SECURITY ENHANCEMENT FOR 3-PEKE PROTOCOL USING PARALLEL MESSAGE TRANSMISSION TECHNIQUE

P.RAJKUMAR
Department Of Computer science and Engineering,
Info Institute of Engineering, kovilpalayam,
Coimbatore-641 107, Tamilnadu, India
mpr_rajkumar83@yahoo.co.in

Dr.C.MANOHARAN
Annai Mathammal sheela engineering college, Tamil Nadu, India
E-mail:c_m_66@yahoo.co.in

Abstract

It is easy to choose and memorize simple and meaningful vocabulary as secret passwords, but it is very hard to meet the requirement of security and efficiency. Hence chang and chang proposed a novel three party encrypted key exchange protocol without using the server’s public keys. The key exchange protocol has achieved great attention due to its simplicity and efficiency. It was claimed that the protocol was practically secure and efficient. On the other hand, the protocol resists all types of password guessing attacks, because the password is of low entropy. It was then Yoon and Yoo who demonstrated the vulnerability of chang and chang’s protocol regarding undetectable on line password guessing attacks. This paper presents an enhanced protocol to eliminate undetectable online password guessing attack proposed by Yoon and Yoo.it also proposed an enhanced protocol which could achieve better performance by requiring only four message transmission round and the performance is analyzed on a set of experiments.

Key words: 3-PEKE, Password authenticated key exchange (PAKE)

1. Introduction

In cryptography, a password-authenticated key agreement method is an interactive method for two or more parties to establish cryptographic keys based on one or more party's knowledge of a password. Password-authenticated key agreement generally encompasses methods such as: Balanced password-authenticated key exchange augmented password-authenticated key exchange, Password-authenticated key retrieval, Multi-server methods, and Multi-party methods. In the most stringent password-only security models, there is no requirement for the user of the method to remember any secret or public data other than the password. Password authenticated key exchange (PAKE) is where two or more parties, based only on their knowledge of a password, establish a cryptographic key using an exchange of messages, such that an unauthorized party (one who controls the communication channel but does not possess the password) cannot participate in the method and is constrained as much as possible from guessing the password. (The optimal case yields exactly one guess per run exchange.) Two forms of PAKE are Balanced and Augmented methods. Balanced PAKE allows parties that use the same password to negotiate and authenticate a shared key. Examples of these are:
Encrypted Key Exchange (EKE), PAK and PPK, SPEKE (Simple password exponential key exchange) J-PAKE (Password Authenticated Key Exchange by Juggling), Augmented PAKE is a variation applicable to client/server scenarios, in which an attacker must perform a successful brute-force attack in order to masquerade as the client using stolen server data. Examples of these are:
Augmented-EKE B-SPEKE, PAK-Z, the Encrypted Key Exchange (EKE) Protocol,
• Provides security and authentication on computer networks using both symmetric and public key cryptography.
• Main idea: a shared secret key is used to encrypt a randomly generated public key.
• Basic EKE: A and B share a common password P and use EKE to authenticate each other and generate a common session key K.
M1. $A \rightarrow B : E_P (K_A)$
M2. $B \rightarrow A : E_P (E_{KA}(K))$
M3. $A \rightarrow B : E_K(N_1)$

M4. B → A : $E_K(N_1,N_2)$
M5. A → B : $E_K(N_2)$

Where P is used as symmetric key, K_A is a randomly generated public key (with corresponding $K^{-1}{}_A$), and K is a randomly generated session key.
• EKE can be implemented with different public-key algorithms: RSA, Diffie-Hellman, ElGamal, etc.
• For example, with Diffie-Hellman, given (g, p), K is generated automatically, and the protocol is even simpler: A does not have to encrypt the first message with password P; she picks a random number r_A and sends M1. A → B: A, g^{rA} mod p, B picks a random number r_B and calculates K = $g^{rA\ rB}$ mod p. He then generates a random string R_B and sends M2. B → A: E_P (g^{rB} mod p), E_K (R_B) , A decrypts to obtain g^{rB} mod p, and then calculates K to decrypt R_B. She generates another random string R_A and sends M3. A →B: E_K (R_A, RB) B checks that R_B is the one he sent and replies to A's challenge M4. B →A: $E_K(R_A)$
• EKE is patented and its use has been suggested for secure public telephones.
• It does suffer from some weaknesses, though: EKE's challenge-response can be strengthened to prevent replay attacks on K. Moreover, EKE requires that both parties possess password P; the Augmented EKE Protocol uses a one-way hash of the user's password as the super-encryption in the Diffie-Hellman variant of EKE.

