International Journal of Advanced Research Foundation
Website: www.ijarf.com, Volume 2, Issue 6, June 2015)

Design and Analysis of Key Encryption Schemes to Secure Two Servers

1 A. Mallareddy, 2 Yashasree J, 3 V Anusha
1 Research Scholar(JNTUH), Department of Computer Science & Engineering,
Professor &HOD(CSE) Sri Indu Institute of Engineering & Technology,
Sheriguda(V), Ibrahimpatnam(M), RR Dist – 501510.
2 Associate Professor, Department of Computer Science & Engineering,
Sri Indu Institute of Engineering & Technology, Sheriguda(V), Ibrahimpatnam(M), RR Dist – 501510.
3 M.Tech (CS) , Department of Computer Science & Engineering,
Sri Indu Institute of Engineering & Technology, Sheriguda(V), Ibrahimpatnam(M), RR Dist – 501510.

E-mail: ***1 mallareddyadudhodla@gmail.com*** ***2 yashasree123@gmail.com*** ***3.*** *anushareddy.v30@gmail.com*

***Abstract*: Password-authenticated key exchange (PAKE) is wherever a consumer and a server, UN agency share a positive identification, manifest one another and meantime establish a cryptanalytic key by exchange of messages. during this setting, all the passwords necessary to manifest shoppers square measure keep in an exceedingly single server. If the server is compromised, due to, for instance, hacking or perhaps corporate executive attack, passwords keep within the server is all disclosed. during this project, we tend to take into account a state of affairs wherever 2 servers work to manifest a consumer and if one server is compromised, the assailant still cannot fake to be the consumer with the knowledge from the compromised server. Current solutions for 2-server PAKE square measure either parallel within the sense that two peer servers equally contribute to the authentication or uneven within the sense that one server authenticates the consumer with the assistance of another server. This project presents a parallel answer for two-server PAKE, wherever the consumer will establish totally different cryptanalytic keys with the 2 servers, severally.**

***Kew* Words:Deffie Hellman protocol, ElGamal coding, 2 passwords solely.**

1. INTRODUCTION

Hacking and intrusion square measure the most important issues in secure transmission of messages between server and consumer, which might be simply done through comprimisation of server. One different thanks to break the safety is by guesswork the passwords additionally the hash values is guessed that the best means is for assailant that is often used or followed by most. Preliminary studies cowl the idea of on-line and offline attacks ,majorly they're sorted below wordbook attacks. Here offline attacks square measure quicker in speed of guesswork the positive identification. Offline attacks square measure the passive attacks during which assailant attempt to guess the positive identification from the spoken communication or communication created between 2 parties and guess the attacks from the wordbook. on-line attacks the assailant guess the positive identification through wordbook and tries to login. on-line tries created to guess the positive identification is additionally known as as active attacks .These each kinds of attack ought to be avoided in consumer server communication. the most aim in server and consumer design is that client's passwords square measure maintained to server solely. once that server gets hacked then security within the transmission of message are reduced.. The design that is best suited is that the public key Infrastructure and also the model used is PKI model. PKI model is employed for positive identification protection. It allows users to firmly non-public|and personal}ly exchange knowledge with the employment of public and private key try. The digital certificate .It uses the general public key cryptography. The positive identification solely genuine key exchange uses short positive identification of user. The second model that is positive identification solely model, wherever positive identification is that the secret key for coding of random designated numbers that is taken into account as a benchmark for the cryptanalytic assumption. Here the most idea is that the user needs to bear in mind the positive identification each communication done is predicated on vital issue that's positive identification because it doesn't need any auxiliary storage.

Humans usually opt for "weak", low-entropy passwords, whereas normal authentication protocols assume the employment of cryptanalytic (i.e., high-entropy) secrets. sadly, protocols designed associated proved secure within the latter setting square measure typically insecure within the former context as a result of these protocols aren't proof against off-line wordbook attacks during which an eavesdropping someone derives data concerning the positive identification from discovered transcripts of login sessions. In recent years, a lot of attention has targeted on planning password-based genuine key-exchange protocols proof against such attacks. (We remark that on-line wordbook attacks — during which associate someone merely tries to login repeatedly, making an attempt every potential positive identification — can not be prevented by cryptanalytic means that however is controlled

International Journal of Advanced Research Foundation
Website: www.ijarf.com, Volume 2, Issue 6, June 2015)

mistreatment different ways outside the scope of this work.) means that of protective against off-line wordbook attacks in an exceedingly single-server setting were initial urged by Gong et al. [21] in an exceedingly "hybrid", PKI-based model wherever users square measure needed to store the server's public key additionally to a positive identification. Bellovin and Merritt [5] were the primary to counsel protocols for password-only genuine key exchange (PAKE), wherever users square measure needed to store solely a brief positive identification. These initial works (and others [6, 23, 28, 34]) were comparatively informal and didn't give definitions or proofs of security.

