ICIC Express Letters
Volume **13**, Number **4**, April **2019**

 ISSN 1881-803X
pp. **293–301**

CRYPTANALYSIS OF AN EFFICIENT PROTOCOL FOR AUTHENTICATED KEY AGREEMENT

Zeyad Mohammad

Department of Computer Science
Al-Zaytoonah University of Jordan
P.O. Box 130, Amman 11733, Jordan
1996@zuj.edu.jo; dasouqi@yahoo.com

Received October 2018; accepted December 2018

Abstract. *In NRSC28, Elkamchouchi and Abu Elkair proposed an efficient protocol for authenticated key agreement, and the authors claimed that the proposed protocol is secure against known attacks. However, this letter shows that Elkamchouchi and Abu Elkair protocol cannot withstand the two types of key compromise impersonation (KCI) attacks namely, general-KCI, and static-KCI. In addition, we show the man-in-the-middle attack and known key security attack on Elkamchouchi and Abu Elkair protocol, and also the protocol cannot provide perfect forward secrecy property. Furthermore, we use the Scyther tool as an automated verification method to demonstrate the security flaws in Elkamchouchi and Abu Elkair protocol.*

Keywords: Key agreement, Key compromise impersonation, Known key security, Man-in-the-middle attack, Perfect forward secrecy, Scyther tool

1. **Introduction.** Diffie-Hellman key exchange (DHKE) protocol is based on a public key cryptography concept which is a fundamental block for distributing a common secret key over an insecure communication model [1]. The common secret key can be used for securing transmitted data over an insecure network model. There are two versions of DHKE protocol namely, the ephemeral and static. The ephemeral DHKE protocol is vulnerable to the man-in-the-middle (MITM) attack because the exchanged public keys are unauthenticated by the certificate authority (CA). In contrast, the static DHKE protocol cannot provide perfect forward secrecy (PFS) due to the establishment of the same common key for each times of protocol execution, where the static public keys have been authenticated by the CA. In 1986, Matsumoto et al. brought a concept of an implicit authenticated Diffie-Hellman protocol in a one round. They proposed the MTI group protocols based on the idea of combining the short-term and long-term secret keys to generate authenticated common key, thereafter this idea is considered a base in designing many two-pass authenticated key agreement (AKA) protocol [2].

The security properties provided by the designed protocols depend on the computed ephemeral shared key between two-party participants of each protocol and what the content of messages exchanged during a protocol run. The computed ephemeral shared key should resist any revelation of ephemeral and static keys allowed by stronger adversaries without breaking the protocol trivially [3]. The concept of a strong adversary was introduced in the extended CK (eCK) security model [4]. No matter using heuristic arguments or formal security model in order to analyze the security claims for the designed protocol, the protocol security analysis should consider the content of messages exchanged or manipulated during a run of the protocol to establish a session key. The shared static key between two-party participants is a part of an AKA protocol equation for generating

DOI: 10.24507/icicel.13.04.293

a session key in the sense that should not be transmitted over an insecure open communication channel consequently to that it might subvert the security claims for a designed AKA protocol. The authors of the proposed protocol provided either formally security proof or heuristically arguments for a sake of proving the security claims of the proposed protocol. However, they did not claim that their proposed protocol is insecure. Since many protocols have been proposed, but some of them have been revealed to have security flaws [3,5-8], many formal verification tools have emerged to analyze and verify the security claims of the proposed protocol. Therefore, the Scyther tool was used to verify the security claimed properties of the ISO/IEC 11770 standard for key management techniques and the result was that some of the protocols cannot fit their security claimed properties in the standard [9].

Elkamchouchi and Abu Elkair proposed an efficient protocol for authenticated key agreement, and they claimed that the proposed protocol provides desirable security properties and resists known attacks on an AKA protocol [10]; henceforth we call it as ElKamchouchi and Abu Elkair protocol in our letter. However, we show that ElKamchouchi and Abu Elkair protocol cannot withstand the two types of key-compromise impersonation (KCI) attacks and MITM attack. In addition, this letter shows that ElKamchouchi and Abu Elkair protocol cannot provide PFS and known key security (KKS) security properties. Moreover, we use the Scyther tool as an automated verification tool to demonstrate the security flaws in ElKamchouchi and Abu Elkair protocol.

