Electrical Power and Energy Systems 125 (2021) 106529

Contents lists available at ScienceDirect

International Journal of Electrical Power and Energy Systems

journal homepage: www.elsevier.com/locate/ijepes

Correcting "PALK: Password-based anonymous lightweight key agreement framework for smart grid"

Shehzad Ashraf Chaudhry

Department of Computer Engineering, Faculty of Engineering and Architecture, Istanbul Gelisim University, Istanbul, Turkey

ARTICLE INFO

Keywords:
Smart grid authentication
Anonymity
Elliptic curve cryptography
Security
Smart meter
Authenticated Key agreement
Incorrectness

ABSTRACT

Very recently in 2020, Khan et al. proposed an authentication scheme (*PALK*) for the smart grid infrastructure. Based on elliptic curve cryptography (ECC), symmetric hash functions and block cipher based encryption/decryption operations, the scheme was argued to work efficiently and securely in smart grid based infrastructure. However, in this paper, we prove that PALK has incorrect login and authentication phase; mainly, due to a superficial ECC operation involving the multiplication of two points over the curve. Moreover, in the scheme of Khan et al. the responding entity without knowing any clue of the initiator, uses the public key of the initiator for the completion of the authentication process, which is also not possible in the presence of multiple communicating devices. These design flaws lead to the situation, where the smart grid entities are unable to complete even a single cycle of authentication. Finally, we propose a quick solution to fix the pertinent flaws of the PALK. The security and correctness of the proposed solution iPALK is proved using formal BAN logic, automated tool ProVerif along with a brief discussion on the correctness of the scheme. The performance comparisons also show that the iPALK not only provides the correctness, but it is more efficient in terms of computation and communication costs.

1. Introduction

In recent times, the smart grid (SG) technology has gained much attention and it is expected to replace the conventional grid soon [2]. With bidirectional communication, the SG can manage the demand response in an efficient way and under consumer supervision. The consumer can directly adjust the power requirements for optimal usage [3,4,8]. The security and privacy are, however, the main concerns for the revolution of SG [5–7]. Very recently, in 2020 Khan et al. [1] proposed a new scheme *PALK: Password-based anonymous lightweight key agreement framework for smart grid*. PALK is proposed to secure communication between two SG entities (say U_a, U_b). Using elliptic curve cryptography and symmetric key operations, PALK supports the establishment of a secure channel between both the initiating and the responding entities (U_a, U_b). They proved the security of their scheme using multiple formal methods and AVISPA based automated methods. Moreover, they provided a discussion on many security features and attack resilience provision of PALK [1]. Khan et al. also compared PALK with ten related schemes in terms of efficiency and security. They substantiated the arguments of both the invincible security and performance efficiency of PALK. However, the discussion in this paper refutes Khan et al.'s claim of security and efficiency as the later part of this paper proves that Khan et al.'s scheme cannot even complete an authentication cycle owing to the critical design flaws. The rest of the paper is organized as follows: Section 2 provides a brief revisit of ECC. The revisit of the scheme of Khan et al. is provided in Section 3. Section 4 explains our arguments on the incorrectness of the scheme of Khan et al. In Section 5, we put forward a quick solution in terms of improved PALK to the pertinent flaws of the PALK. Section 6 proves the correctness and security of the proposed iPALK through formal, informal, and automated tools. The performance analysis and compassion of the proposed iPALK with original PALK are performed in Section 7. Finally, Section 8 concludes the paper.

2. Fundamentals of Elliptic Curve Cryptography

In this section, we briefly revisit ECC related concepts. The ECC is proved as more efficient as compared with RSA, DSA, and Diffie-Hellman cryptographic protocols and is defined by a curve $E_q(\alpha,\beta)$: $y^2 = x^3 + \alpha x + \beta \bmod q$, where the pair $\{\alpha,\beta\} \in Z_q$. The pair $\{\alpha,\beta\}$ is chosen carefully to satisfy $4\alpha^3 + 27\beta^2 \bmod q \neq 0$, where q is a prime number such that $|q| \geqslant 160$ bits. ECC curve $E_q(\alpha,\beta)$ consists of several points (x_i, y_i), including a point $\mathcal{O}$ as the point on infinity, which serves as an identity element while $E_q(\alpha,\beta)$ forms an abelian group. The ECC has only two operations:

https://doi.org/10.1016/j.ijepes.2020.106529
Received 22 June 2020; Received in revised form 8 September 2020; Accepted 11 September 2020
Available online 7 October 2020

- **ECC Point addition:** Given two points $A = (x_a, y_a)$ and $B = (x_b, y_b)$, the addition of A and B returns another point $C = A + B$ on the same curve. As C is a point over the curve, so it also consists of two coordinates (x_c, y_c), where $x_c = \lambda^2 - x_a - x_b$ and $y_c = (\lambda(x_a - x_c) - y_a)$. The λ is defined in the following ways:

$$\lambda = \begin{cases} \frac{3x_a^2 + \alpha}{2y_a} \bmod q & \text{if } A = B, \\ \frac{y_b - y_a}{x_b - x_a} \bmod q & \text{if } A \neq B \end{cases}$$

- **ECC Point Scalar Multiplication:** Given an integer $k \in Z_q$ and a point $A = (x_a, y_a)$, the scalar multiplication produces a point on the same curve $(D = kA)$ through repeated addition of A, thus $D = kA = A + A + A + \ldots + A$ (k *times*). As D is a point, so it is represented by two coordinate i.e. $D = (x_d, y_d)$.

Except for the two operations: (1) Point addition and (2) Point scalar multiplication, no other operation can be performed on ECC point/s. Specifically, for the two points A and B (whether $A = B$ or $A \neq B$), no such method exists which can find multiplication $(A.B)$ of these points.

3. Revisit of the PALK by Khan et al.

In the following subsections, we revisit the "PALK" proposed by Khan et al. to provide secure smart grid access to users. PALK consists of two types of entities (1) Trusted Authority (TA), responsible for construction and publication of system parameters and registration of communicating entities (2) Users/SG devices (U_A/U_B), to authenticate each other through the establishment of a secure channel.