Encrypted Key Exchange (also known as EKE) is a family of password-authenticated key agreement methods described by Steven M. Bellovin and Michael Merritt.[1] Although several of the forms of EKE were later found to be flawed, the surviving, refined, and enhanced forms of EKE effectively make this the first method to amplify a shared password into a shared key, where the shared key may subsequently be used to provide a zero-knowledge password proof or other functions. In the most general form of EKE, at least one party encrypts an ephemeral (one-time) public key using a password, and sends it to a second party, who decrypts it and uses it to negotiate a shared key with the first party.

In 2012, P.Rajkumar and C.Manoharan [2] proposed 3PEKE protocol (PMT). It is shown that STW-3PEKE suffers not only undetectable on-line password guessing attacks but also off-line password guessing attacks. Then, they proposed an improved 3PEKE protocol (LSH-3PEKE) using server's public key to prevent these attacks. Thereafter, in 2001, Lin et al. [5] proposed a new 3PEKE protocol (LSSH-3PEKE) without the use of server's public keys.

Chang and Chang [4] also proposed a three-party encrypted key exchange (ECC-3PEKE) protocol without using the server's public keys that provides the round efficiency and possesses the advantages of LSH-3PEKE [5] and LSSH-3PEKE [6]. Chang–Chang claimed that their proposed ECC-3PEKE protocol is secure, practical, simultaneously possesses round and computation efficiencies. But an undetectable on-line password guessing was proved on ECC protocol by Yoon and Yoo.
For security enhancement, we propose an enhanced protocol which utilizes the parallel message transmission mechanism to achieve better performance efficiency by reducing one message transmission round compared to Chang and Chang 3PEKE protocols without undetectable on-line password guessing attacks, and the performance is analyzed on a Comprehensive set of experiments.

2. The existing protocol

This section briefly reviews of Chang–Chang's ECC-3PEKE protocol. Some of the notations used are defined as follows:
• A, B: two communication parties.
• S: the trusted server.
• ID_A, ID_B, ID_S: the identities of A, B, and S, respectively.
• PW_A, PW_B: the passwords securely shared by A with S and B With S, respectively.
• $E_{PW}(\cdot)$: a symmetric encryption scheme with a password PW.
• r_A, r_B: the random numbers chosen by A and B, respectively.
• p: a large prime.
• g: a generator of order p-1.
• R_A, R_B, R_S: the random exponents chosen by A, B, and S, respectively.
• N_A, N_B: $N_A=g^{RA}$ mod p and $N_B=g^{RB}$ mod p.
• $F_S(\cdot)$: the one-way trapdoor hash function (TDF) , where only S knows the trapdoor.
• $f_K(\cdot)$: the pseudo-random hash function (PRF) indexed by a key K.
• K_{AS}, K_{BS}: a one-time strong keys shared by A with S and B with S, respectively.
There are Six Steps in Chang–Chang's ECC-3PEKE protocol as follows.

Step 1 :A→B: {ID_A, ID_B, ID_S, E_{PWA}(NA), $F_S(r_A)$, f_{KAS} (N_A)}User A chooses a random integer number r_A and a random exponent R_A $._R Z_p$., and then computes $N_A=g^{RA}$ and $K_{AS}=N_A{}^{rA}$. Then, A encrypts N_A by using his/her password PWA like $E_{PWA}(N_A)$, and computes two hash values F_S (r_A) and f_{KAS} (N_A). Finally, A sends {ID_A, ID_B,

ID_S, $E_{PWA}(N_A)$, $F_S(r_A)$, $f_{KAS}(N_A)$} to B.