This letter is organized as follows. In Section 2, we review Elkamchouchi and Abu Elkair protocol and its security properties. In Section 3, we show that Elkamchouchi and Abu Elkair protocol cannot withstand KCI and MITM, and do not have PFS and KKS security properties. In Section 4, we demonstrate the security flaws in Elkamchouchi and Abu Elkair protocol by using the Scyther tool. We conclude our letter in the last section.

2. Review of Elkamchouchi and Abu Elkair Protocol.

This section reviews Elkamchouchi and Abu Elkair protocol [10].

2.1. Notations.

The notations used in this letter are summarized in Table 1.

Table 1. The notations used in this letter

Notations	Descriptions
p	Large prime (usually at least 1024 bits).
q	Prime (typically of 160 bits) with $q \mid p-1$.
G	Subgroup of Z_p^*. G is often a subgroup of order q.
g	Generator of G.
A, B	The initiator and responder communicating parties, respectively.
E	Malicious station. E has full control of communications over a network.
r_A, r_B	Random integers, chosen by A and B, respectively.
x_A, x_B	The static private keys of A and B, respectively.
$\bar{r}_A$, $\bar{r}_B$	The multiplicative inverse modulo the group order of r_A, r_B, respectively.
y_A, y_B	The public static keys of A and B, where $y_A \equiv g^{x_A}$ and $y_B \equiv g^{x_B}$.

2.2. Security models and properties.

The Dolev-Yao (DY) formal model introduced the capabilities of an adversary over an open communication model in order to analyze cryptographic protocol security [11]. The BR model presented a concept of an entity authentication from the matching conversation [12]. The CK model brought a session-state query for analyzing the protocol security [13]. The eCK model introduced an ephemeral-key query instead of the session state in the CK model in the sense of capturing attacks from the leakage of ephemeral and static keys [4]; thereafter many formal security models

have been proposed to clarify the difference between ephemeral-key reveal and session-state reveal [14-17].

We list some desirable security attributes for AKA protocols, for further details in the security properties of AKA, see [3,5,6,15,18,19] and we address three types of KCI according to stronger adversary attacks.

Let A and B be two honest parties. The fundamental security requirements for a key agreement protocol are as follows.

- *The static-KCI*: If party A's static private key is compromised, an adversary is able to impersonate A. However, this should not enable him to impersonate other entities to A.
- *The ephemeral-KCI*: If party A's ephemeral private key in a non-matching session is compromised, an adversary is able to impersonate A. However, this should not enable him to impersonate other entities to A.
- *The general-KCI*: If one of the both secret keys of party A's in a non-matching session is not compromised, an adversary should not be enabled to impersonate any entity to other entities in the network model.
- *PFS*: if eavesdropper might reveal any possible pairs of secret information without both of private keys (static and ephemeral secret keys) owned by that party, it should not have any effect on the secrecy of previously established session keys.
- *Known key security (KKS)*: A protocol run should result in a unique secret session key. If this key is compromised, it should have no impact on either a passive adversary to compromise future session keys, or impersonation by an active adversary in the future.

2.3. **Review of an efficient protocol for authenticated key agreement.** Elkamchouchi and Abu Elkair protocol consists of three phases: the registration phase, the transfer and verification phase, and the key generation phase as shown in Figure 1 [10].
Registration Phase. The prime number p and its generator g are registered to the public file. Each user needs to register his/her public static key in the public file like A and B as follows.

- A selects a static random integer number x_A, $2 \leq x_A \leq p-2$, and registers $y_A \equiv g^{x_A} \bmod p$ to the public file.
- B selects a static random integer number x_B, $2 \leq x_B \leq p-2$, and registers $y_B \equiv g^{x_B} \bmod p$ to the public file.