3.1. Initialization

For the successful execution of this phase, the TA chooses an elliptic curve (EC) $E_q(\alpha, \beta)$ along with $P \in E_q(\alpha, \beta)$ as a public base point and a hash function $h(..)$. The TA chooses x_T as it's own private key and announces $\{E_q(\alpha, \beta), q, P, h(..)\}$.

3.2. Registration

Each user/device (U_i) initiates this phase and gets registers with the TA by selecting it's credentials i.e. identity ID_i and password PW_i. U_i then selects $x_i \in Z_q^*$ and computes $A_i = h(PW_i||x_i||ID_i), X_i = A_i.P$ and $PWI_i = PW_i \oplus h(ID_i||X_i)$. Then U_i sends $M_1 = \{PWI_i, ID_i, X_i, T_{R1}\}$ to TA using private channel, where T_{R1} is the current timestamp at U_i side. The TA receives M_1 and after checking the validity of $T_{R2} - T_{R1} \leqslant \Delta T$, chooses $x_T \in z_q^*$ and computes: $PW_i^* = PWI_i \oplus h(ID_i||X_i)$, $B_i = h(PW_i^*||x_T||ID_i)$, $Y_i = B_i.P, W_i = X_i + Y_i, S_i = h(ID_i||W_i||PWI_i), S_i^{'} = S_i \oplus h(W_i||X_i)$. The TA sends $M_2 = \{W_i, B_i, S_i^{'}\}$. The U_i receives M_2 and computes $S_i^* = S_i^{'} \oplus h(W_i||X_i)$, $SK_i = A_i + B_i + S_i^*, PK_i = SK_i.P$ and verifies $PK_i \overset{?}{=} (W_i + S_i^*.P)$ and on success, U_i keeps $\{SK_i, PK_i\}$ as it's secret and public key pair, respectively and stores $\{S_i^*, W_i\}$ in it's database.

3.3. Login and key agreement

The following steps are executed between the initiating entity U_A and responding entity U_B to perform mutual authentication and to establish a session key:

PALK 1 The U_A being initiator submits $\{ID_i, PW_i\}$ pair and computes $PWI_A = PW_A \oplus h(ID_A||X_A), R_A = h(ID_A||W_A||PWI_A)$ and checks the equality $R_A \overset{?}{=} S_A^*$. On success, U_A chooses $a \in Z_q^*$ and computes: $ID_{A1} = ID_A \oplus h(X_A||PWI_A||W_A)$, $L_1 = h(a.P||ID_A||X_A)$, $K_{A1} = h((T_1 \oplus aP)||aP$, $E_1 = E_{K_{A1}}(ID_{A1}, L_1, W_A, PWI_A, X_A)$ where T_1 denotes current timestamp and $E_{K_{A1}}$ denotes symmetric encryption. U_A then computes $Z = a.P, C^{'} = a \oplus h(Z.SK_A.P||T_1)$ and sends $M_1 = \{E_1, T_1, C^{'}, Z\}$ to U_B.

PALK 2 The U_B receives M_1, checks timestamp freshness through following relation $T_2 - T_1 \leqslant \Delta T$ and on success, U_B computes $C = C^{'} \oplus h(Z.PK_A||T_1)$ and $K_{B1} = h((T_1 \oplus CP)||CP)$. Now U_B decrypts E_1 using K_{B1} and extracts $(ID_{A1}, L_1, W_A, PWI_A, X_A)$ and computes $ID_A^* = ID_{A1} \oplus h(X_A||PWI_A||W_A)$, $L_1^* = h(a.P||ID_A||W_A)$ and checks $L_1^* \overset{?}{=} L_1$. If the equality holds, U_B chooses $b \in Z_q^*$ and computes $L_2 = h(ID_A^*||ID_B||W_A||W_B)$, $MAC_B = h(ID_A^*||ID_B||X_A||X_B||W_A||W_B||T_3)$, session key $SK_{BA} = h(ID_A^*||ID_B||L_2||MAC_B||W_A||W_B||b.a.P||T_3)$, $K_{B2} = h(L_1^*||ID_A^*||T_1||C)$, $ID_{B1} = ID_B \oplus h(X_B||ID_{A1}||L_1^*)$, $E_2 = E_{K_{B2}}(ID_{B1}, W_B, X_B, MAC_B, b.P, L_2)$, where $E_{K_{B2}}$ denotes symmetric encryption. U_B then sends $M_2 = \{E_2, T_3\}$ to U_A.

PALK 3 The U_A receives M_2, checks timestamp freshness through following relation $T_4 - T_3 \leqslant \Delta T$ and on success, U_A computes $K_{A2} = h(L_1||ID_A||T_1||a.P)$. Now U_A decrypts E_2 using K_{A2} and extracts $(ID_{B1}, W_B, X_B, MAC_B, b.P, L_2)$. The U_A now computes $ID_B^* = ID_{B1} \oplus h(X_B||ID_{A1}||L_1), L_2^* = h(ID_A||ID_B^*||W_A||W_B)$ and checks $L_2^* \overset{?}{=} L_2$. If the equality holds, U_A computes $MAC_A = h(ID_A||ID_B^*||X_A||X_B||W_A||W_B||T_3)$ and checks $MAC_A \overset{?}{=} MAC_B$. If the equality holds, U_A computes the session key $SK_{AB} = h(ID_A||ID_B^*||L_2^*||MAC_A||W_A||W_B||a.b.P||T_3)$.