1. DEFINITIONS AND PRELIMINARIES

We assume the reader is accustomed to the model of Bellare et al. [1] (building on [3, 4]) for passwordbased key exchange within the single-server case. Here, we tend to generalize their model and gift formal definitions for two-server protocols. whereas the model given here is basically adore the model planned by MacKenzie et al. [31] (with the most distinction being that we tend to don't assume a PKI), we are able to change matters slightly since we tend to target the two-server setting completely. For convenience we tend to initial describe the model for the case of a "passive" opponent corrupting one among the servers, so discuss in brief the modifications required to handle Associate in Nursing "active" opponent. (As mentioned below, in each the "passive" and "active" cases the opponent is unengaged to interfere with all communication between the shopper and also the servers. These cases solely take issue within the power of the opponent to manage the actions of the corrupted servers: specifically, a "passive" opponent is unable to manage the actions of corrupted servers, whereas Associate in Nursing "active" opponent will.) we tend to initial gift a general summary of the system. For simplicity, we tend to assume that each shopper C within the system shares its secret pw with specifically 2 servers A and B. during this case area unit saying} that servers A and B are related to C. (A single server could also be related to multiple shoppers.) additionally to holding secret shares, these servers may additionally be provisioned with whimsical different info (that needn't be hold on by C). Any such info is provisioned by some incorrupt, central mechanism (a computer user, say) at the commencement of the protocol. This doesn't represent a restriction in apply, since the servers should be provisioned with correct secret shares anyway, then any extra info are often provided to the servers at that point. moreover, the servers don't have any restriction — because the shopper will — on the quantity of three info they will store. Associate in Nursing (honest) execution of a protocol between shopper C and associated servers A and B ought to end in the shopper holding 2 (independent) session keys skC,A,skC,B, and servers A and B holding skA,C and skB,C, severally, with skC,A = skA,C and skC,B = skB,C.

In a single-server PAKE protocol, if the server is compromised, user passwords hold on within the server square measure all disclosed. to handle this issue, in 2000, Ford and Kaliski planned the primary threshold PAKE protocol within the PKIbased model, within which n servers join forces to manifest a shopper. afterward, in 2001, Joblon removed the necessity for PKI and instructed a protocol with the similar property within the password-only model. each the edge PAKE protocols weren't shown to be secure formally. In 2002, MacKenzie et al. gave a protocol within the PKI-based setting, which needs solely t out of n servers to join forces to manifest a shopper and is secure as long as fewer servers square measure compromised. They were the primary to produce a proper security proof for his or her threshold PAKE protocol within the random oracle model. In 2003, Di Raimondo and Gennaro planned a protocol within the password-only setting, which needs but 1/3 of the servers to be compromised, with a proper security proof within the customary model.

In this paper, we tend to propose a replacement bilaterally symmetrical answer for two-server PAKE. all told existing two-server PAKE protocols, 2 servers square measure provided random secret shares pw1 and pw2 subject to pw1 and pw2. In our protocol, we offer one server S1 with Associate in Nursing secret writing of the secret Associate in Nursingd another server S2 with an secret writing of the secret wherever pk1 and pk2 square measure the secret writing keys of S1 and S2, severally. additionally, 2 servers square measure provided random secret shares b1 and b2, wherever H could be a hash perform. just like the secret pw is secret unless the 2 servers conspire. though we tend to use the conception of public key cryptosystem, our protocol follows the password-only model. The secret writing and coding key pairs for the 2 servers square measure generated by the shopper and delivered to the servers through totally different secure channels throughout the shopper registration, because the shopper in any 2-server PAKE protocol sends two halves of the secret to the 2 servers on the QT, severally. In fact, a server shouldn't apprehend the secret writing key of another server and is restricted to control on the secret writing of the secret on the premise of the homomorphic properties

2. A PROTOCOL SECURE AGAINST PASSIVE ADVERSARIES

Description of the Protocol we tend to assume the reader is accustomed to the decisional Diffie-Hellman (DDH) assumption [15], strong5 one-time signature schemes, and also the Cramer-Shoup secret writing theme [14] with labels. A highlevel depiction of the protocol is given in Figures 1–3, and a a lot of careful description, still as some informal discussion regarding the protocol, follows.