Transfer and Verification, and Key Generation Phases. To establish the session key between A and B they must do the following.
(1) A generates the ephemeral key r_A such that $2 \leq r_A \leq p-2$, and calculates $\bar{r}_A$ ($\bar{x}_A$ in the original, we correct it to $\bar{r}_A$) where $r_A \cdot \bar{r}_A \equiv 1 \bmod p-1$.
(2) B generates the ephemeral key r_B such that $2 \leq r_B \leq p-2$, and calculates $\bar{r}_B$ ($\bar{x}_B$ in the original, we correct it to $\bar{r}_B$) where $r_B \cdot \bar{r}_B \equiv 1 \bmod p-1$.
(3) A gets B's public key y_B from the public file, and calculates $y_B^{x_A} \equiv g^{x_A x_B} \bmod p$, and then sends it to B.
(4) B gets A's public key y_A from the public file, and calculates $y_A^{x_B} \equiv g^{x_A x_B} \bmod p$, and then sends it to A.
(5) B receives A's value and:

- Computes $(y_B^{x_A})^{r_B} \bmod p$
- Calculates $((y_B^{x_A})^{r_B})^{\bar{r}_B} \bmod p$ and compares it with $y_A^{x_B} \equiv g^{x_A x_B} \bmod p$, if the comparison is true, B sends $(y_B^{x_A})^{r_B} \bmod p$ to A and computes the session key $k_{AB} \equiv ((y_A^{x_B})^{r_A})^{r_B} * (y_A^{x_B})^{r_A} * (y_B^{x_A})^{r_B} \bmod p \equiv g^{x_A x_B (r_A + r_B + r_A r_B)} \bmod p$

(6) Unless the comparison is true, B will reject the received value.
From the A's point of view:
(1) A receives $y_A^{x_B} \bmod p$ and

- Computes $(y_A^{x_B})^{r_A} \bmod p$
- Calculates $((y_A^{x_B})^{r_A})^{\bar{r}_A} \bmod p$ and compares it with $y_B^{x_A} \equiv g^{x_A x_B} \bmod p$, if the comparison is true, A sends $(y_A^{x_B})^{r_A} \bmod p$ to B and computes the session key $k_{AB} \equiv ((y_B^{x_A})^{r_B})^{r_A} * (y_B^{x_A})^{r_B} * (y_A^{x_B})^{r_A} \bmod p \equiv g^{x_A x_B (r_A + r_B + r_A r_B)} \bmod p$

(2) Unless the comparison is true, A will reject the received vector.

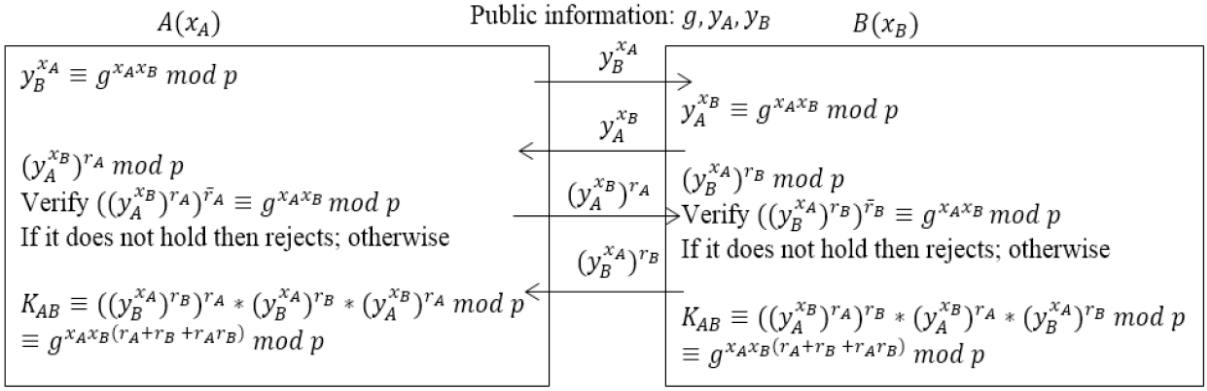

FIGURE 1. Elkamchouchi and Abu Elkair protocol

3. **Security Flaws in Elkamchouchi and Abu Elkair Protocol.** This section demonstrates the flaws in Elkamchouchi and Abu Elkair protocol.

3.1. **KCI attacks.** We show that Elkamchouchi and Abu Elkair protocol is vulnerable to the static-KCI attack in Section 3.1.1. In Section 3.1.2, we demonstrate that Elkamchouchi and Abu Elkair protocol cannot withstand the general-KCI attack.

