Abstract—This paper endeavors to present a novel framework for the generic structure of a verifier-based password authenticated Three-Party Encrypted Key Exchange (3P-EKE) protocol which yields more efficient protocol than the ones knew before. A previous framework presented by Archana and Premchand is more secured against all types of attacks like password guessing, replay, pre-play, man-in-the-middle attack etc. But unfortunately, this protocol does not solve the problem of a server compromise. These proofs help as inspiration to search for another framework. The framework we offer produces more efficient 3P-EKE protocol, and, in addition, delivers perceptive clarification about the existing attacks that do not solve in the previous framework. Moreover, it allows direct change from a class of verge private-key encryption to a hybrid (symmetric & Asymmetric) one without significant overhead.


I. INTRODUCTION

A vital job of cryptography is to guard the confidentiality of messages transferred over the unsecured network. To provide security, messages can be encrypted by using a key (secret information) so that an intruder cannot decode the message. However, encoding the messages may not be only the solution, because an intruder may take the most active role as the network is open and reachable. We may need to change the key according to the session to establish a secure communication through the unsecured open network.

Consecutively, a password-authenticated two-party encrypted key exchange (2P-EKE) protocols are used to exchange a session key based on a low-entropy password. In this network, each party who wants to communicate needs to memorize a low-entropy password, which implies high maintenance of passwords. Due to this drawback, a password-authenticated three-party encrypted key exchange (3P-EKE) protocols has its demand as on date. According to Ding & Hoster [1], many of such 3P-EKE protocols suffer from any one of the three types of password guessing attacks. An intruder can guess the correct password by continuously trying until he succeeds, which is known as password guessing attack.

An Ideal password authenticated key exchange protocol should satisfy the security requirements like, Mutual Authentication, Resistant to password guessing attacks, Session Key (SK) security, Resistant to Trivial Attack, Resistant to Pre-play Attack, Resistant to Replay Attack, Resistant to Man-in-the-middle Attack, Server spoofing security, Perfect forward secrecy, backward secrecy, Known-Key Security, etc.

Based on the low entropy passwords shared between a user and a server, the password-authenticated key exchange (PAKE) protocols are classified into two types. They are:

A. Symmetric model

As the name implies, symmetric (identical) low entropy password shared between a Trusted Party (server) and a user in establishing a secure session key. If a Trusted Party is compromised, an intruder will get succeed in performing an attack on the legitimate user.

B. Asymmetric model

As the name implies, the password distribution is asymmetric in nature; i.e., a user will share the different knowledge (a verifier) about the low entropy password with the Trusted Party in establishing a secure session key. If a Trusted Party is compromised, the password table does not reveal the direct information about the password. In this way, a server spoofing is avoided.

Henceforth, the design of a novel framework which is smart in establishing a secure session key with less computational overhead; which prove to be secure against the attacks like password guessing, a server spoofing and also provide mutual authentication, Backward Secrecy, and Forward Secrecy is the need of the hour. This paper endeavors to propose a novel framework for establishing a secure session key based on an asymmetric model by using PCLA keys. PCLA is a new public key
cryptosystem based on the logarithmic approach proposed by Archana et al. in 2012[2].

Further, the rest of the paper is organized as follows: Related works is discussed in section II. In section III, we listed notations used in the proposed protocol. A framework for the proposed protocol is described in section IV. Security Analysis of the proposed protocol is done in section V. Finally, we made concluding remarks in section VI.

II. RELATED WORK

Diffie-Hellman (1976) [3] key exchange protocol is suffered from a man-in-the-middle attack due to the lack of an authentication. To assure a good access control, many applications require a robust client authentication. In such scenario, password-authenticated key exchange (PAKE) protocols have their own identity.

Bellovin and Merritt (1992)[4] have first proposed password-based authenticated encrypted key exchange protocol for the two-party network. But, due to the server compromise (server hacking: e.g., In 2012, more than million LinkedIn passwords are stolen) this protocol no longer proved to be secure. Hence, to eliminate such a problem he proposed an improvement over it known as Augmented EKE protocol (1993) [5], where a server instead of storing the actual passwords, it stores the verifiers of the passwords which prevents from a server compromise but it does not solve the problem of off-line dictionary attacks.