Step 2 :B→S: {ID_A, ID_B, ID_S, $E_{PWA}(N_A)$, $F_S(r_A)$, $f_{KAS}(N_A)$, $E_{PWB}(N_B)$, $F_S(r_B)$, $f_{KBS}(N_B)$}. User B chooses a random integer number r_B and a random exponent $R_B Zp$., and then computes $N_B=g^{RB}$ and $K_{BS}=NB^{rB}$. Then, B encrypts N_B by using his/her password PW_B like $E_{PWB}(N_B)$, and computes two hash values $F_S(r_B)$ and $f_{KBS}(N_B)$. Finally, B sends {ID_A, ID_B, ID_S, $E_{PWA}(N_A)$, $F_S(r_A)$, $f_{KAS}(N_A)$, $E_{PWB}(N_B)$, $F_S(r_B)$, $f_{KBS}(N_B)$} to S.

Step 3 : S→B: {N_B^{RS}, $f_{KAS}(ID_A, ID_B, K_{AS}, N_B^{RS})$, N_A^{RS}, $f_{KBS}(ID_A, ID_B, K_{BS}, N_A^{RS})$} Server S decrypts $E_{PWA}(NA)$ and $E_{PWB}(NB)$ by using PWA and PWB to get N_A and N_B, respectively. Then, S gets r_A and r_B from $F_S(r_A)$ and $F_S(r_B)$ by using a trapdoor, respectively. To authenticate A and B, S computes $K_{AS}=N_A^{rA}$ and $K_{BS}=N_B^{rB}$ and then verifies $f_{KAS}(N_A)$ and $f_{KBS}(N_B)$, respectively. If successful, S chooses a random exponent $R_S \cdot_R Z_p$. and then computes N_A^{RS} and N_B^{RS}, respectively. Finally, S computes two hash values $f_{KAS}(ID_A, ID_B, K_{AS}, N_B^{RS})$ and $f_{KBS}(ID_A, ID_B, K_{BS}, N_A^{RS})$, and sends {$N_B^{RS}$, $f_{KAS}(ID_A, ID_B, K_{AS}, N_B^{RS})$, N_A^{RS}, $f_{KBS}(ID_A, ID_B, K_{BS}, N_A^{RS})$} to B.

Step 4:B→A: {N_B^{RS}, $f_{KAS}(ID_A, ID_B, K_{AS}, N_B^{RS})$, $f_K(ID_B, K)$} By using $K_{BS}=N_B^{rB}$, B authenticates S by checking $f_{KBS}(ID_A, ID_B, K_{BS}, N_A^{RS})$. If successful, B computes the session key $K=(N_A^{RS})^{RB}=g^{RSRARB}$ and hash value $f_K(ID_B, K)$, and then sends {N_B^{RS}, $f_{KAS}(ID_A, ID_B, K_{AS}, N_B^{RS})$, $f_K(ID_B, K)$} to A.

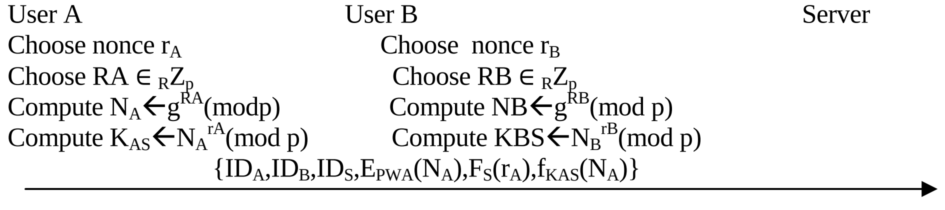
Step 5:A→B: {$f_K(ID_A, K)$} By using $K_{AS}=N_A^{rA}$, A authenticates S by checking $f_{KAS}(ID_A, ID_B, K_{AS}, N_B^{RS})$. If successful, A computes the session key $K=(N_B^{RS})^{RA}=g^{RSRARB}$, and
authenticates B by checking $f_K(ID_B, K)$. If the authentication is passed, A computes and sends $f_K(ID_A, K)$ to B.
Step 6: B authenticates A by checking $f_K(ID_A, K)$. If successful, B confirms A's knowledge of the session key $K=g^{RSRARB}$.