3.1.1. *The static-KCI.* The static-KCI attack on Elkamchouchi and Abu Elkair protocol is shown in Figure 2. The following is the attack steps:
(1) A generates the ephemeral key r_A such that $2 \leq r_A \leq p-2$, and calculates $\bar{r}_A$ where $r_A \cdot \bar{r}_A \equiv 1 \bmod p-1$.
(2) E generates the ephemeral key r_E such that $2 \leq r_E \leq p-2$, and calculates $\bar{r}_E$ where $r_E \cdot \bar{r}_E \equiv 1 \bmod p-1$.
(3) A gets B's public key y_B from the public file, and calculates $y_B^{x_A} \equiv g^{x_A x_B} \bmod p$, and then sends it to B.
(4) E gets B's public key y_B from the public file, and calculates $y_B^{x_A} \equiv y_A^{x_B} \equiv g^{x_A x_B} \bmod p$, and then sends it to A, because the shared static key has the same value between A and B stations.
(5) E receives A's value and:

- Computes $(y_B^{x_A})^{r_E} \bmod p$
- E is not necessary to calculate $((y_A^{x_B})^{r_E})^{\bar{r}_E} \bmod p$ and compares it with $y_A^{x_B} \equiv g^{x_A x_B} \bmod p$; because E is a dishonest party, and trivially computes the session key $k_{AE} \equiv ((y_B^{x_A})^{r_A})^{r_E} * (y_B^{x_A})^{r_A} * (y_B^{x_A})^{r_E} \bmod p \equiv g^{x_A x_B (r_A + r_E + r_A r_E)} \bmod p$

From the A's point of view:
(1) A receives $y_B^{x_A} \bmod p$ and

- Computes $(y_B^{x_A})^{r_A} \bmod p$

- Calculates $((y_B^{x_A})^{r_A})^{\bar{r}_A} \bmod p$ and compares it with $y_B^{x_A} \equiv g^{x_A x_B} \bmod p$, if the comparison is true, A sends $(y_B^{x_A})^{r_A} \bmod p$ to B and computes the session key $k_{AE} \equiv ((y_B^{x_A})^{r_E})^{r_A} * (y_B^{x_A})^{r_E} * (y_B^{x_A})^{r_A} \bmod p \equiv g^{x_A x_B (r_A + r_E + r_A r_E)} \bmod p$

(2) Unless the comparison is true, A will reject the received vector.

Clearly, E can impersonate A to other parties using the shared secret key; therefore, Elkamchouchi and Abu Elkair protocol cannot withstand the static-KCI attack.

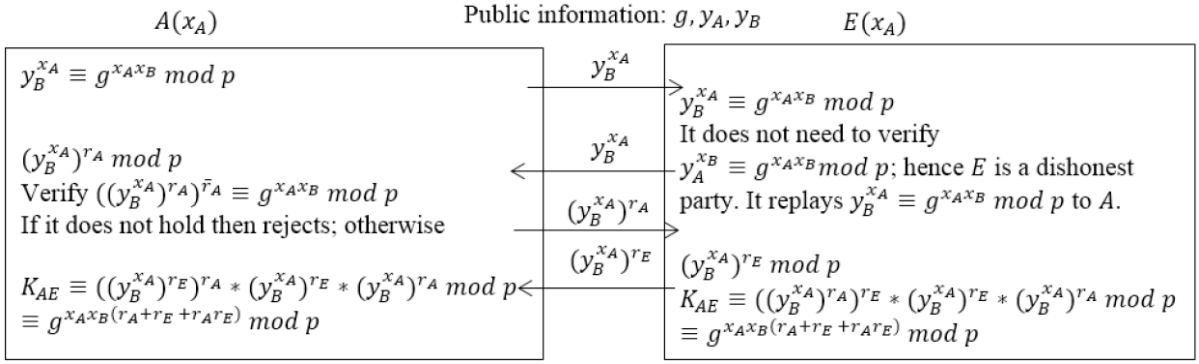

FIGURE 2. The static-KCI attack on Elkamchouchi and Abu Elkair protocol

3.1.2. *The general-KCI attack.* The exchanged messages between the two participating stations contain the shared static key before establishing a session key; therefore, an adversary can intercept a transmission between the two stations to obtain the shared static key between A and B stations; thereafter, an adversary can launch the general-KCI attack on Elkamchouchi and Abu Elkair protocol. The following is the attack steps.