Subsequently, Gong et al.(1993)[6] proposed a three-party password-based authenticated key exchange protocol using a server’s public key, where the clients are given a risk to verify and keep the public key safely. Many improvements proposed by various researchers in terms of a security and computational efficiency [7, 8, 9, 10].

Abdalla et al. (2005) [11] proposed a ‘provable secure’ one-time password-based authentication and key exchange (OPKeyX) technology for grid computing; where a user changes the password from one session to another session to eliminate the problem of password sniffing. Lin et al. (2008)[12] proposed an efficient verifier-based password-authentication key exchange protocol by using elliptic curve cryptography. Unfortunately, Yang et al. (2011) [13] showed the flaws of Lin et al. protocol and proposed an improvement over the Efficient verifier-based password-authentication key exchange protocol via elliptic curves.

A Novel ECC-3PEKE protocol is proposed by Chang et al. (2004) [14], which proved to be practical, efficient and secure. However, Yoon et al. (2008) [15] notified an undetectable online password guessing attack and proposed an improvement over ECC-3PEKE protocol. Subsequently, PSRJ protocol has been proposed by Padmavathy et al. (2009) [16], which is also an improvement over ECC-3PEKE protocol. They claimed that the proposed protocol achieves better computational complexity and also secure against dictionary attacks.

Later Chang et al. (2009) [17] discussed why Yoon-Yoo’s Protocol is still insecure. R. Padmavathy (2010) [18] cryptanalyzed the PSRJ protocol and to overcome an attack she proposed an improvement over the existing one by using reduced modular exponentiation operations. Successively, an impersonation attack has been shown on the ECC-3PEKE protocol by Shirisha Tallapally (2010) [19]. Next, Archana et al. (2012) [20] showed detectable online password guessing the attack on PSRJ protocol.

Also, Kulkarni et al.(2007) [21] proposed a novel key exchange protocol based on verifier-based password authentication for three parties; where each client instead of storing the direct password itself it computes a one-way hash function on each password and stores the corresponding result in a server's password table. Subsequently, Shaban et al.(2008)[22] proposed an improvement over the Kulkarni et al.’s protocol in terms of computational complexity, by showing the reduced rounds from 7 to 4 without using symmetric encryption/decryption. But unfortunately, Archana et al. (2015) [23] cryptanalyzed the Shaban et al.’s protocol by showing the detectable online password guessing attack. Kulkarni et al.’s protocol are proved as secure against the dictionary attacks but it is computationally more expensive than our proposed protocol.

A previous framework presented by Archana et al. “in press” [24] is more secured against all types of attacks like password guessing, replay, pre-play, man-in-the-middle attack etc. But unfortunately, this protocol does not solve the problem of a server compromise. These proofs help as inspiration to search for another framework which eliminates the problems, may occur in the previous framework.

III. NOTATIONS

The list of notations along with their descriptions used in this paper is given in Table 1. In fact, Id_a, Id_b, Id_p are the identities of client-A, client-B, Trusted Party-TP respectively, which are known publicly.
Verifier-based Password Authenticated 3P-EKE Protocol using PCLA Keys