3. The enhanced protocol

In our protocol, we utilize the parallel message transmission mechanism to reduce one message transmission round in comparison with the literature .The motivation is taken from security enhanced protocol proposed by Lo-Yeh. [8]. The detailed procedures of our enhanced protocol are described as follows (Fig. 1)
Step1.A→S:ID_A, ID_B, ID_S, $E_{PWA}(N_A)$, $F_S(r_A)$, $f_{KAS}(N_A)$
B→.S:ID_A,ID_B,ID_S, $E_{PWB}(N_B)$, $F_S(r_B)$, $f_{KBS}(N_B)$.

Client A generates two random numbers R_A and r_A, and calculates $E_{PWA}(N_A)$, $F_S(r_A)$ and $f_{KAS}(N_A)$, where $K_{AS}=N_A^{rA}$ (mod p) and $N_A=g^{RA}$ (mod p). Next, A sends these three messages to S via his/her own private communication channel. Meanwhile, client B calculates $N_B=g^{RB}$ (mod p), $K_{BS}=N_B^{rB}$ (mod p), $E_{PWB}(N_B)$, $F_S(r_B)$ and $f_{KBS}(N_B)$ with two newly generated random numbers R_B and r_B. Then, B transmits $E_{PWB}(N_B)$, $F_S(r_B)$ and $f_{KBS}(N_B)$ to S via his/her own private communication channel. Both of A and B operate the above procedures at the same time.
Step 2. S →A: N_B^{RS}, $f_{KAS}(ID_A, ID_B, K_{AS}, N_B^{RS})$
S→ B: N_A^{RS}, $f_{KBS}(ID_A, ID_B, K_{BS}, N_A^{RS})$
Once receiving the message sent from A and B , S first utilizes a trapdoor to obtain r_A and r_B from $F_S(r_A)$ and $F_S(r_B)$, whether computed value $f_{KAS}(N_A)$(or $f_{KBS}(N_B)$) and received value $f_{KAS}(N_A)$ (or $f_{KBS}(N_B)$) are identical or not. If this verification holds, S continues the residual procedures of this protocol. Otherwise, S terminates this protocol at current session. Next, S computes N_B^{RS}, N_A^{RS}, and corresponding hashed credential $f_{KAS}(ID_A, ID_B, K_{AS}, N_B^{RS})$ and $f_{KAS}(ID_A, ID_B, K_{BS}, N_A^{RS})$. Finally, S sends these messages to A and B simultaneously.
Step 3. B →. A: $f_K(ID_B, K)$
Step 4. A →. B: $f_K(ID_A, K)$.

Shared Information: ID_A , ID_B , ID_S , p, g , E(.) , F_S(.) , f_K (.)
Information held by User A : PW_A
Information held by User B : PW_B
Information held by server S : PW_A , PW_B


User A
User B
Server
Choose nonce r_A
Choose nonce r_B
Choose $RA \in_R Z_p$
Choose $RB \in_R Z_p$
Compute $N_A \leftarrow g^{RA}$(modp)
Compute $NB \leftarrow g^{RB}$(mod p)
Compute $K_{AS} \leftarrow N_A^{rA}$(mod p)
Compute $KBS \leftarrow N_B^{rB}$(mod p)
{ID_A,ID_B,ID_S,$E_{PWA}(N_A)$,$F_S(r_A)$,$f_{KAS}(N_A)$}

