g^{x'} \leftarrow N_A \oplus H(P'_A, id_A, id_B)$ and sends it to the server S in online transaction in Step 1 of our Protocol 2. To verify the correctness of his guessed password he needs to compute $b \leftarrow Z_A \oplus H(P_A, id_A, id_B, g^x, (U_A)^z)$ and $a \leftarrow Z_B \oplus H(P_B, id_A, id_B, g^y, (U_B)^z)$ which is impossible as he needs the values of P_1 and P_2 for computing $(U_A)^z$ and $(U_B)^z$. Similarly remaining off-line also, using the transferred messages N_A, N_B, Z_A, Z_B, L, an attacker cannot verify the correctness of his guessed password.

3) Man in the middle attack: In Step 2 of our protocol, S authenticates the 2 communicating parties A and B from the messages $N_A \leftarrow g^x \oplus H(P_A, id_A, id_B)$ and $N_B \leftarrow g^y \oplus H(P_B, id_A, id_B)$ sent by B. A and B authenticate S, from $Z_A \leftarrow b \oplus H(P_A, id_A, id_B, g^x, (U_A)^z)$ and $Z_B \leftarrow a \oplus H(P_B, id_A, id_B, g^y, (U_B)^z)$ as P_A, P_B are known only to S. Finally, A authenticates B from $\alpha \leftarrow H(id_A, id_B, K)$. 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.

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 $g^x \leftarrow N_A \oplus H(P_A, id_A, id_B)$ and $g^y \leftarrow N_B \oplus H(P_B, id_A, id_B)$ where P_A and P_B 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 P_A, and P_B of the users A and B are compromised, the attacker cannot calculate the session key as P_1 and P_2 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.

4 Formal Verification and Validation of Proposed Protocols

4.1 AVISPA

Automated validation of internet security protocols and applications (AVISPA) [1] 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) [3] 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) [2] to analyze the proposed protocols.

4.2 Specification and Verification of Proposed Protocol

We verified the security of proposed protocols using AVISPA and SPAN. For this we define three basic roles played by Alice (A), Bob (B) and Server (S). PW_A and PW_B are the passwords of A and B where PW_A = (PA,P1) and PW_B = (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 gets UA = exp(G,P1) from A and UB = exp(G,P2) from B. Hence UA and UB 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". SPAN is used to symbolically execute the HLPSL protocol specification and hence provides a better understanding of the specification. For analyzing the protocol using AVISPA tool the following notations have been used.

$g \rightarrow G$(value of g is stored in G)
$x \rightarrow X, y \rightarrow Y$
$z \rightarrow Z, IDA \rightarrow A$
$IDB \rightarrow B$

Running the AVISPA and SPAN tool on the proposed protocol(Protocol 2)returns the following output.

% OFMC
% Version of 2006/02/13
SUMMARY
SAFE
DETAILS
BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
C:\progra~1\SPAN\testsuite\results\PP2.if
GOAL
as_specified
BACKEND
OFMC
COMMENTS STATISTICS
parseTime: 0.00s
searchTime: 0.37s
visitedNodes: 48 nodes
depth: 7 plies

Similarly Protocol 1 can be verified using AVISPA.

5 Security and Efficiency of S-3PKE Protocols

In this Section we discuss the efficiency and security of the proposed protocols and existing competitive protocols based on Abdalla and Pointcheval's concept. We include the following protocols: Lu and Cao's S-3PAKE protocol [17] (2007), Kim and Choi's Fixed STPKE′ protocol [15] (enhancement of S-3PAKE)(2009), Tallapally and Padmavathy's Modified STPKE′ protocol [21] an enhancement of fixed STPKE′ protcol (2010), HS-3PAKE protocol [14] by Yoon and Yoo (2011) and proposed protocols, Protocol 1 and Protocol 2.

From Table 1, we find that proposed Protocol 1 uses 5 rounds of communication, requires the computation of 15 exponentiation operations and 10 hash functions without the use of xor operations, while Protocol 2 also uses 5 rounds of communication and requires the computation of only 11 exponentiation operations which is less than the requirement of the existing protocols along with the hash functions and xor operations.

In Table 2 we list known attacks for each protocol. An attack weakens the security of the protocol. We find that almost all the existing protocols are prone to different attacks.