Table 1. List of Notations

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client-A/Client-B</td>
<td>Two parties who want to communicate with each other</td>
</tr>
<tr>
<td>Eve-E</td>
<td>An Attacker</td>
</tr>
<tr>
<td>Trusted Party TP</td>
<td>Trusted third party (a server)</td>
</tr>
<tr>
<td>IdA, IdB, Idtp</td>
<td>Identities of Client-A, Client-B and Trusted Party-TP</td>
</tr>
<tr>
<td>PwdA, PwdB</td>
<td>Passwords of Client-A &amp; Client-B used to generate Verifiers</td>
</tr>
<tr>
<td>VA, VB</td>
<td>Verifiers of Client-A &amp; Client-B</td>
</tr>
<tr>
<td>NewPwd</td>
<td>New Password for Backward Secrecy</td>
</tr>
<tr>
<td>Keyem, Keytp</td>
<td>PCLA Public &amp; Private keys of Trusted Party</td>
</tr>
<tr>
<td>EP(a)</td>
<td>A symmetric Encryption scheme with a password pwd</td>
</tr>
<tr>
<td>DP(a)</td>
<td>A symmetric Decryption scheme with a password pwd</td>
</tr>
<tr>
<td>EP_bw()</td>
<td>A Asymmetric Encryption scheme with a PCLA keys</td>
</tr>
<tr>
<td>DP_bw()</td>
<td>A Asymmetric Decryption scheme with a PCLA keys</td>
</tr>
<tr>
<td>p</td>
<td>A large prime number</td>
</tr>
<tr>
<td>G</td>
<td>A generator in GF(Group Field)</td>
</tr>
<tr>
<td>ra, rb</td>
<td>Random numbers chosen by Client-A, Client-B respectively</td>
</tr>
<tr>
<td>REa, REb, REtp</td>
<td>Random Exponents of Client-A, Client-B and Trusted party respectively</td>
</tr>
<tr>
<td>M_a, M_b</td>
<td>M_a=g^{ra}\mod p, M_b=g^{rb}\mod p</td>
</tr>
<tr>
<td>K_key, K_temp</td>
<td>K_key=M_a^{ra}\mod p, K_temp=M_b^{rb}\mod p are one time strong keys shared by Client-A &amp; Client-B with Trusted Party respectively</td>
</tr>
<tr>
<td>htp()</td>
<td>A one-way trapdoor function, where only trusted party knows the trapdoor tp</td>
</tr>
<tr>
<td>Tp()</td>
<td>A pseudo random hash function indexed by a key K</td>
</tr>
<tr>
<td>SK</td>
<td>Session Key</td>
</tr>
</tbody>
</table>

IV. FRAMEWORK FOR PROPOSED PROTOCOL

This section endeavors to propose a novel verifier-based password-authenticated 3P-EKE protocol using the PCLA keys. PCLA is a new public key cryptosystem based on the logarithmic approach proposed by Archana et al. More details about this algorithm (PCLA) are given in the reference paper [2]. The proposed protocol has been divided into three stages. They are:

- Initialization Stage
- Key Agreement Stage
- Key Computation Stage

A. Initialization Stage

At this 1st stage, the clients who want to communicate with each other have to register with Trusted Party in advance. The procedure for registration is as follows:

**Step 0:** Client-A and Client-B compute the verifiers V_A=H(Id_a, Idtp, Pwd_a) and V_B=H(Id_b, Idtp, Pwd_b) by choosing low random passwords Pwd_a and Pwd_b respectively. Now Client-A and Client-B sends the verifiers V_A & V_B to Trusted Party respectively through a secure channel.

B. Key Agreement Stage

In this 2nd stage of the protocol, the actual procedure for session key agreement begins. The protocol executes as per the following steps of the protocol.

**Step KA1:** Client-A generates two random numbers viz., RE_a, ra ∈ Zp, and computes K_key=M_a^{ra}\mod p where M_a=g^{ra}\mod p and sends the credentials {Id_a, Idtp, VA, EP_{Keyp}(Pwd_a), htp(Pwd_a\oplus M_a), F_{Knap}(M_a)} to Trusted Party.

Similarly, client-B generates two random numbers viz., RE_b, rb ∈ Zp, and computes K_key=M_b^{rb}\mod p where M_b=g^{rb}\mod p and sends the credentials {Id_b, Idtp, VA, EP_{Keyp}(Pwd_b), htp(Pwd_b\oplus M_b), F_{Knap}(M_b)} to Trusted Party.

**Step KA2:** Upon receiving the credentials from client-
A and client-B, Trusted Party decrypts EP_{Keypr}(Ppwd_a) & EP_{Keypr}(Ppwd_b) by using its PCLA private key Keypr, i.e., DP_{Keypr}(EP_{Keypr}(Ppwd_a)) & DP_{Keypr}(EP_{Keypr}(Ppwd_b)) and gets the low entropy passwords Ppwd_a, Ppwd_b respectively. Now Trusted Party computes H(Id_a, Id_p, Ppwd_a) & H(Id_a, Id_p, Ppwd_b) and retrieves the verifier V_A & V_B from its table and checks whether both the numbers are equal. If not, then it terminates the protocol at the current session.