International Journal of Advanced Research Foundation
Website: www.ijarf.com, Volume 2, Issue 6, June 2015)

Initialization. throughout the formatting section, public parameters (i.e., a standard reference string) square measure generated and created offered to all or any parties. For security parameter k, the general public parameters for our protocol contain a gaggle G (written multiplicatively) having such as prime order letter of the alphabet with |q| = k; we tend to assume the hardness of the DDH downside in G. to boot, the parameters embody random generators g1, g2, g3, h, c, d ∈ G \ and a hash perform H : ∗ → Zq chosen arbitrarily from a collision-resistant hash family. As a part of the formatting, every server S is provisioned with Associate in Nursing El Gamal public-/secret-key combine (pkS, skS), wherever pkS = g skS one . If A and B square measure related to constant shopper C, then A (resp., B) is given pkB (resp., pkA). we tend to stress that, in distinction to the PKI-based model, the shopper isn't assumed or needed to grasp the general public keys of any of the servers

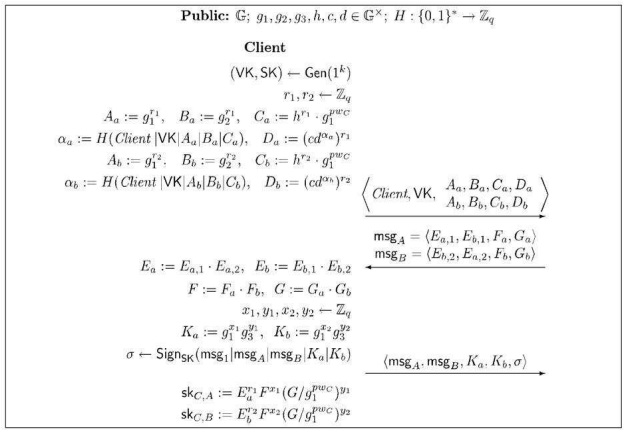

Figure 1: An execution of the protocol from the client's point of view.

Given a message m ∈ G and a public key (i.e., group element) pk, we let M ← ElGpk(m) denote the act of choosing a random r ∈ Zq and setting M = (g r 1 , pkrm). We let M[1] refer to the first component of this ciphertext, and let M[2] refer to the second. Note that if sk is the corresponding secret key (i.e., pk = g sk 1), then we have m = M[2] M[1]sk . Passwords and password shares are provisioned in the following way: a password pwC is chosen randomly for each client C and we assume that this password can be mapped in a one-to-one fashion to Zq. If A and B are the servers associated with a client C, then password shares pwA,C , pwB,C ∈ Zq are chosen uniformly at random subject to pwA,C +pwB,C = pwC mod q, with pwA,C given to server A and pwB,C given to server B. In addition, both A and B are given ComA,C, Com0 A,C, ComB,C, and Com0 B,C, where: ComA,C def = ElGg3 (g pwA,C 1) = ¡ g ra 1 , gra 3 g pwA,C 1 ¢ Com0 A,C def = ElGpkA (g pwA,C 1) ComB,C def = ElGg3 (g pwB,C 1) = ¡ g rb 1 , g rb 3 g pwB,C 1 ¢ Com0 B,C def = ElGpkB (g pwB,C 1). (Note that the keys for these El Gamal encryptions differ.) Server A (resp., server B) is additionally given the randomness ra (resp., rb) used to construct ComA,C (resp., ComB,C). Protocol execution. At a high level one can view our protocol as two executions of the KOY*.