4. Incorrectness of the PALK by Khan et al.

In this section, it is to argue that the scheme of Khan et al. entails incorrectness and may be stuck into the computation of an operation without any defined solution. Specifically, during login and authentication phase the initiator U_A computes $Z = a.P$ and $C^{'} = a \oplus h(Z.SK_A.P||T_1)$. Now, U_A along with the other parameter E_1 sends $\{E_1, T_1, Z, C^{'}\}$ to responder entity U_B. The U_B upon receiving $\{E_1, T_1, Z, C^{'}\}$ and after verifying freshness of the message, computes $C = C^{'} \oplus h(Z.PK_A||T_1)$ after then U_B proceeds with the rest of the steps as mentioned in the original paper of Khan et al. Two ambiguities/flaws arise here:

1. $Z = a.P$ is a point over curve $E_q(\alpha, \beta)$, the private key SK_A (of U_A) is an integer and P is also a point over $E_q(\alpha, \beta)$. The computation of $C^{'} = a \oplus h(Z.SK_A.P||T_1)$, requires computing $Z.SK_A.P$. The $Z.SK_A.P$ is supposed to be computed in any of the following two ways:
   (a) First SK_A is multiplied with Z, which results in another point $\widehat{Z} = Z.SK_A$ and now $\widehat{Z}.P$ is supposed to be computed for getting the result of $Z.SK_A.P$. Here, we are left with a multiplication of two points $\widehat{Z}$ and P.
   (b) Likewise, if SK_A is first multiplied with P, it also results in another point $PK_A = SK_A.P$ (the public key of U_A). Now $Z.PK_A$ is supposed to be computed for getting the result of $Z.SK_A.P$. This case is also similar to the above one and we are again left with the multiplication of two points Z and PK_A.

   Both the above-explained cases to compute $Z.SK_A.P$ results into a multiplication of two points and no method/algorithm exists to compute multiplication of two points over an elliptic curve. Hence, the scheme cannot proceed further and a halt position may occur. Similarly, on responding side U_A during computation of $C = C^{'} \oplus h(Z.PK_A||T_1)$, again tries to multiply a point Z with another point PK_A (where $PK_A = SK_A.P$ is the public key of U_A). This situation is also a halt. Hence, the scheme is incorrect and cannot complete execution normally to provide authentication and key agreement between two SG entities.

2. Secondly, the responding entity U_B is using public key PK_A of the initiating entity U_A for the computation of $C = C' \oplus h(Z.PK_A||T_1)$; whereas, U_B received $\{E_1, T_1, Z, C'\}$ tuple from U_A. The received tuple does not give any clue to identify the initiator. As SG is a communication infrastructure of multiple entities, therefore, if the responding entity knows who has sent a request, it can use the public key of that specific initiator for the computation of $C = C' \oplus h(Z.PK_A||T_1)$. Here, the responding entity has no information about the initiator. This ambiguity leads towards the total failure scenario of the PALK by Khan et al. for normal completion.

Hence, two above argued flaws in the PALK (scheme of Khan et al.), render the scheme as incorrect and cannot be deployed in any infrastructure.

5. iPALK-improved scheme

The existing Khan et al.'s PALK scheme incorporates two design flaws: (1) superficial ECC operation involving multiplication of two points, and (2) unknown sender identification. Therefore, the improvement consists of amending the steps to remove both of these flaws. Our improvement keeps initialization and registration phases of the PALK as it is and the following are the modified steps executed among two entities U_A and U_B during login and authentication phase:

iPALK 1 The U_A being initiator submits $\{ID_i, PW_i\}$ pair and computes $PWI_A = PW_A \oplus h(ID_A||X_A)$, $R_A = h(ID_A||W_A||PWI_A)$ and checks the equality $R_A \overset{?}{=} S_A^*$. On success, U_A chooses $a \in Z_q^*$ and computes $Z_1 = a.P$, $L_1 = h(Z_1||ID_A||X_A)$, $K_{A1} = h((T_1 \oplus Z_1)||Z_1)$, $E_1 = E_{K_{A1}}(ID_A, L_1, W_A, X_A)$, where T_1 denotes current timestamp and $E_{K_{A1}}$ denotes symmetric encryption. U_A then computes $Z_2 = a.PK_B = a.SK_BP$ and sends $M_1 = \{E_1, T_1, Z_2\}$ to U_B.

iPALK 2 The U_B receives M_1, checks timestamp freshness through following relation $T_2 - T_1 \leqslant \Delta T$ and on success, U_B computes $Z_1 = Z_2.SK_B^{-1}$, and $K_{B1} = h((T_1 \oplus Z_1)||Z_1)$. Now U_B decrypts E_1 using K_{B1} and extracts (ID_A, L_1, W_A, X_A). U_B now computes $L_1^* = h(Z_1||ID_A||X_A)$ and checks $L_1^* \overset{?}{=} L_1$. If the equality holds, U_B chooses $b \in Z_q^*$ and computes $L_2 = h(ID_A||ID_B||W_A||W_B)$, $MAC_B = h(ID_A||ID_B||X_A||X_B||W_A||W_B||T_3)$, session key $SK_{BA} = h(ID_A||ID_B||L_2||MAC_B||W_A||W_B||b.Z_1||T_3)$, $K_{B2} = h(L_1^*||ID_A||T_1||Z_1)$, $Z_3 = b.P$, $E_2 = E_{K_{B2}}(ID_B, W_B, X_B, MAC_B, Z_3, L_2)$, where $E_{K_{B2}}$ denotes symmetric encryption. U_B then sends $M_2 = \{E_2, T_3\}$ to U_A.

iPALK 3 The U_A receives M_2, checks timestamp freshness through following relation $T_4 - T_3 \leqslant \Delta T$ and on success, U_A computes $K_{A2} = h(L_1||ID_A||T_1||Z_1)$. Now U_A decrypts E_2 using K_{A2} and extracts $(ID_B, W_B, X_B, MAC_B, Z_3, L_2)$. The U_A now computes $L_2^* = h(ID_A||ID_B||W_A||W_B)$ and checks $L_2^* \overset{?}{=} L_2$. If the equality holds, U_A computes $MAC_A = h(ID_A||ID_B||X_A||X_B||W_A||W_B||T_3)$ and checks $MAC_A \overset{?}{=} MAC_B$. If the equality holds, U_A computes the session key $SK_{AB} = h(ID_A||ID_B||L_2^*||MAC_A||W_A||W_B||a.Z_3||T_3)$.

6. Correctness/security proofs

This section solicits the correctness and/or security of the proposed iPALK through informal discussion, formal and automated methods.