If yes, then it implies that client-A & client-B is verified at first level and Trusted Party continues with the residual procedure of the protocol. Trusted party retrieves Ppwd_a®M_a & Ppwd_b®M_b from hpwd(Ppwd_a®M_a) & hpwd(Ppwd_b®M_b) by using trapdoor [25] ‘tp’ and compute’s M_a=(pwd_a®M_a)® Ppwd_a & M_b=(pwd_b®M_b)® Ppwd_b respectively. Now, it gets r_a=(M_a®r_a)®M_a & r_b=(M_b®r_b)®M_b from the credential Epwd_a(M_a®r_a) & Epwd_b(M_b®r_b) by decrypting with low entropy password Ppwd_a & Ppwd_b, i.e., Dpwd_a(Epwd_a(M_a®r_a)) & Dpwd_b(Epwd_b(M_b®r_b)). Next, Trusted Party performs the second level of verification by calculating F_{Keypr}(M_a) & F_{Keypr}(M_b) after the computation of K_{hpwd}=M_a®mod p & K_{hpwd}=M_b®mod p respectively. That is, it compares the computed value of F_{Keypr}(M_a) (or F_{Keypr}(M_b)) with the received value of F_{Keypr}(M_a) (or F_{Keypr}(M_b)). If not identical, it terminates the protocol at the current session.

If both are identical then verification of client-A & client-B is passed and it continues with the residual procedure of the protocol. Now, a Trusted Party chooses a random exponent RE_a∈Z_p to compute M_a®RE_a mod p and M_b®RE_a mod p and encrypts these values with its PCLA private key. Then Trusted Party sends these credentials to client-A and client-B simultaneously, i.e., Trusted Party → Client-A:{EP_{Keypr}(M_a®RE_a mod p)}, and Trusted Party → Client-B:{EP_{Keypr}(M_b®RE_a mod p)}.

The detail of key agreement stage is depicted in Fig. 2.

### C. Key Computation Stage

In order to compute a secure session key, this stage accomplishes to begin with the verification of the Trusted Party in a smart way.

**Step KC1:** Upon receiving the credentials from a Trusted Party, client-A decrypts EP_{Keypr}(M_a®RE_a mod p) by using the PCLA public key of Trusted Party i.e., DP_{Keypr}(EP_{Keypr}(M_a®RE_a mod p)) to get M_a®RE_a mod p. In this way, client-A authenticates the Trusted Party.

Similarly, client-B also authenticates the Trusted Party in the same way.

Now, client-A computes a mutual session key SK=(M_a®RE_a)®R_{ds} mod p=((g_{Rds}®RE_a)®R_{ds} mod p & F_{Sk}(Id_a,SK)) and sends it to client-B. Similarly, client-B computes a mutual session key SK=(M_b®RE_a)®R_{ds} mod p=((g_{Rds}®RE_a)®R_{ds} mod p & F_{Sk}(Id_b,SK)) and sends it to client-A.

i.e., Client-A → Client-B: {F_{Sk}(Id_a,SK)}), and Client-B → Client-A: {F_{Sk}(Id_b,SK)}.

**Step KC2:** Upon receiving the incoming credentials F_{Sk}(Id_a, SK) and F_{Sk}(Id_b, SK) from client-A and client-B respectively, they verify each other and can ensure that the mutual session key is SK=(M_a®RE_a)®R_{ds} (mod p) = (M_b®RE_a)®R_{ds} (mod p).

The detail of key computation stage is illustrated in Fig. 3.

### D. Password Change Mechanism

If any one of the client (say Client-A) suspects a ‘leak of information’, then it invokes a password change mechanism of our proposed protocol, which is helpful in providing backwater secrecy. The steps in this mechanism are as follows:

**Step PC1:** Client-A preserves the session key SK from the previous session.

**Step PC2:** Client-A, first encrypts the session key SK by using PCLA public key Keypr of Trusted Party. Next,
it computes \( H(\text{Id}_a, \text{Id}_p, \text{Pwd}_a) \oplus H(\text{Id}_a, \text{Id}_p, \text{NewPwd}_a) \oplus \text{SK} \) and sends the following credentials to Trusted Party to reset the new password verifier.