Protocol, one between the client and server A (using server B to assist with the authentication), and one between the client and server B (using server A to assist with the authentication). When a client with password pwC wants to initiate an execution of the protocol, this client computes Cramer-Shoup "encryptions" of pwC for each of the two servers. In more detail (cf. Figure 1), the client begins by running a key-generation algorithm for a one-time signature scheme, yielding verification key VK and signing key SK. The client next chooses random r1 ∈ Zq and computes Aa = g r1 1 , Ba = g r1 2 , and Ca = h r1 · g pwC 1 . The client then computes αa = H(Client|VK|Aa|Ba|Ca) and sets D = (cdαa) r1 . This exact procedure is carried out a second time using an independent random value r2 ∈ Zq. The client sends msg1 def = hClient, VK, Aa, Ba, Ca, Da, Ab, Bb, Cb, Dbi to each server as the first message of the protocol. Note that this corresponds to two independent "encryptions" of pwC using the label Client |VK. The servers act symmetrically, so for simplicity we describe the actions of server A (cf. Figure 2).Upon receiving msg1 , server A sends "shares" of (1) two values of the form g x 1 g y 2h z (cdα) w (for α ∈ {αa, αb}), one for server A and one for server B, and (2) an El Gamal encryption of pwC . In more detail, server A chooses random xa, ya, za, wa ∈ Zq and computes Ea,1 = g xa 1 g ya 2 h za(cdαa) wa. It also chooses random x 0 a , y0 a , z0 a , w0 a ∈ Zq and computes Eb,1 = g x 0 a 1 g y 0 a 2 h z 0 a(cdαb) w0 a. Finally, it sets (Fa, Ga) equal to ComA,C (which, recall, is an El Gamal encryption of g pwA,C 1 using "public key" g3 and randomness ra). It sends the message msgA def = hEa,1, Eb,1, Fa, Gai to the client. After receiving a second-round message from each server, the client combines the values thus received by multiplying them component-wise to obtain hEa, Eb, F, Gi (cf. Figure 1). Note that (1) neither server knows the representation of Ea (resp., Eb) with respect to the basis g1, g2, h,(cdαa) (resp., g1, g2, h,(cdαb)), and (2) the values (F, G) form an El Gamal encryption of the client's password pwC (with respect to public key g3). The client next chooses random values x1, y1, x2, y2 ∈ Zq, computes Ka = g x1 1 g y1 3 and Kb = g x2 1 g y2 3 , and computes a signature σ on msg1 |msgA|msgB|Ka|Kb using the secret key SK that it had previously generated. It sends msg3 def = hmsgA, msgB, Ka, Kb, σi to each server as the final message of the protocol. (Of course, msgA need not be sent to A and similarly for msgB. Furthermore, it would suffice for the client to sign and send hashes of these messages.) Finally, the client computes session keys skC,A := E r1 a F x1 (G/gpwC 1) y1 skC,B := E r2 b F x2 (G/gpwC 1) y2 . Upon receiving hmsgA, msgB, Ka, Kb, σi from the client, server A verifies that the (Fb, Gb) component of msgB is equal to ComB,C (recall that A stores ComB,C), and that σ is a valid signature on msg1 |msgA|msgB|Ka|Kb

International Journal of Advanced Research Foundation
Website: www.ijarf.com, Volume 2, Issue 6, June 2015)

with respect to VK. If verification fails, the server terminates the execution. Otherwise, servers A and B jointly execute the Compute protocol (cf. Figure 3 and described next) in order to compute their session keys. Before describing the Compute protocol, we introduce notation for manipulation of El Gamal ciphertexts. If M, M0 are two El Gamal ciphertexts (encrypted with respect to the same public key pk), then we let M × M0 denote (M[1] · M0 [1], M[2] · M0 [2]). Note that if M is an encryption of m and M0 is an encryption of m0 , then M × M0 is an encryption of m · m0 . For x ∈ Zq, we let Mx denote the ciphertext (M[1]x , M[2]x). Here, the resulting ciphertext is an encryption of mx .