6.1. Discussion on correctness

The PALK scheme proposed by Khan et al. entails two design flaws: (1) multiplication of two ECC points, and (2) unknown sender identification. To remove both these flaws, in the proposed iPALK, we used the pubic key of the receiver to compute $Z_2 = a.PK_B = a.SK_BP$. Moreover, the parameter PWI_A was also unnecessary because it is not stored in the memory of the receiving device. Hence, sending PWI_A in E_1 is useless and only the secret parameters Z_1, X_A and the public identity ID_A can serve the purpose. So, we just keep these values in the computation of L_1 and the value of L_1 is verified on the receiver side using these values. Similarly, in iPALK the computation and transmission of incorrect parameter C' are also avoided, which saves significant computation and communication costs. When U_B receives the message $M_1 = \{E_1, T_1, Z_2\}$, using the multiplicative inverse of its' own private key SK_B, U_B computes $Z_1 = Z_2.SK_B^{-1}$ and based on computed Z_1, computes the decryption key $K_{B1} = h((T_1 \oplus Z_1)||Z_1)$. Now computed K_{B1} is used to decrypt E_1 and rest of the procedure continues. Likewise, when U_A receives $M_2 = \{E_2, T_3\}$, it computes decryption key $K_{A2} = h(L_1||ID_A||T_1||Z_1)$ correctly using the secret L_1 and Z_1 parameters and then continues for the rest of the procedure. Therefore, the proposed iPALK is free of such design flaws and uses only the correct procedure to complete the authentication among two entities of a smart grid environment.

6.2. Correctness of shared session key

In this subsection, we prove that the session key shared on both sides (U_A and U_B) is the same. During a successful cycle of authentication procedure the initiating entity U_A computes session key as follows:

$$SK_{BA} = h(ID_A||ID_B||L_2||MAC_B||W_A||W_B||b.Z_1||T_3) \tag{1}$$

The receiving entity U_B computes the session key as follows:

$$SK_{AB} = h\left(ID_A\middle|\middle|ID_B\middle|\middle|L_2^*\middle|\middle|MAC_A\middle|\middle|W_A\middle|\middle|W_B\middle|\middle|a.Z_3\middle|\middle|T_3\right) \tag{2}$$

Now we show that the session key computed in Eq. 1 is the same as computed in Eq. 2. Per discussion in Section 6.1, the receiver U_B computes accurate K_{B1} and using this key decrypts E_1 and extracts (ID_A, L_1, W_A, X_A). Likewise, when the initiator U_A receives the reply, it generates correct $K_{A2} = h(L_1||ID_A||T_1||Z_1)$ and using this, U_B extracts $(ID_B, W_B, X_B, MAC_B, Z_3, L_2)$ from E_2. Now, both the initiating and responding entities know the parameters $\{ID_A, ID_B, W_A, W_B, T_3\}$. U_B computes $MAC_B = h(ID_A||ID_B||X_A||X_B||W_A||W_B||T_3)$ and U_A computes the same $MAC_A = h(ID_A||ID_B||X_A||X_B||W_A||W_B||T_3)$. It can seen that both $MAC_B = MAC_A$ are same. Additionally, U_B computes $L_2 = h(ID_A||ID_B||W_A||W_B)$ and U_A verifies this value after computing $L_2^* = h(ID_A||ID_B||W_A||W_B)$, both of these values $L_2 = L_2^*$ are also same. Following relationship also proves that $b.Z_1 = a.Z_3$:

$$b.Z_1 = b.a.P \tag{3}$$

$$= a.b.p \tag{4}$$

$$= a.Z_3 \tag{5}$$

Therefore, all values in the computation of session key on both initiator and responder sides are the same and the session key is shared correctly among two smart grid devices.

6.3. Formal proof of correctness using BAN logic

In this section, we evaluate the security properties of the contributed model in consideration with Burrows-Abadi-Needham logic (BAN) logic [9]. The set of logics in BAN logic help us to evaluate the authentication protocol on the grounds of mutual authenticity and confidentiality of the session key as established among the legal participants. In this analysis, the principals (p and p') are the agents interacting in an authentication protocol. The employed notations are illustrated in Table 1. Moreover, few postulates or rules are defined in Table 2 to support the analysis.

The goals which are followed for proving the mutual authenticity in this analysis, are as follows:

Table 1
BAN logic notations.

Symbols	Representations
$p\vert\equiv\Phi$	p believes the statement Φ
$p\triangleleft\Phi$	p sees Φ
$p\vert\sim\Phi$	p once said Φ. Previously, the agent p had submitted Φ
$p\vert\Rightarrow\Phi$	p bears jurisdiction over Φ
$\#(\Phi)$	Φ is fresh
$\{\Phi,\Phi^{'}\}\vartheta$	Φ and $\Phi^{'}$ are encrypted through public key ϑ
$(\Phi,\Phi^{'})$	Φ and $\Phi^{'}$ depict the parts of message $(\Phi,\Phi^{'})$
$\langle\Phi,\Phi^{'}\rangle\vartheta$	Φ and $\Phi^{'}$ are encrypted with the symmetric key ϑ
$(\Phi,\Phi^{'})\vartheta$	Φ and $\Phi^{'}$ are hashed with a key ϑ
$p\leftrightarrow^{\vartheta}p^{'}$	p and $p^{'}$ exchange the messages using a secure key ϑ

Table 2
BAN logic postulates.