(1) A generates the ephemeral key r_A such that $2 \leq r_A \leq p-2$, and calculates $\bar{r}_A$ where $r_A \cdot \bar{r}_A \equiv 1 \bmod p-1$.

(2) E generates the ephemeral key r_E such that $2 \leq r_E \leq p-2$, and calculates $\bar{r}_E$ where $r_E \cdot \bar{r}_E \equiv 1 \bmod p-1$.

(3) A gets B's public key y_B from the public file, and calculates $y_B^{x_A} \equiv g^{x_A x_B} \bmod p$, and then sends it to B, hence E intercepts the sent message from A and records it thereafter E replays the recorded message to A, because the shared static key between A and B stations has the same value.

(4) E receives A's value and:

- Computes $(y_B^{x_A})^{r_E} \bmod p$
- E is not necessary to calculate $((y_A^{x_B})^{r_E})^{\bar{r}_E} \bmod p$ and compares it with $y_A^{x_B} \equiv g^{x_A x_B} \bmod p$; because E is a dishonest party, and trivially computes the session key $k_{AE} \equiv ((y_B^{x_A})^{r_A})^{r_E} * (y_B^{x_A})^{r_A} * (y_B^{x_A})^{r_E} \bmod p \equiv g^{x_A x_B (r_A + r_E + r_A r_E)} \bmod p$

From the A's point of view:

(1) A receives $y_B^{x_A} \bmod p$ and

- Computes $(y_B^{x_A})^{r_A} \bmod p$
- Calculates $((y_B^{x_A})^{r_A})^{\bar{r}_A} \bmod p$ and compares it with $y_B^{x_A} \equiv g^{x_A x_B} \bmod p$, if the comparison is true, A sends $(y_B^{x_A})^{r_A} \bmod p$ to B and computes the session key $k_{AE} \equiv ((y_B^{x_A})^{r_E})^{r_A} * (y_B^{x_A})^{r_E} * (y_B^{x_A})^{r_A} \bmod p \equiv g^{x_A x_B (r_A + r_E + r_A r_E)} \bmod p$

(2) Unless the comparison is true, A will reject the received vector.

Clearly, E can impersonate A to other parties using the shared secret key; therefore, Elkamchouchi and Abu Elkair protocol cannot withstand the general-KCI attack.

3.2. **MITM attack.** The adversary E masquerades as B to A, and masquerades as A to B, by intercepting messages between A and B, and substitutes them by its own messages. At the end of the protocol run, A has established the secret $K_{AE} \equiv g^{x_A x_B (r_A + r_E + r_A r_E)} \bmod p$

session key with E, and B has established a secret session key $K_{BE} \equiv g^{x_A x_B (r_B + r_E + r_B r_E)} \bmod p$; therefore, Elkamchouchi and Abu Elkair protocol cannot withstand MITM attack.

3.3. **PFS attack.** The session secret key in Elkamchouchi and Abu Elkair protocol is $g^{x_A x_B (r_A + r_B + r_A r_B)} \bmod p$, which can be computed from knowledge of the ephemeral keys r_A or r_B and the transmitted messages; therefore, Elkamchouchi and Abu Elkair protocol does not provide PFS.

3.4. **KKS attack.** The exchanged messages in the third and fourth passes of Elkamchouchi and Abu Elkair protocol are $(y_A^{x_B})^{r_A} \bmod p$ and $(y_B^{x_A})^{r_B} \bmod p$ from A and B stations, respectively. The following is the attack steps:
(1) E intercepts the exchanged messages in the third and fourth passes and records them. E issues *EphemeralKeyReveal(sid)* or *EphemeralKeyReveal(sid*)* queries to obtain one of both ephemeral secret keys that owned by A and B stations, respectively.
(2) If E obtained the ephemeral private key r_A of A party then E does the following to obtain the shared static key between A and B:

- Computes $\bar{r}_A$, where $r_A \cdot \bar{r}_A \equiv 1 \bmod p - 1$
- Computes $((y_A^{x_B})^{r_A})^{\bar{r}_A} \bmod p \equiv y_A^{x_B} \bmod p$

Hence, the protocol is a symmetric protocol in case E revealed the ephemeral private key of B's party, E can compute the shared static key in the same way. Therefore, Elkamchouchi and Abu Elkair protocol cannot withstand KKS attack.