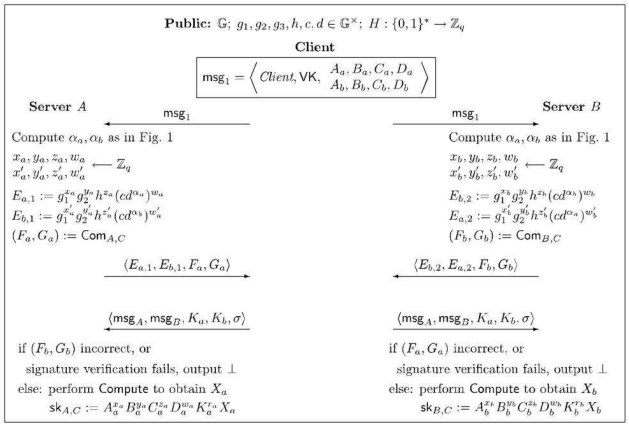

Figure 2: Execution of the protocol from the servers' points of view; see text for details. The Compute protocol is given in Figure 3.

With this in mind, we now describe the Compute protocol. Since the protocol is symmetric, we simply describe it from the point of view of server A. This server sets M1 to be an El Gamal encryption (with respect to pkA) of g −za 1 . It then sends M1 to server B, who computes M0 2 ← M pwB,C 1 × ¡ Com0 A,C¢−z 0 b × ElGpkA (g −z 0 b pwB,C 1 A x 0 b a B y 0 b a C z 0 b a D w0 b a Krb a) and sends this value back to server A (recall that rb is the randomness used to construct ComB,C). Finally, server A decrypts M0 2 and multiplies the result by g −za·pwA,C 1 to obtain Xa. Note that Xa = g −(za+z 0 b)·pwC 1 · ³ A x 0 b a B y 0 b a C z 0 b a D w0 b a Krb a ´ , (1) using the fact that pwC = pwA,C + pwB,C mod q. In addition to the above, in the first step of the protocol the servers exchange the messages received from the client thus far; each server then verifies that these messages match their own view (and terminates if not). Although omitted in the above description, we assume that the client and servers always verify that incoming messages are well-formed, and in particular that all appropriate components of the various messages indeed lie in G (we assume that membership in G can be efficiently verified).

3. HANDLING ACTIVE ADVERSARIES

Here, we describe the necessary changes to the protocol in order to handle active adversaries. We then sketch the appropriate modifications to the proof given in the previous section.

Overview of Changes to the Protocol At a high level, the changes we make can be summarized as follows: Proofs of correctness. We require servers to give proofs of correctness for their actions during the Compute protocol. We stress that we use only the fact that these are proofs (and not proofs of knowledge) and therefore we do not require any rewinding in our proof of security. This is crucial, as it enables us to handle concurrent executions of the protocol. Nevertheless, as part of the proofs of correctness we will have the servers encrypt certain values with respect to (additional) per-server public keys provisioned during protocol initialization. This will, in fact, enable extraction of certain values from the adversary during the security proof. Commitments to password shares. The protocol as described in the previous section already assumes that each server is provisioned with appropriate El Gamal encryptions of the password share of the other server. We will use these encrypted values (along with the proofs of correctness discussed earlier) to "force" a corrupted server to use the correct password share in its computations. Simulating proofs for non-corrupted servers. During the course of the proof of security it will be necessary for non-corrupted servers to deviate from the prescribed actions of the protocol, yet these servers must give "valid" proofs of correctness to corrupted servers. We cannot use "standard" zero-knowledge proofs in our setting, since (1) this would require rewinding which we explicitly want to avoid, and (2) potential malleability issues arise due to the fact that a corrupted server may be giving its proof of correctness at the same time a non-corrupted server is giving such a proof (this is so even if we force sequential executions of the proofs of correctness within any particular instance, since multiple instances may be simultaneously active).

To enable simulatability in a concurrent setting we rely on techniques of MacKenzie [29] described in greater detail below. 5.2 Detailed Description of Changes to the Protocol We first discuss the necessary modifications to the initialization phase. In addition to the values already discussed in Section 4.1: (1) each server S is given a random triple tripleS = (US,1, US,2, US,3) of elements chosen uniformly at random from G. Furthermore, (2) if servers A and B are associated with the same client C, then B is given tripleA and A is given tripleB. We next describe the necessary changes to the protocol itself. In what follows, we use witnessindistinguishable Σ-protocols (with negligible soundness error7) [12] of various predicates and it will be useful to develop some notation. If Ψ represents a predicate (defined over some public values), we let Σ[Ψ] denote a Σ-protocol for this predicate. If Ψ1, Ψ2 are two predicates, then we let Σ[Ψ1 ∨ Ψ2] denote a Σ-protocol for the "or" of these