6 Conclusion

Although many password based key exchange protocols are being developed, most of them are vulnerable to vari-

Table 1: Performance of S-3PKE protocols

Protocol	Random no. $A\backslash B\backslash S$	Mod.expon. $A\backslash B\backslash S$	Hash function $A\backslash B\backslash S$	XOR operation	Round efficiency
S-3 PAKE	1 1 1	4 4 6	4 4 2	0 0 0	5
Fixed STPKE'	2 2 1	5 5 6	4 4 2	1 1 2	5
Modified STPKE'	1 1 1	4 4 6	4 4 2	0 0 0	4
HS-3PAKE	1 1 1	2 2 2	4 4 4	2 2 4	5
Proposed Protocol 1	1 1 1	4 4 7	4 4 2	0 0 0	5
Proposed Protocol 2	1 1 1	3 3 5	4 4 4	2 2 4	5

Table 2: Known attacks on the existing protocols

Protocol	Known attacks
S-3PAKE protocol	Undetectable online dictionary attack by Guo et al.(2008) Off line dictionary attack by Nam et al. (2009) Off line dictionary attack by Debiao et al. (2010) Impersonation of initiator attack Impersonation of responder attack Man in the middle attack by Chung and Ku (2008)
Fixed STPKE'protocol	Undetectable online password guessing attack by Tallapally(2010)
Modified STPKE'protocol	Man in the middle attack
HS-3PAKE protocol	Undetectable on line password guessing attack Off line password guessing attack by Yoon and Yoo (2011)
Proposed Protocol 1	—
Proposed Protocol 2	—

ous attacks. With the increasing need for authentication and secure communication we have proposed two simple three party key exchange protocols via twin Diffie-Hellman problem and showed that they are more secure and efficient than the existing protocols and can resist all the known attacks. Finally, we have also validated the proposed protocols using AVISPA, an automated tool for the verification of security protocols.

References

[1] "Avispa - a tool for automated validation of internet security protocols,". http://www.avispa-project.org.

[2] "Span - a security protocol animator for avispa,". http://www.irisa.fr.

[3] "Specification of the problems in the high-level specification language,". http://www.avispa-project.org.

[4] M. Abdalla and D. Pointcheval, "Simple password-based encrypted key exchange protocols," in *Proceedings of the CT-RSA, YEAR =.*

[5] S. M. Bellovin and M. Merritt. "Encrypted key exchange:password-based protocols secure against dictionary attacks,", 1992.

[6] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *Crypto' 01*, vol. LNCS 2139, pp. 213–229, 2001.

[7] D. Cash, E. Kiltz, and V. Shoup, "The twin diffie-hellman problem and applications," *Journal of Cryptology*, vol. 22, pp. 470–504, 2009.

[8] H. R. Chung and W. C. Ku, "Three weaknesses in a simple three-party key exchange protocol," *Information Sciences*, vol. 178, no. 1, pp. 220–229, 2008.

[9] R. Cramer and V. Shoup, "A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack," in *Crypto' 98*, vol. LNCS 1462, pp. 13–25, Berlin, 1998.

[10] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, pp. 644–654, 1976.

[11] Y. Ding and P. Horster, "Undetectable on-line password guessing attacks," *ACM Operating Systems Review*, vol. 29, no. 4, pp. 77–86, 1995.

[12] H. Guo, Z. Li, Y. Mu, and X. Zhang, "Cryptanalysis of simple three party key exchange protocol," *Computers Security*, vol. 27, pp. 16–21, 2008.

[13] D. He, J. Chen, and J. Hu, "Cryptanalysis of a simple three party key exchange protocol," *Informatica*, vol. 34, pp. 337–339, 2010.

[14] H. F. Huang, "A simple three-party password-based key exchange protocol," *International Journal of Communication Systems*, vol. 22, pp. 857–862, 2009.

[15] H. S. Kim and J. Y. Choi, "Enhanced password-based simple three-party key exchange protocol," *Computers and Electrical Engineering*, vol. 35, no. 1, pp. 107–114, 2009.

[16] C. L. Lin, H. M. Sun, and T. Hwang, "Three party-encrypted key exchange:attacks and a solution," *ACM Operating Systems Review*, vol. 34, no. 4, pp. 12–20, 2000.

[17] R. Lu and Z. Cao, "Simple three-party key exchange protocol," *Computers Security*, vol. 26, no. 1, pp. 94–97, 2007.
[18] J. Nam, J. Paik, H. K. Kang, U. M. Kim, and D. Won, "An off-line dictionary attack on a simple three-party key exchange protocol," *IEEE Communications Letters*, vol. 13, no. 3, pp. 205–207, 2009.
[19] M. Steiner, G. Tsudik, and M. Waidner, "Refinement and extension of encrypted key exchange," *ACM Operating Systems Review*, vol. 29, no. 3, pp. 22–30, 1995.
[20] H. M. Sun, B. C. Chen, and T. Hwang, "Secure key agreement protocols for three-party against guessing attacks," *The Journal of Systems and Software*, vol. 75, no. 2, pp. 63–68, 2005.
[21] S. Tallapally and R. Padmavathy, "Cryptanalysis on a three party key exchange protocol-stpke'," *Journal of Information processing systems*, vol. 6, no. 1, pp. 43–52, 2010.