4. **Security Analysis via the Scyther Tool.** Cremers proposed the Scyther tool as a formal automated verification method for analyzing the security claims for the cryptographic protocol, where the Scyther is based on refinement algorithm [20]. Basin and Cremers extended the capabilities of the Scyther tool to capture weak PFS (wPFS), K-CI, and adversaries capable of strong corruptions and session state-reveal queries [21]. The Scyther had used to analyze and verify the security claims for the standard key management protocols (ISO/IEC 11770), and the new designed AKA protocols [9,22].

We verify the security flaws in Elkamchouchi and Abu Elkair protocol by using the Scyther tool version compromise-0.9.2 since it provides PFS, KCI, session-state reveal and ephemeral key reveal. The code of Elkamchouchi and Abu Elkair protocol in the Scyther description language is shown in Table 2.

Referring to Table 2 in [21] in order to verify the protocol in DY model, we set settings of the adversary model as shown in Figure 3, and the verification result shows that the protocol does not achieve the secrecy to the session key, and the shared static key as shown in Figure 4. Figure 4 shows that the protocol cannot withstand the general-KCI attack as shown in Section 3.1.2.

Table 2. The code of Elkamchouchi and Abu Elkair protocol

Role A	Role B
{*fresh x: Nonce;*	{*fresh y: Nonce;*
var Y, skab: Ticket;	*var X, skba: Ticket;*
send_1(A, B, exp(pk(B), sk(A)));	*recv_1(A, B, skba);*
recv_2(B, A, skab);	*match(skba, exp(pk(A), sk(B)));*
match(skab, exp(pk(B), sk(A)));	*send_2(B, A, exp(skba, y));*
send_3(A, B, exp(skab, x));	*recv_3(A, B, X);*
recv_4(B, A, Y);	*send_4(B, A, exp(exp(pk(A), sk(B)), y));*
claim(A, SKR, exp(Y, x));	*claim(B, SKR, exp(X, y));*
claim(A, SKR, exp(pk(B), sk(A)));}	*claim(B, SKR, exp(pk(A), sk(B)));*}