International Journal of Advanced Research Foundation
Website: www.ijarf.com, Volume 2, Issue 6, June 2015)

predicates. Given Σ-protocols for Ψ1 and Ψ2, there are standard techniques to combine these so as to obtain a Σ-protocol for Ψ1 ∨ Ψ2 [13]. We define the predicate DDHS, for any server S with tripleS = (U1, U2, U3), as follows: DDHS(U1, U2, U3) def = [∃x, y s.t. U1 = g x 1 ∧ U2 = g y 1 ∧ U3 = g xy 1] ; i.e., DDHS denotes the predicate asserting that tripleS is a Diffie-Hellman triple. The only change in the protocol is the Compute component, which is modified in the following ways (we describe the changes from the point of view of server A, but they are applied symmetrically.

4. RESULTS

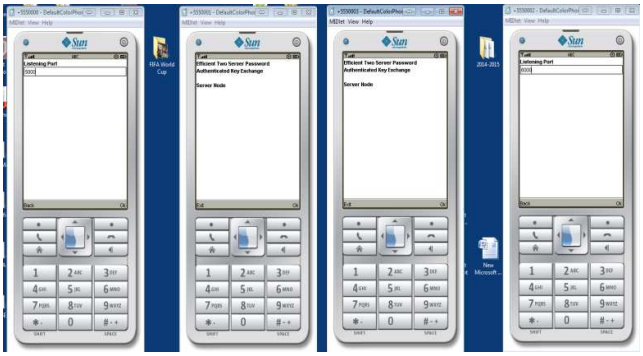

Figure 3: The above screen shows mobile node launched to listen on ports 5000 and 6000 respectively and server node initialized. On the screen we cannot use mouse so, we can use bellow the screen buttons.

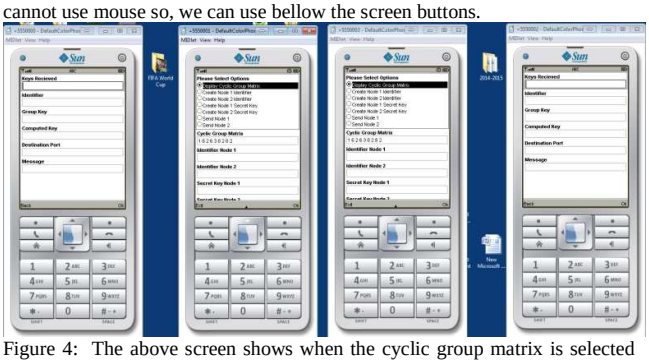

Figure 4: The above screen shows when the cyclic group matrix is selected then the two server nodes displaying cyclic group keys. In the above screen we are seeing the matrix of cyclic group is 162638282.

Figure 5 : The above screen shows that now we are creating a node 1 secret key. Then the two server nodes computing node 1 secret keys. The two sever nodes have their own keys. These two keys are different.

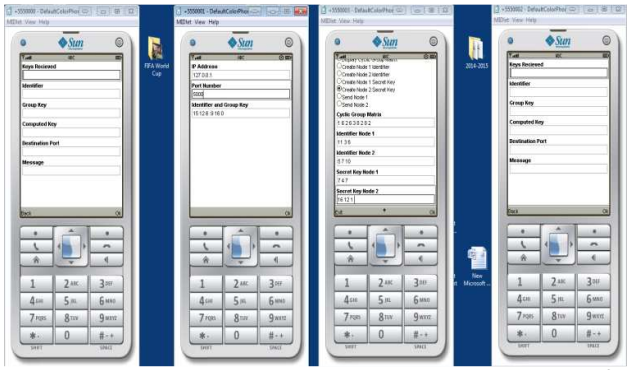

Figure 6 : The above screen shows that IP Address, Port Number, Identifier and Group Key. We have enter IP Address, Port Number, Identifier and Group Key. First server node sending identifier and group keys to node 1.

Figure 7: The above screen shows first mobile node received identifier and group keys and computed its own key from second server.

Figure 8: The above screen shows second mobile node received identifier and group keys and computed its own key we observe that both nodes will have same keys computed.