Symbols	Representations
Γ_1. Message meaning rule	$\frac{p\vert\equiv p\leftrightarrow^{\vartheta}p^{'},p\triangleleft<\Phi>_{\Phi}}{p\vert\equiv p^{'}\vert\sim\Phi}$
Γ_2. Nonce verification rule	$\frac{p\vert\equiv\#(\Phi),p\vert\equiv p^{'}\vert\sim\Phi}{p\vert\equiv p^{'}\vert\sim\Phi}$
Γ_3. Jurisdiction rule	$\frac{p\vert\equiv p^{'}\Rightarrow\Phi,p\vert\equiv p^{'}\vert\equiv\Phi}{p\vert\equiv\Phi}$
Γ_4. Freshness conjuncatenation rule	$\frac{p\vert\equiv\#(\Phi)}{p\vert\equiv\#(\Phi,\Phi^{'})}$
Γ_5. Belief rule	$\frac{p\vert\equiv(\Phi),p\vert\equiv(\Phi^{'})}{p\vert\equiv(\Phi,\Phi^{'})}$
Γ_6. Session key rule	$\frac{p\vert\equiv\#(\Phi),p\vert\equiv p^{'}\vert\equiv\Phi}{p\vert\equiv p^{'}\leftrightarrow^{\vartheta}p^{'}}$
Γ_7. Public key encryption rule	$\frac{p\vert\equiv_{\vartheta}\mapsto p^{'},p\triangleleft\{\Phi\}_{\vartheta}-1}{p\vert\equiv p^{'}\vert\sim p^{'}}$

G1: $U_B|\equiv U_A\leftrightarrow^{SK}U_B$

G2: $U_B|\equiv U_A|\equiv U_A\leftrightarrow^{SK}U_B$

G3: $U_A|\equiv U_A\leftrightarrow^{SK}U_B$

G4: $U_A|\equiv U_B|\equiv U_A\leftrightarrow^{SK}U_B$

Our protocol may be generically characterized as.

$m_1: U_A\rightarrow U_B: E_1, T_1, Z_2$
$m_2: U_B\rightarrow U_A: E_2, T_3$

We represent the generic protocol in idealized form as:

$m_1: U_A\rightarrow U_B: E_1,\ T_1,\ Z_2: \langle ID_A,\ (ID_A, X_A)_{Z_1},\ W_A,\ PWI_A,\ X_A\rangle_{KA_I},\ T_1,\ \{Z_1\}_{PK_B}$
$m_2: U_B\rightarrow U_A: E_2,\ T_3: \langle ID_B,\ W_B,\ X_B,\ (ID_A, W_A, X_A)_{(ID_B||X_B||W_B)},\ Z_3,\ (ID_A, W_A)_{(ID_B||W_B)}\rangle_{KB_I}, T_3$

The premises to support the analysis are delineated as:

$\varphi1:\ U_A|\equiv\#a, T_1$
$\varphi2:\ U_B|\equiv\#b, T_3$
$\varphi3:\ U_A|\equiv U_B\leftrightarrow^{SK_{ab}}U_A$
$\varphi4:\ U_B|\equiv U_B\leftrightarrow^{SK_{ab}}U_A$
$\varphi5:\ U_A|\equiv U_B|\Rightarrow b.P\approx Z_3$
$\varphi6:\ U_B|\equiv U_A|\Rightarrow a.P\approx Z_1$

After employing the above illustrated idealizations, symbols, premises and rules, we have the undermentioned derivations.

6.3.1. Correctness/accuracy based on mutual authentication

In order to verify the mutual authenticity-related properties between U_A and U_B, we envisage the m_1 and m_2 messages into idealized forms as:

$m_1: U_A\rightarrow U_B: \langle ID_A, (ID_A, X_A)_{Z_1}, W_A, PWI_A, X_A\rangle_{KA_I}, T_1, \{Z_1\}_{PK_B}$
$m_2: U_B\rightarrow U_A: \langle ID_B,\ W_B,\ X_B,\ (ID_A, W_A, X_A)_{(ID_B||X_B||W_B)},\ Z_3,\ (ID_A, W_A)_{(ID_B||W_B)}\rangle_{KB_I}, T_3$

Lemma 1. *U_B can prove the legitimacy of login request generated from U_A.*

Proof. U_A generates $m_1 = (E_1, T_1, Z_2)$ and submits towards U_B that verifies the authenticity of the source of the message as given below:

On the application of seeing rule, we have the following derivation,
$\mathcal{F}1: U_B\triangleleft\langle ID_A, (ID_A, X_A)_{Z_1}, W_A, PWI_A, X_A\rangle_{KA_I}, T_1, \{Z_1\}_{PK_B}$
Applying $\mathcal{F}1, \varphi4, \Gamma_1$,
$\mathcal{F}2: U_B|\equiv U_A|\sim\langle ID_A, (ID_A, X_A)_{Z_1}, W_A, PWI_A, X_A\rangle_{KA_I}, T_1, \{Z_1\}_{PK_B}$
Using $\varphi1, \varphi6$ and Γ_4,
$\mathcal{F}3: U_B|\equiv\#\langle ID_A, (ID_A, X_A)_{Z_1}, W_A, PWI_A, X_A\rangle_{KA_I}, T_1, \{Z_1\}_{PK_B}$
Using $\mathcal{F}2, \mathcal{F}3$ and Γ_2, we have.
$\mathcal{F}4: U_B|\equiv U_A|\equiv\langle ID_A, (ID_A, X_A)_{Z_1}, W_A, PWI_A, X_A\rangle_{KA_I}, T_1, \{Z_1\}_{PK_B}$
Using $\varphi4, \mathcal{F}4$ and the application of Γ_3, we have.
$\mathcal{F}5: U_B|\equiv a.P\approx Z_1$

Thus, after having verified the freshness of timestamp T_1, $U_B$proves the authenticity of source of that message. □

Lemma 2. *U_A can suitably prove the legitimacy of the received response from U_B.*

Proof. In our contributed scheme, U_B constructs (E_2, T_3) message and submits to U_A besides timestamp T_3, in the response of login request from U_A. Then, U_A verifies the genuineness of U_B through monitoring the freshness of parameters as given below.