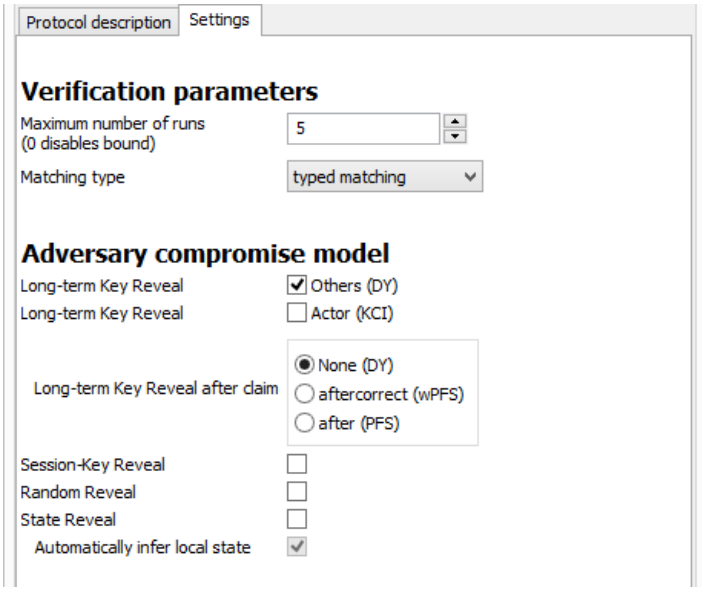

FIGURE 3. Settings of the adversary model

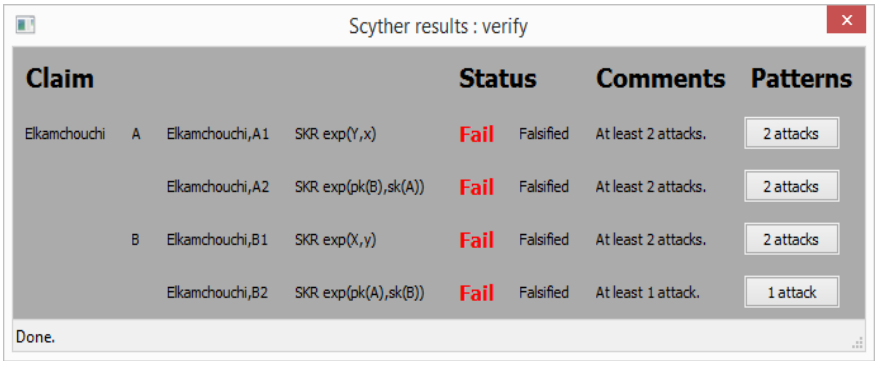

FIGURE 4. Verification result of the general-KCI attack on Elkamchouchi and Abu Elkair protocol

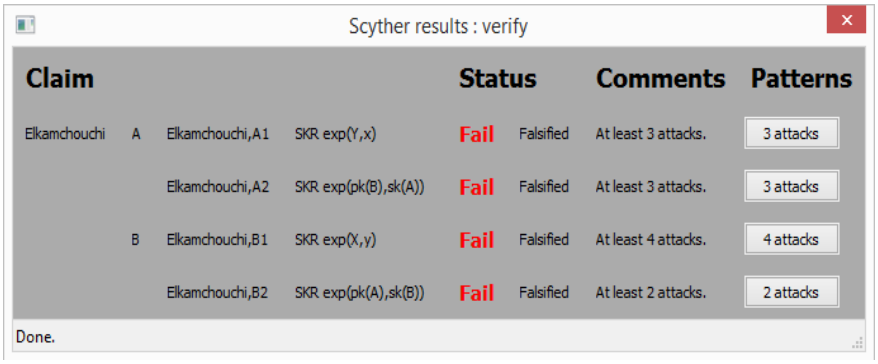

FIGURE 5. Verification result of the static-KCI attack on Elkamchouchi and Abu Elkair protocol

According to the attacks presented in Section 3.1.1, we modify settings of the adversary model in Figure 3 to the actor (KCI) in order to verify the static-KCI attacks on the protocol and the result of the verification is depicted in Figure 5 that shows the protocol cannot withstand the attack as demonstrated in Section 3.1.1. Figure 5 shows that Elkamchouchi and Abu Elkair protocol cannot withstand the static-KCI attacks in case the static private key of A party is compromised to an adversary.

In order to verify the KKS and PFS security properties, we modify settings of the adversary to random reveal and session-state and the verification result showed that the protocol does not achieve the secrecy claims according to the protocol description in the Scyther tool.

5. **Conclusions.** The security analysis of designed protocol should consider the number of messages exchanged between two intended parties in analyzing protocol security. Since the exchanged messages include the shared static key during the protocol run, this inclusion subverts the security claims of the protocol. We have shown that Elkamchouchi and Abu Elkair protocol cannot withstand two types of KCI attacks, MITM attack, KKS attack and does not have PFS. Further, we have used the Scyther tool to demonstrate the attacks on Elkamchouchi and Abu Elkair protocol.

In the future work, we will propose a new authenticated key agreement protocol which can withstand the attacks presented in this letter. Furthermore, we will analyze and verify the security claims of the proposed protocol by using the Scyther tool.

REFERENCES

[1] W. Diffie and M. E. Hellman, New directions in cryptography, *IEEE Trans. Inf. Theory*, vol.22, no.6, pp.644-654, 1976.

[2] T. Matsumoto, Y. Takashima and H. Imai, On seeking smart public-key-distribution systems, *IEICE TRANSACTIONS (1976-1990)*, vol.69, pp.99-106, 1986.

[3] Z. Mohammad, Y.-C. Chen, C.-L. Hsu and C.-C. Lo, Cryptanalysis and enhancement of two-pass authenticated key agreement with key confirmation protocols, *IETE Technical Review*, vol.27, no.3, pp.252-265, 2010.