International Journal of Advanced Research Foundation
Website: www.ijarf.com, Volume 2, Issue 6, June 2015)

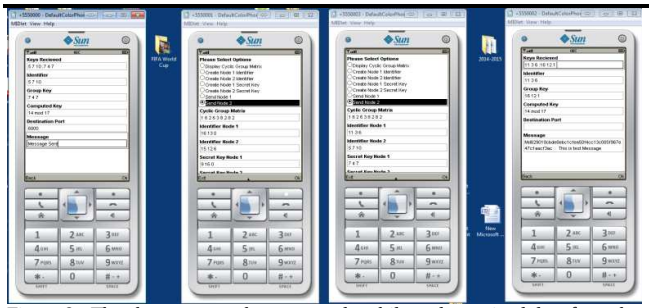

Figure 9: The above screen shows second mobile node received data from the first mobile node in secured format shown as hexadecimal and interprets the correct message.

9. CONCLUSION

In this paper, we've bestowed a cruciate protocol for two-server password-only authentication and key exchange. Security analysis has shown that our protocol is secure against passive and active attacks just in case that one in all the 2 servers is compromised. Performance analysis has shown that our protocol is a lot of economical than existing cruciate and uneven two-server PAKE protocols.

REFERENCES

[1]. Xun Yi, San Ling, and Huaxiong Wang Efficient Two-Server Password-OnlyAuthenticated Key Exchange. IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 24, NO. 9, SEPTEMBER 2013.

[2]. M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated Key Exchange Secure Against Dictionary Attacks. Adv. in Cryptology — Eurocrypt 2000, LNCS vol. 1807, Springer-Verlag, pp. 139–155, 2000.

[3]. M. Bellare and P. Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. Proc. 1st ACM Conference on Computer and Communications Security, ACM, pp. 62–73, 1993.

[4]. M. Bellare and P. Rogaway. Entity Authentication and Key Distribution. Adv. in Cryptology — Crypto 1993, LNCS vol. 773, Springer-Verlag, pp. 232–249, 1994.

[5]. M. Bellare and P. Rogaway. Provably Secure Session Key Distribution: the Three Party Case. 27th ACM Symposium on Theory of Computing (STOC), ACM, pp. 57–66, 1995.

[6]. S.M. Bellovin and M. Merritt. Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks. IEEE Symposium on Research in Security and Privacy, IEEE, pp. 72–84, 1992. 25

[7]. S.M. Bellovin and M. Merritt. Augmented Encrypted Key Exchange: a Password-Based Protocol Secure Against Dictionary Attacks and Password File Compromise. 1st ACM Conf. on Computer and Comm. Security, ACM, pp. 244–250, 1993.

[8]. M. Boyarsky. Public-Key Cryptography and Password Protocols: The Multi-User Case. 7th Ann. Conf. on Computer and Comm. Security, ACM, pp. 63–72, 1999.

[9]. V. Boyko, P. MacKenzie, and S. Patel. Provably Secure Password-Authenticated Key Exchange Using Diffie-Hellman. Adv. in Cryptology — Eurocrypt 2000, LNCS vol. 1807, Springer-Verlag, pp. 156–171, 2000.

[10]. J. Brainard, A. Juels, B. Kaliski, and M. Szydlo. Nightingale: A New Two-Server Approach for Authentication with Short Secrets. 12th USENIX Security Symp., pp. 201–213, 2003.

[11]. R. Canetti, O. Goldreich, and S. Halevi. The Random Oracle Methodology, Revisited. J. ACM 51(4): 557–594, 2004.

[12]. R. Canetti, S. Halevi, J. Katz, Y. Lindell, and P. MacKenzie. Universally-Composable Password Authenticated Key Exchange. Adv. in Cryptology — Eurocrypt 2005, LNCS vol. 3494, Springer-Verlag, pp. 404–421, 2005.

[13]. R. Cramer. Modular Design of Secure Yet Practical Cryptographic Protocols. PhD Thesis, CWI and University of Amsterdam, 1996.

[14]. R. Cramer, I. Damg˚ard, and B. Schoenmakers. Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. Adv. in Cryptology — Crypto 1994, LNCS vol. 839, Springer-Verlag, pp. 174–187, 1994.

[15]. R. Cramer and V. Shoup. Design and Analysis of Practical Public-Key Encryption Schemes Secure against Adaptive Chosen Ciphertext Attack. SIAM J. Computing 33(1): 167–226, 2003.