On the application of seeing rule, we have the derivation as: $\mathcal{F}6: U_B\triangleleft\langle ID_B, W_B, X_B, (ID_A, W_A, X_A)_{(ID_B||X_B||W_B)}, Z_3, (ID_A, W_A)_{(ID_B||W_B)}\rangle_{KB_I}, T_3$.
Using $\mathcal{F}6, \varphi3$ and $\Gamma1$,
$\mathcal{F}7: U_A\Big|\equiv U_B|\sim\langle ID_B,\ W_B,\ X_B,\ (ID_A, W_A, X_A)_{(ID_B||X_B||W_B)},\ Z_3,\ (ID_A, W_A)_{(ID_B||W_B)}\rangle_{KB_I}, T_3$
Using $\varphi2, \varphi5$, and Γ_4.
$\mathcal{F}8: U_A\Big|\equiv\#\langle ID_B,\ W_B,\ X_B,\ (ID_A, W_A, X_A)_{(ID_B||X_B||W_B)},\ Z_3,\ (ID_A, W_A)_{(ID_B||W_B)}\rangle_{KB_I}, T_3$
Using $\mathcal{F}7, \mathcal{F}8$ and Γ_2, we have.
$\mathcal{F}9: U_A\Big|\equiv U_B\Big|\equiv\langle ID_B,\ W_B,\ X_B,\ (ID_A, W_A, X_A)_{(ID_B||X_B||W_B)},\ Z_3,\ (ID_A, W_A)_{(ID_B||W_B)}\rangle_{KB_I}, T_3$
Using $\varphi3, \mathcal{F}9$ and applying Γ_3, we have.
$\mathcal{F}10: U_A|\equiv b.P\equiv Z_3$

Consequently, after having verified the freshness of timestamp, U_A validates the accuracy of source of the message.□

Theorem 1. *U_A and U_B can mutually authenticate each other.*

Proof. In consideration with Lemma 1, U_B may suitably prove the legitimacy of the login request from U_A. Likewise, referring to Lemma 2, U_A may also validate the genuineness of the response as received from U_B. Consequently, we can infer that U_A and U_B mutually authenticate each other. □

6.3.2. Session key agreement

A single session key, $SK = h(ID_A||ID_B||L_2||MAC_{A/B}||W_A||W_B||abP, T_3)$, could be agreed upon among the legal participants in the scheme. This mutual agreement in relation to session key among the members may be reached as:

Using $\varphi2, \mathcal{F}4$, and Γ_2, we have.
$\mathcal{F}11: U_B|\equiv U_A|\equiv U_B\leftrightarrow^{SK}U_A$ **(G2)**
Using $\varphi2, \mathcal{F}11$, and Γ_6.
$\mathcal{F}12: U_B|\equiv U_B\leftrightarrow^{SK}U_i$ **(G1)**

Using $\varphi 1, \mathcal{F}9$, and Γ_2, we have.
$\mathcal{F}13 : U_A | \equiv U_B | \equiv U_B \leftrightarrow^{SK} U_A$ **(G4)**
Using $\varphi 1, \mathcal{F}13$, and Γ_6.
$\mathcal{F}14 : U_A | \equiv U_B \leftrightarrow^{SK} U_A$ **(G3)**
Thus, the above set of logical statements with respect to BAN logic analysis adequately affirms the fact that the demonstrated scheme may genuinely attain mutual authenticity and key agreement between the legal members (U_A and U_B).

6.4. *Automated verification through ProVerif*

We used ProVerif [10] to verify the correctness and security of the improved scheme. The ProVerif is an applied π calculus-based automated tool and can verify: (1) Correctness, (2) Secrecy, and (3) Anonymity. ProVerif models cryptographic operations as constructors and equations. To verify the correctness, secrecy, and anonymity properties of the proposed scheme, we modeled the steps as explained in Section 5. The ProVerif's formal model for verification consisting of declaration, processes, and events/main part, we define the variables, constants, and public/private channels as well as constructors and equations in Fig. 1, while the processes and events are defined in Fig. 2. The two processes UA and UB are defined to exchange a session key among each other; whereas, we have also defined start and end events of both UA and UB. To verify the correctness of the improved scheme, to test the secrecy of the session key, and to verify the anonymity, we applied four queries as illustrated in Fig. 1. The results are as follows:

1. Query inj-event(end_UA(IDUA[])) ==> inj-event(start_UA(IDUA[])) is true.
2. Query inj-event(end_UB(IDUB[])) ==> inj-event(start_UB(IDUB[])) is true.
3. Query not attacker(IDA[]) is true.
4. Query not attacker(SKAB[]) is true.

The result 1 and result 2 attest that both the process UA and UB initiated and terminated successfully. This confirms the correctness of the proposed scheme; while result 3 confirms that IDi is not revealed to the attacker. This confirms the anonymity property. The result 4 confirms that SKAB is not revealed to the attacker.

```
 (*.................... Channels ...................*)
free ChSec:channel [private]. (*Secure channel between UA
     and UB*)
free ChPub:channel. (*Public Channel between UA and UB*)
 (*........... Constants and Variables ..........*)
free PWi :bitstring [private].
free IDUA :bitstring [private].
free SKAB :bitstring [private].
free SKB :bitstring [private].
free SKBA :bitstring [private].
free IDA:bitstring.
free PWA:bitstring.
free WA:bitstring.
free WB:bitstring.
free XB:bitstring.
free SA':bitstring.
free Z3:bitstring.
free MACB:bitstring.
free IDB:bitstring.
free XA:bitstring.
free IDUA :bitstring.
free IDUB :bitstring.
(*========Constructors========*)
fun h(bitstring):bitstring.
fun ECPM(bitstring,bitstring):bitstring.
fun Inverse(bitstring):bitstring.
fun Concat(bitstring,bitstring):bitstring.
fun XOR(bitstring,bitstring):bitstring.
fun Mul(bitstring,bitstring):bitstring.
fun EKA1(bitstring,bitstring):bitstring.
fun EKB2(bitstring,bitstring):bitstring.
(*======Equations========*)
equation forall a:bitstring; Inverse(Inverse(a))=a.
equation forall a:bitstring, b:bitstring; XOR(XOR(a,b),b)
     =a.
 (*..................... Queries ....................*)
query id:bitstring; inj event(end_UA(IDUA)) ==>
inj event(start_UA(IDUA)).
query id:bitstring; inj event(end_UB(IDUB)) ==>
inj event(start_UB(IDUB)).
query attacker(IDUA).
query attacker(SKAB).
(*=====*Events*=====*)
event start_UA(bitstring).
event end_UA(bitstring).
event start_UB(bitstring).
event end_UB(bitstring).
```