**i.e., Client-A \( \rightarrow \) Trusted Party**: \{\text{Id}_a, \text{Id}_p, \text{EP}_{\text{Keymp}}(\text{SK}), H(\text{Id}_a, \text{Id}_p, \text{Pwd}_a) \oplus H(\text{Id}_a, \text{Id}_p, \text{NewPwd}_a) \oplus \text{SK}, h_p(H(\text{Id}_a, \text{Id}_p, \text{NewPwd}_a))\}

**Step PC3:** Upon receiving the credentials from client-A, Trusted Party computes SK by decrypting \( \text{EP}_{\text{Keymp}}(\text{SK}) \) with the PCLA private key i.e., \( \text{DP}_{\text{Keypr}}(\text{EP}_{\text{Keymp}}(\text{SK})) \) and also retrieves the old verifier from the password table and computes the new password verifier as follows: \( H(\text{Id}_a, \text{Id}_p, \text{NewPwd}_a) = (H(\text{Id}_a, \text{Id}_p, \text{Pwd}_a) \oplus H(\text{Id}_a, \text{Id}_p, \text{NewPwd}_a) \oplus \text{SK}) \oplus H(\text{Id}_a, \text{Id}_p, \text{Pwd}_a) \oplus \text{SK}. \)

Now, Trusted party check for validation by extracting the new verifier from the received credential \( h_p(H(\text{Id}_a, \text{Id}_p, \text{NewPwd}_a)) \) by using a trapdoor ‘tp’; if computed \( H(\text{Id}_a, \text{Id}_p, \text{NewPwd}_a) \) is equal to the received one, Trusted Party updates the password table accordingly and sends ‘Accepted’ message to the client-A. Otherwise, Trusted Party rejects the request by sending the ‘Denied’ message to the client.

**i.e., Trusted Party**: Accepted/Denied.

---

**Fig.3. Key Computation Stage**

**Fig.4. Password Change Mechanism**
V. SECURITY ANALYSIS

The following security requirements are satisfied by the proposed protocol; which proves that the proposed protocol is not only efficient but also secure.

A. Resistant to an off-line dictionary attack

An attacker Eve-E may try to mount off-line password guessing attack to guess the password. She intercepts \{Id_a, Id_b, Ep_{prev}(Pwd_a), Ep_{pwda}(M_a \oplus r_a), h_{pwda}(Pwd_a \oplus M_a), F_{pwd}(M_a)\} and may guess a password to extract (M_a \oplus r_a), but it is impossible for her to get M_a until trapdoor ‘tp’ is known, which is known only to Trusted Party. This implies that she cannot verify the hash value F_{pwd}(M_a) which ascertains an offline password guessing attack on the proposed protocol is impossible.

Hence, the proposed protocol is resistant to off-line dictionary attack.

B. Resistant to server spoofing attack

Assume an intruder Eve-E succeeds in getting the password table of Trusted Party. Since only the verifier V of clients is stored in password table, Eve-E cannot mimic the client and compute SK.

Hence, the proposed protocol is resistant to a server spoofing attack.

C. Provides the mutual authentication

The proposed protocol promotes the mutual authentication and realizes the session key security to a great extent. The following are the scenarios where the mutual authentication can be proved.

• First Scenario: Client-A and Client-B use the public key Key_{pu} of Trusted Party to hide the corresponding passwords. Only Trusted Party knows the private key Key_{pr} to decrypt it. Hence, for an intruder Eve-E, it is not possible to get the passwords of client-A & client-B.

• Second Scenario: Client-A and Client-B use the trapdoor ‘tp’ to hide the random exponents RE_a in M_a & Pwd_a and RE_b in M_b & Pwd_b. Since only Trusted Party knows the trapdoor ‘tp’ and passwords Pwd_a & Pwd_b he can very well authenticate Client-A and Client-B after receiving the messages sent in step KA1 of the protocol.

• Third Scenario: Trusted Party sends \{EP_{Key_{pu}}(M_b \oplus r_b \mod p)\} to client-A & \{EP_{Key_{pu}}(M_b \oplus r_b \mod p)\} to client-B in step KA2 of the protocol. This message can be used to authenticate Trusted Party.

• Fourth Scenario: Client-A and Client-B derive a key from M_b^{R_{EP}} and M_b^{R_{EP}} respectively, as mentioned in step KC1 of the protocol. With the help of F_{SK}(Id_a, SK) & F_{SK}(Id_a, SK) both client-A and client-B can authenticate each other respectively as mentioned in step KC2 of the protocol.

Hence, the mutual authentication is provided by the proposed protocol.