Appendix

The HLPSL specification for the proposed Protocol 2:

%% PROTOCOL: S3PKE protocol using twin DH
%% ALICE_BOB_SERVER:
%% Macros:
%%$FM1$: $H(PA.A.B.exp(G,X).exp(UA,Z)).exp(L,Z)$
%%$FM2$: $H(PB.A.B.exp(G,Y).exp(UB,Z).exp(L,Z)$
%%Key: $exp(exp(GY,Z),X) = exp(exp(GX,Z),Y)$
%%GX: $exp(G,X)$
%%GY: $exp(G,Y)$
%%1.$A \rightarrow B : xor(exp(G,X),H(PA.A.B))$
%%2.$B \rightarrow S : xor(exp(G,X),H(PA.A.B))$,
%% $xor(exp(G,Y),H(PB.A.B))$
%%3.$S \rightarrow B : xor(exp(GY,Z),FM1)$,
$xor(exp(GX,Z)FM2)$
%%4.$B \rightarrow A : xor(exp(GY,Z),FM1).H(A.B.Key)$
%%5.$A \rightarrow B : H(A.B.Key)$
%% 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,
UA : public_key,
GY, Key, L : 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 ∧ $RCV(xor(exp(GY',Z'),$
$H(PA.A.B.exp(G,X).exp(UA',Z'))).L) = | >$
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 ∧ $request(A,B,key,Key)$
∧ $secret(Key,sec\_m\_Key,A,B)$
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,
UB :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 ∧ $Key' := exp(exp(GX',Z'),Y)$
∧ $SND(H(A.B.Key'))$
∧ $request(B,A,key1,Key)$
∧ $secret(Key,sec\_v\_Key,B,A)$
∧ $witness(B,A,key,Key')$
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,
UA, UB : 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 ∧ $Z' := new()$
∧ $UA' := new()$

$\wedge\ UB' := new()$
$\wedge\ GY' := new()$
$\wedge\ GX' := new()$
$\wedge\ SND(xor(exp(GY', Z'),$
$H(PA.A.B.exp(G, X').exp(UA', Z')).$
$exp(G, Z')).$
$xor(exp(GX', Z'),\ H(PB.A.B.$
$exp(G, Y').exp(UB', Z')).exp(G, Z')))$
end role
role session(A, B, S : agent,
H : hash_func,
PA, PB : symmetric_key,
UA, UB :public_key,
G : text)
def=
local SND, RCV : channel (dy)
composition
alice$(A, B, S, SND, RCV, H, PA, G)$
$\wedge$ bob$(A, B, S, SND, RCV, H, PA, PB, G)$
$\wedge$ 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,
ua, ub, ui :public_key,
g : text
intruder_knowledge = $\{a, b, s, g, h, pi, ua, ub, ui\}$
composition
session$(b, a, s, h, pa, pb, ua, ub, g)$
$\wedge$ session$(i, b, s, h, pi, pb, ui, ub, g)$
$\wedge$ session$(a, i, s, h, pa, pi, ua, ui, g)$
end role
goal
authentication_on key
authentication_on key1
secrecy_of sec_m_Key, sec_v_Key
end goal
environment()

H. K. Pathak received Post Graduate degree in Mathematics from Pt. Ravishanker Shukla University, Raipur. He was awarded Ph.D in 1988 by the same University. He has published more than 185 research papers in various international journals in the field of non linear analysis-Approximation and expansion, Calculus of variations and optimal controls Optimization, Field theory and polynomials, Fourier analysis, General topology, Integral equations, Number theory, Operations research, Mathematical programming, Operator theory, Sequences, series, summability. At present he is Professor and Head in S.o.S in Mathematics in Pt. Ravishanker Shukla University.

Manju Sanghi received the post graduate degree in Mathematics from Ravishanker Shukla University, Raipur in 1996. Since 2001 she has been working as lecturer in Rungta college of Engineering & Technology Bhilai. Currently she is pursuing PhD from School of studies in Mathematics Pt. Ravishanker Shukla University Raipur. Her research interests include cryptography, Network security.