[4] B. LaMacchia, K. Lauter and A. Mityagin, Stronger security of authenticated key exchange, *International Conference on Provable Security*, Berlin, Heidelberg, pp.1-16, 2007.

[5] Z. Mohammad and C.-C. Lo, Vulnerability of an improved elliptic curve Diffie-Hellman key agreement and its enhancement, *Proc. of EBISS2009*, pp.1-5, 2009.

[6] Z. Mohammad, C.-L. Hsu, Y.-C. Chen and C.-C. Lo, Cryptanalysis of a secure and efficient three-pass authenticated key agreement protocol based on elliptic curves, *Journal of Internet Technology*, vol.14, no.2, pp.247-250, 2013.

[7] P. Yu and W.-G. Shieh, The weaknesses and light improvement of Choi et al.'s anonymous multi-server authenticated key agreement scheme using smart cards and biometric, *ICIC Express Letters*, vol.12, no.2, pp.175-182, 2018.

[8] J. Kar, Cryptanalysis of provably secure certificateless short signature scheme by solving linear diophantine equations, *ICIC Express Letters*, vol.11, no.3, pp.619-624, 2017.

[9] C. Cremers and M. Horvat, Improving the ISO/IEC 11770 standard for key management techniques, *International Journal of Information Security*, vol.15, no.6, pp.659-673, 2016.

[10] H. M. Elkamchouchi and E. F. Elkair, An efficient protocol for authenticated key agreement, *Proc. of the 28th National Radio Science Conference (NRSC)*, pp.1-7, 2011.

[11] D. Dolev and A. Yao, On the security of public key protocols, *IEEE Trans. Information Theory*, vol.29, no.2, pp.198-208, 1983.

[12] M. Bellare and P. Rogaway, Entity authentication and key distribution, *Proc. of the 13th Annual International Cryptology Conference on Advances in Cryptology*, Berlin, pp.232-249, 1993.

[13] R. Canetti and H. Krawczyk, Analysis of key-exchange protocols and their use for building secure channels, *Proc. of International Conference on the Theory and Applications of Cryptographic Techniques*, Berlin, pp.453-474, 2001.

[14] C. J. Cremers, Session-state reveal is stronger than ephemeral key reveal: Attacking the NAXOS authenticated key exchange protocol, *Proc. of International Conference on Applied Cryptography and Network Security*, Berlin, pp.20-33, 2009.

[15] C. Cremers and M. Feltz, Beyond eCK: Perfect forward secrecy under actor compromise and ephemeral-key reveal, *Designs, Codes and Cryptography*, vol.74, no.1, pp.183-218, 2015.

[16] M. Feltz and C. Cremers, Strengthening the security of authenticated key exchange against bad randomness, *Designs, Codes and Cryptography*, vol.86, no.3, pp.481-516, 2018.

[17] J.-D. Wu, Y.-M. Tseng and S.-S. Huang, Efficient leakage-resilient authenticated key agreement protocol in the continual leakage eCK model, *IEEE Access*, 2018.

[18] S. Blake-Wilson, D. Johnson and A. Menezes, Key agreement protocols and their security analysis, *Proc. of IMA International Conference on Cryptography and Coding*, Berlin, pp.30-45, 1997.

[19] Z. Mohammad, C.-L. Hsu, Y.-C. Chen and C.-C. Lo, An efficient and secure three-pass authenticated key agreement elliptic curve based protocol, *International Journal of Innovative Computing, Information and Control*, vol.7, no.3, pp.1273-1284, 2011.

[20] C. Cremers, The Scyther tool: Verification, falsification, and analysis of security protocols, *Proc. of International Conference on Computer Aided Verification*, Berlin, pp.414-418, 2008.

[21] D. Basin and C. Cremers, Modeling and analyzing security in the presence of compromising adversaries, *Proc. of European Symposium on Research in Computer Security*, Berlin, pp.340-356, 2010.

[22] M. Lavanya and V. Natarajan, LWDSA: Light-weight digital signature algorithm for wireless sensor networks, *Sadhana*, vol.42, no.10, pp.1629-1643, 2017.