Fig. 1. ProVerif-Declarations.

```
 (*...........Login and Authentication........*)
 (*.....................UA....................*)
let pUA=
event start_UA(IDUA);
let PWIA = XOR(PWA, h(Concat(IDA,XA))) in
let RA = h(Concat(IDA,(WA,PWIA))) in
if RA = SA' then
new P:bitstring;
new a:bitstring;
let Z1 = ECPM(a,P) in
let L1 = h(Concat(Z1,(IDA,XA))) in
new T1:bitstring;
let KA1 = h(Concat((XOR(T1,Z1)),Z1)) in
let E1 = EKA1(IDA, (L1,WA, PWIA,XA)) in
new PKB:bitstring;
new C':bitstring;
let Z2 = ECPM(a,PKB) = ECPM(a,SKB) in
out(ChPub,(E1,T1,C',Z2));
in(ChPub,(E2:bitstring,T3:bitstring));
let KA2 = h((Concat(L1,(IDA,T1,Z1)))) in
let L2 = h((Concat(IDA,(IDB,WA,WB)))) in
if L2 = L2 then
let MACA = h(Concat(IDA,(IDB,XA,XB,WA,WB,T3))) in
if MACA = MACB then
let xSKAB = h(Concat(IDA,(IDB,L2,MACA,WA,WB,(ECPM(a,Z3)),
     T3))) in
event end_UA(IDUA)
else 0.
 (*.....................UB....................*)
let pUB=
event start_UB(IDUB);
in(ChPub,(E1:bitstring,T1:bitstring,C':bitstring,Z2:
     bitstring));
new SK1B:bitstring;
let Z1 = ECPM(Z2,SK1B) in
let KB1 = h(Concat((XOR(T1,Z1)),Z1)) in
let L1=h(Concat(Z1,(IDA, XA))) in
if L1=L1 then
let L2 = h(Concat(IDA,(IDB,WA,WB))) in
new T3:bitstring;
let xMACB = h(Concat(IDA,(IDB,XA,XB,WA,WB,T3))) in
new b:bitstring;
new P:bitstring;
let xSKBA = h(Concat(IDA,(IDB,L2,MACB,WA,WB,(ECPM(b,Z1)),
     T3))) in
let KB2 = h(Concat(L1,(IDA,T1,Z1))) in
let xZ3 = ECPM(b,P) in
let E2 = EKB2(IDB,(WB,XB,MACB,Z3,L2)) in
out(ChPub,(E2,T3));
event end_UB(IDUB)
else 0.
process (( !pUB) | (!pUA))
```

Fig. 2. ProVerif-Processes and Events.

7. Performance comparisons

This section provides a brief analysis/comparison of the efficiency of the improved iPALK and original PALK protocols in terms of storage, communication, computation, and respective running times. The iPALK is an improvement of the original PALK protocol to provide correctness by eliminating the design flaws and in the improved iPALK, we took the registration phase of the PALK as it is. Therefore, the storage costs of both the schemes (PALK and iPALK) are exactly same. Therefore, it is not being considered for comparison purposes. The iPALK completes authentication procedure by performing $5T_{pm} + 4T_{ed} + 14T_{hm}$ operations, where T_{pm} denotes point multiplication over an elliptic curve, T_{ed} stands for symmetric encryption and decryption operations; whereas, T_{hm} represents a oneway hash/mac function. For simplicity, we consider the same running time for each of the operations as was considered in the original article proposing PALK [1]. Therefore, $T_{pm} \approx 2.226$ ms, $T_{ed} \approx 0.0046$ and $T_{hm} \approx 0.0023$. The iPALK completes authentication in 11.1806 ms as a contrast to PALK which is 17.8701 ms.

Similarly, except for encryption operations, we consider same assumptions of PALK protocol for computing communication costs of each of the PALK and iPALK, which are as follows: the ECC point is considered as 320 bit long, the oneway hash/mac is kept as 160 bit long, identity is kept as 64 bit and timestamp as 32 bit of length; whereas, random numbers are assumed to be of 128 bit long. For encryption/decryption operations, we consider advanced encryption standard (AES) with 128 bit block size, as the consideration of 320 bit block size (as considered by Khan et al. in original PALK protocol) seems impractical because AES limits the block size to 128 bits. The original PALK protocol completes authentication in two messages $M_1 = \{E_1, T_1, C', Z\}$ transmitted

Table 3
Performance comparisons.

Scheme	PALK	iPALK
C_1	$8T_{pm} + 4T_{ed} + 19T_{hm}$	$5T_{pm} + 4T_{ed} + 14T_{hm}$
C_2	17.8701	11.1806
C_3	3136	2688

Note: ECC Point Multiplication: T_{pm}; Symmetric Encryption/Decryption: T_{ed}; One way Hash/MAC function: T_{hm}. C_1: Computation Cost; C_2: Running time in ms; C_3: Communication cost in bits.