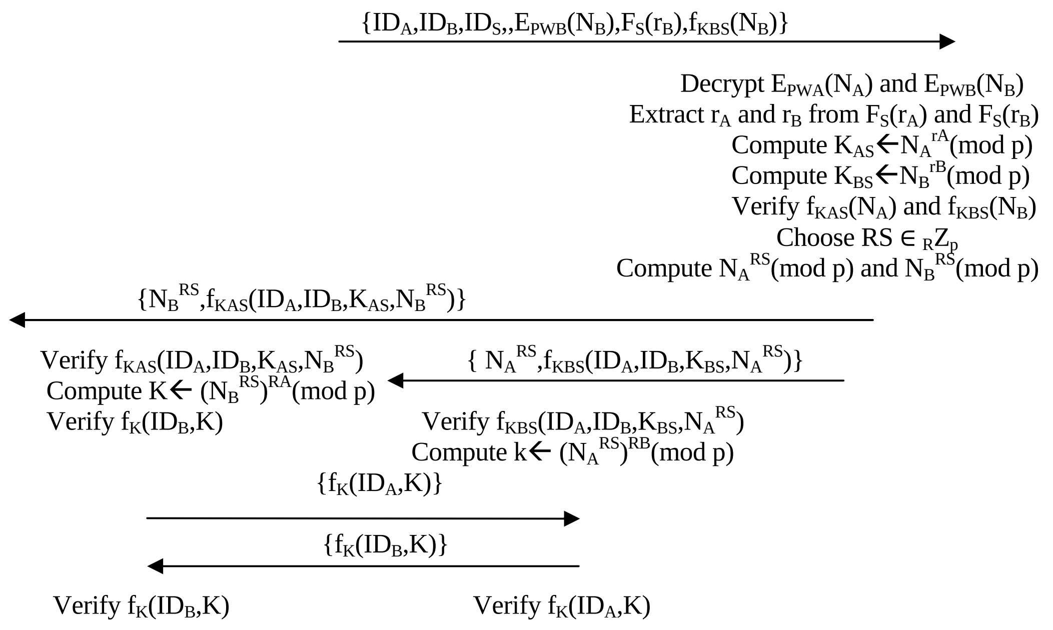

$\{ID_A,ID_B,ID_S,,E_{PWB}(N_B),F_S(r_B),f_{KBS}(N_B)\}$
Decrypt $E_{PWA}(N_A)$ and $E_{PWB}(N_B)$
Extract r_A and r_B from $F_S(r_A)$ and $F_S(r_B)$
Compute $K_{AS} \leftarrow N_A^{rA}$(mod p)
Compute $K_{BS} \leftarrow N_B^{rB}$(mod p)
Verify $f_{KAS}(N_A)$ and $f_{KBS}(N_B)$
Choose RS $\in_R Z_p$
Compute N_A^{RS}(mod p) and N_B^{RS}(mod p)
$\{N_B^{RS},f_{KAS}(ID_A,ID_B,K_{AS},N_B^{RS})\}$
Verify $f_{KAS}(ID_A,ID_B,K_{AS},N_B^{RS})$
Compute $K \leftarrow (N_B^{RS})^{RA}$(mod p)
Verify $f_K(ID_B,K)$
$\{ N_A^{RS},f_{KBS}(ID_A,ID_B,K_{BS},N_A^{RS})\}$
Verify $f_{KBS}(ID_A,ID_B,K_{BS},N_A^{RS})$
Compute $k \leftarrow (N_A^{RS})^{RB}$(mod p)
$\{f_K(ID_A,K)\}$
$\{f_K(ID_B,K)\}$
Verify $f_K(ID_B,K)$
Verify $f_K(ID_A,K)$

Fig 1. The Enhanced protocol

Upon obtaining the transmitted messages sent from S, B first verifies $f_{KBS}(ID_A, ID_B, K_{BS}, N_A^{RS})$ to authenticate S. If this verification is passed, B believes the received N_A^{RS} is valid and then computes the session key $K=(N_A^{RS})^{RB}$ (mod p) and $f_K(ID_B, K)$. Otherwise, B terminates this protocol. Finally, B sends the $f_K(ID_B, K)$ to A. Note that $f_K(ID_B, K)$ will be used by client A to verify the legality of client B and the established session key K. At the same time, A veri.es $f_{KAS}(ID_A, ID_B, K_{AS}, N_B^{RS})$ to authenticate S. If this verification does not hold, A terminates this protocol. Otherwise, A computes the session key $K=(N_B^{RS})^{RA}$ (mod p) and $f_K(ID_A, K)$. Finally, A sends the $f_K(ID_A, K)$ to B.