D. Provides backward secrecy

The proposed protocol can provide backward secrecy, where compromise of Pwd_a will not lead to the compromise of NewPwd_a. Assume, a client-A suspects a ‘leak of information’ to Eve-E, then immediately client-A request to Trusted Party to change its password from Pwd_a to NewPwd_a. Let us assume, subsequently Eve-E intercepted the password change request, i.e., \{Id_a, Id_b, Ep_{prev}(SK), H(Id_a, Id_b, Pwd_a) \oplus H(Id_a, Id_b, NewPwd_a) \oplus SK, h_{pwda}(H(Id_a, Id_b, NewPwd_a))\} sent to Trusted Party by client-A. However, in this process Eve-E cannot compute SK by using Pwd_a, hence, he cannot compute NewPwd_a from the intercepted message \{Id_a, Id_b, Ep_{Key_{pu}}(SK), H(Id_a, Id_b, Pwd_a) \oplus H(Id_a, Id_b, NewPwd_a) \oplus SK, h_{pwda}(H(Id_a, Id_b, NewPwd_a))\}.

Hence, the backward secrecy is provided by the proposed protocol.

E. Provides the forward secrecy

The session key is computed as follows: SK= (M_a^{R_{EP}} \oplus RE_a) (mod p)=(M_a^{R_{EP}} \oplus RE_a) (mod p). If the Eve-E gets \{EP_{key_{pu}}(M_b^{R_{EP}} \mod p)\} or \{EP_{key_{pu}}(M_b^{R_{EP}} \mod p)\} then in order to obtain the session key, she should know the public key of Trusted Party and RE_b or RE_a. The session keys generated in different sessions are independent since RE_a and RE_b are randomly chosen by client-A and client-B respectively. This indicates that Eve-E cannot obtain previous session keys even if she obtains the session key used in this run.

Hence, the forward secrecy is provided by the proposed protocol.

VI. CONCLUSION

In this paper, we proposed a novel verifier-based password authenticated 3P-EKE protocol using PCLA keys, which provides perceptive justification about the existing attacks that do not solve in the previous framework. That is, our proposed protocol is proved to be secure against offline dictionary attacks and server spoofing attack. Further, we have also proved that our protocol provides mutual authentication, backward secrecy and also forward secrecy.

REFERENCES


Authors’ Profiles

Archana Raghuvamshi is presently working as an Assistant Professor in Dept. of CSE, UCOE, Adikavi Nannaya University, Rajahmundry. She is having 13+ year of teaching experience.

She received her Bachelor’s Degree BSc (M.S.Cs), Master’s Degrees M.C.A and M.Tech(CSE) from Osmania University, Hyderabad. She did course work in ADS and WMN in IITM (Indian Institute of Technology, Madras). She is persuing Ph.D. (CSE) in JNTUK, Kakinada. She published four research papers in IEEE Digital library and another six research papers in various peer reviewed International Journals. Her research interest includes Cryptography and Information Security, Security in Cloud Computing etc.

Ms. Archana Raghuvamshi is a,

1. Professional Member of ACM  
2. Member of Professional Body IAENG  
3. Member of IACSIT  
4. Associate Member of theIRED

Prof. Premchand Parvataneni is presently working as a professor in Department of Computer Science and Engineering at University College of Engineering, Osmania University, Hyderabad (Telangana). He received his Bachelor’s Degree B.Sc (Engg.) from RIT, Jamshedpur. He received his Master’s M.E (CE) from AU (Andhra University), Visakhapatnam. He received his Ph.D.(CSE) from AU. He has published more than 50 publications in various International Journals and Conference proceedings. His research interest includes Cryptography and Network Security, Image Processing, Software Engineering etc.

Prof.Premchand is having 40+ years of teaching experience.
in various Universities. He was as a Director in AICTE, New Delhi. And also, he has been held for the various positions like Head, Chairman of BOS, Additional Controller of Examinations in the Professional wing, Osmania University, Hyderabad.

**How to cite this paper:** Archana Raghuvamshi, Premchand Parvataneni,"Verifier-based Password Authenticated 3P-EKE Protocol using PCLA keys", International Journal of Computer Network and Information Security(IJCNIS), Vol.8, No.6, pp.59-66, 2016.DOI: 10.5815/ijcnis.2016.06.07