from U_A to U_B and $M_2 = \{E_2, T_3\}$ sent from U_B to U_A. For M_1, $\{E_1, T_1, C', Z\}$ are sent, the length in bits of $T_1 = 32, C' = 320$ and $Z = 320$; whereas, $E_1 = E_{K_{A1}}(ID_{A1}, L_1, W_A, PWI_A, X_A)$ and the length of $ID_{A1} = 64$, $L_1 = 160, W_A = 320, PWI_A = 160, X_A = 320$. Therefore, total length of $E_1 = 1024$ bit and considering 128 bit block size, it requires to send 8 block $=128 \times 8 = 1024$ bits. Therefore, total bits transmitted for $M_1 = \{1024 + 32 + 320 + 320 = 1696\}$. Likewise, $M_2 = \{E_2, T_3\}$ takes 1440 bits for transmission, where $T_3 = 32$ bits and the size of $E_2 = E_{K_{B2}}(ID_{B1}, W_B, X_B, MAC_B, b.P, L_2)$ is $\{64 + 320 + 320 + 160 + 320 + 160 = 1344\}$ and to carry 1344 bits, we need $11, 128$ bit long blocks, accumulating the bit size of E_2 to $128 \times 11 = 1408$. Therefore, $M_2 = \{1408 + 32 = 1440\}$ bits long. Hence, the total communication cost of PALK is $1696 + 1440 = 3136$ bits. The iPALK completes authentication by exchanging two messages $M_1 = \{E_1, T_1, Z_2\}$ and $M_2 = \{E_2, T_3\}$. For transmission of M_1, the $E_1 = E_{K_{A1}}(ID_A, L_1, W_A, X_A)$ has accumulative size $\{64 + 160 + 320 + 320 = 864\}$ and total blocks needed to encrypt E_1 are 7 and the size for transmission is $128 \times 7 = 896$. Therefore, total communication cost for transmission of M_1 is $\{1024 + 320 + 64 = 1248\}$. Similarly, the size of $E_2 = E_{K_{B2}}(ID_B, W_B, X_B, MAC_B, Z_3, L_2)$ in $M_2 = \{E_2, T_3\}$ is $\{64 + 320 + 320 + 160 + 320 + 160 = 1344\}$ and it needs 11 blocks each of 128 bits length for encryption. Therefore total encrypted size of $E_2 = 128 \times 11 = 1408$ and total size of $M_2 = 1408 + 32 = 1440$ bits. Hence, the total communication cost of iPALK is $= 1248 + 1440 = 2688$. The comparison of proposed iPALK with original PALK demonstrates that iPALK has reduced 14.2% communication and 37.4% computation overhead as compared to Khan et al.'s PALK protocol. Table 3 depicts the performance comparisons of the proposed iPalk with PALK.

8. Conclusion

This paper unveils the incorrectness of the scheme of Khan et al. It has been shown in this paper that the login and authentication phase of Khan et al.'s PALK scheme is faulty owing to the extraneousness operation of the multiplication of two ECC points. Moreover, it is also shown that the receiving entity uses the public key of the initiator without recognizing the initiator. Consequently, it is proved in this paper that Khan et al.'s authentication scheme for the smart grid cannot complete a single authentication cycle. Therefore, the scheme cannot be deployed in any sort of infrastructure including the smart grid. Finally, we proposed improvements to remove the design flaws of Khan et al.'s PALK scheme with an aim to avoid such crucial mistakes in the future. The proposed improvements (iPALK) not only provides correctness but also reduces over 14% communication and 37% computation costs as compared with the original PALK protocol.

Author Statement

S.A. Chaudhry is the sole author of this paper.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Supplementary material

Supplementary data associated with this article can be found, in the online version, at https://doi.org/10.1016/j.ijepes.2020.106529.

References

[1] Khan AA, Kumar V, Ahmad M, Rana S, Mishra D. PALK: Password-based anonymous lightweight key agreement framework for smart grid. Int J Electrical Power Energy Syst 2020;121:106121.

[2] Kabalci Y. A survey on smart metering and smart grid communication. Renew Sustain Energy Rev 2016;57:302–18.

[3] Fang X, Misra S, Xue G, Yang D. Smart grid – the new and improved power grid: a survey. IEEE Commun Surv Tutor 2012;14:944–80.

[4] DeBlasio R, Tom C. Standards for the smart grid. In: Proceedings of the IEEE energy 2030 conference (ENERGY). Abu Dhabi; 4–5 November 2008. p. 1–7.

[5] Mahmood K, Arshad J, Chaudhry SA, Kumari S. An enhanced anonymous identity-based key agreement protocol for smart grid advanced metering infrastructure. Int J Commun Syst 2019;32:e4137. https://doi.org/10.1002/dac.4137.

[6] He D, Kumar N, Zeadally S, Wang H. Certificateless provable data possession scheme for cloud-based smart grid data management systems. IEEE Trans Industr Inf 2018;14(3):1232–41. https://doi.org/10.1109/TII.2017.2761806. March.

[7] He Debiao, Wang Huaqun, Khan Muhammad Khurram, Wang Lina. Lightweight anonymous key distribution scheme for smart grid using elliptic curve cryptography. IET Commun 2016;10(14):1795–802. https://doi.org/10.1049/iet-com.2016.0091 IET.

[8] Chaudhry SA, Alhakami H, Baz A, Al-Turjman F. Securing demand response management: a certificate-based access control in smart grid edge computing infrastructure. IEEE Access 2020;8:101235–43. https://doi.org/10.1109/ACCESS.2020.2996093.

[9] Burrow M, Abadi M, Needham R. A logic of authentication. ACM Trans Comput Syst 1990;8:18–36. https://doi.org/10.1145/77648.77649.

[10] Blanchet Bruno, et al. ProVerif 2.00: automatic cryptographic protocol verifier, user manual and tutorial. Version from 2018:05–16.

Shehzad Ashraf Chaudhry received the master's and Ph.D. degrees (with Distinction) from International Islamic University Islamabad, Pakistan, in 2009 and 2016, respectively. He is currently working as an Associate Professor with the Department of Computer Engineering, Faculty of Engineering and Architecture, Istanbul Gelisim University, Istanbul, Turkey. He has authored over 100 scientific publications appeared in different international journals and proceedings, including 76 in SCI/E journals. With an H-index of 24 and an I-10 index 51, his work has been cited over 1900 times. He has also supervised over 35 graduate students in their research. His current research interests include lightweight cryptography, elliptic/hyper elliptic curve cryptography, multimedia security, E-payment systems, MANETs, SIP authentication, smart grid security, IP multimedia subsystem, and next generation networks. He occasionally writes on issues of higher education in Pakistan.

Dr. Chaudhry was a recipient of the Gold Medal for achieving 4.0/4.0 CGPA in his Masters. Considering his research, Pakistan Council for Science and Technology granted him the Prestigious Research Productivity Award, while affirming him among Top Productive Computer Scientist in Pakistan. He has served as a TPC member of various international conferences and is an Active Reviewer of many ISI indexed journals. He also serves as guest editor in some prestigious journals.