After A and B successfully examine the validation of the incoming messages $f_K(ID_B, K)$ and $f_K(ID_A, K)$, both of them can ensure that they actually share the secret session key $K=(N_B^{RS})^{RA}$ (mod p)$=(N_A^{RS})^{RB}$(mod p) at present. Otherwise, the protocol will be terminated.

4. Security and efficiency analysis

The following are the security requirements to be met by a password key exchange protocol.
Mutual authentication
Resistance to the password guessing attacks.
Transmission round and computation complexity.

Our enhanced protocol is satisfying the above requirements. The following section presents the brief report on the security analysis of our protocol with respect to requirements.

4.1. Mutual authentication

First, A & B uses the trapdoor function F_S to hide the random number r_A & r_B and P_A & P_B to encrypt N_A & N_B in step 1, as described in section 3 . since only s knows the trap door , P_A & P_B , only S can authenticate A/B after receiving the message sent in step 1.

Second, S sends. $\{N_B^{RS}, f_{KAS}(ID_A, ID_B, K_{AS}, N_B^{RS})\}$ to A,$\{ N_A^{RS}, f_{KBS}(ID_A, ID_B, K_{BS}, N_A^{RS}\}$ to B in step 2. This message can be used to authenticate ‘S’ as mentioned in step 2 in section 3.

Third, A and B derives key from N_B^{RS} and N_A^{RS} respectively as mentioned in step 2 in section 3. With the help of $f_K(ID_B, K)$, $f_K(ID_A, K)$ A & B can authenticate each other.

4.2. Resistance to the password guessing attacks

First, A directly sends the message $E_{PWA}(N_A)$, $F_S(r_A)$, $f_{KAS}(N_A)$ to server, B directly sends the message $E_{PWB}(N_B)$, $F_S(r_B)$, $f_{KBS}(N_B)$ to server hence there is no chance to guess the password of A by B as mentioned by Yoon and Yoo. In Chang and Chang protocol they are sending the message $E_{PWA}(N_A)$, $F_S(r_A)$, $f_{KAS}(N_A)$ through B , hence ‘B’ is trying to guess the password. In our proposed protocol there is no chance for B to guess A’ s password or A to guess B’ s password.

Second, if the message $E_{PWA}(N_A)$, $F_S(r_A)$, $f_{KAS}(N_A)$ is trapped by third party even then he cannot get any information from the obtained message since N_A is encrypted with password PW_A , r_A can be opened if the trapdoor is known , which is known only to the server. Without N_A and r_A , K_{AS} $=N_A^{rA}$(mod p) cannot be determined . Hence third party cannot mount any attack.

Third, even if the password PW_A is guessed by the attacker and N_A is retrieved by decrypting $E_{PWA}(N_A)$, then in order to get authenticate by the server he should know $K_{AS}=N_A^{rA}(\text{mod } p)$, which cannot be determined since it is computationally intractable mathematically hard problem known as DLP.

4.3. Transmission round and computation complexity

In 3PEKE research field, the development of an efficient protocol should take the number of transmission rounds (and steps) and the computation complexity into account. From the view point of the transmission round, our protocol adopts the parallel message transmission mechanism (i.e $A\rightarrow S$ and $B\rightarrow S$) . To achieve fewer transmission rounds than the protocols proposed by Chang and Chang(i.e. $A\rightarrow B\rightarrow S$).we can maintain the same security level even after deleting the XOR operations when compared with Lo-Yeh protocol.

5. Experimental results

The purpose of experimental results is to show the total running time needed for the expensive operations involved in various steps of the proposed protocol.

A data set is generated for problem (p) of size ranging from 128 bits-2048 bits. The more expensive steps in the protocol are: TDF, pseudorandom hash function, computing N_A, computing K_{AS}, symmetric encryption.

The following tables summarize the results of the computing time of the above expensive steps.

Table 1 represents the computing time required for encrypting the random numbers ranging from 128 bits to 2048 bits using RSA-Trap door function [10].

Table 1. Running time for TDF($F_S(r_A)$)

Bits	Running time in micro sec
128	1031
2048	350950

Table 2 represents the computing time needed to encrypt N_A ranging from 128 bits to 2048 bits with Data encryption standard i.e. symmetric encryption algorithm.

Table 2 Running time for $E_{PWA}(N_A)$

bits	Running time in micro-sec
64	351.4
2048	4778.5

Table 3 represents the computing time required for encrypting the random numbers ranging from 128 bits to 2048 bits using RSA-Trap door function with examples.

Table 3 Example for TDF

Bits	Random number (rA)	Running time in micro sec
128	340282366901131422834808524128598884352	1031
2048	323170060713110073007148766886699519604441026697154840321303454275246551 388678908931972014115229134636887179609218980194941195591504909210950881 523864482831206308773673009960917501977503896520501298848071354375082612 946199228570582655015123222838162870996287878647091415112738669228259962 835992893177845095553604919600714185604974185848259578797738153649670952 181159107797325129218196095958022711727009508322179060242184126269986104 746597115725933898242386283050902594622856838764562450822346538959295775 641909289938298903878924477917485758414127714946049106246475586434238305 84281562693227069197419997513229803716608	350950

Table 4 represents the computing time needed for calculating NA=g^{RA}(mod p) & K_{AS}=$N_A{}^{rA}$(mod p) for different size of problems(p)[9].

Table 4 Running time for calculating NA, KAS

Sl.no.	prime	Running time (**N_A**) in micro sec	Running time (**K_{AS}**) in micro sec
1	4324122104434447665362908248086967822904859	16641	276.3
20	2000000000000000000000000000000000000000000000000000000 0000000000000000000000000000000000000000000000000000000 0000000000000000000000000000000000000000000000000000000 0000000000000000000000000000000000000000000000000000000 0000000000000000000000000000000000000000000000000000000 0000000000000000000501759	119460	100990

6. Conclusion

We have proposed an enhanced password-key exchange protocol which is in-vulnerable to undetectable on-line password attacks, with reduced transmission rounds using Parallel Message Transmission protocol. Our proposed protocol is achieving better performance efficiency by requiring only four message transmission rounds and the performance is analyzed on a comprehensive set of experiments. The above results shows that the proposed protocol is secure, efficient and practical.

References

[1] W. Diffie and M. Hellman, "New Directions in cryptography", IEEE Transactions on Information theory, Vol 22 ,no. 6 , pp 644-54, (1976).

[2] P.Rajkumar and C.Manoharan," Parallel Message Transmission Technique for Password Key Exchange Protocol" European Journal of Scientific Research, Vol.77 No.4 (2012), pp.471-476.

[3] SM. Bellovin and M. Merrit, " Encrypted key exchange: password based protocols secure against dictionary attacks". IEEE sysmposium on re-search in security and privacy, IEEE Computer society press :72-84,(1992).

[4] M. Steiner and G. Tsudik, M. Waidner "Refinement and extention of encrypted key exchange", ACM Operating Systems Review, vol 29,no 3, pp 22-30, (1995).

[5] CL. Lin, HM. Sun, M. Steiner, T. Hwang " Three-party excrypted key exchange without server public Keys" IEEE Communication letters, vol 5, no.12,pp 497- 9 , (2001).

[6] CC. Chang and YF. Chang, "A novel three party encrypted key exchange protocol", Computer Standards and Interfaces, vol 26 , no 5, (pp 471-6),(2004).

[7] EJ. Yoon and KY. Yoo, "Improving the novel three-party encrypted key exchange protocol", Computer Standards and Interfaces, 30:309-314 , (2008).

[8] N.W.Lo,Kuo-Hui Yeh, "Cryptanalysis of two three-party Encrypted key exchange protocols", In press, computer standards and interfaces.

[9] R.Padmavathy and Chakravarthy Bhagavati, "Methods to solve Discrete Logarithm problem for Ephemeral Keys" ARTCOM, (2009).

[10] Fuw-Yi Yang, Improvement on a Trap door Hash Function,