IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 24, 2020, accepted October 14, 2020, date of publication October 28, 2020,
date of current version November 16, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3034447

RESEAP: An ECC-Based Authentication and Key Agreement Scheme for IoT Applications

MASOUMEH SAFKHANI[1], NASOUR BAGHERI[2,3], SARU KUMARI[4], HAMIDREZA TAVAKOLI[5], SACHIN KUMAR[6], (Member, IEEE), AND JIAHUI CHEN[7], (Member, IEEE)

[1]Computer Engineering Department, Shahid Rajaee Teacher Training University, Tehran 16788-15811, Iran
[2]Electrical Engineering Department, Shahid Rajaee Teacher Training University, Tehran 16788-15811, Iran
[3]School of Computer Science (SCS), Institute for Research in Fundamental Sciences (IPM), Farmanieh Campus, Tehran 19538-33511, Iran
[4]Department of Mathematics, Chaudhary Charan Singh University, Meerut 250004, India
[5]Department of Electrical and Computer Engineering, Hakim Sabzevari University, Sabzevar 96179-76487, Iran
[6]Department of Computer Science and Engineering, Ajay Kumar Garg Engineering College, Ghaziabad 201009, India
[7]School of Computers, Guangdong University of Technology, Guangzhou 510006, China

Corresponding author: Jiahui Chen (csjhchen@gmail.com)

This work was supported by the National Natural Science Foundation of China under Grant 61902079.

ABSTRACT Although the Internet of Things (IoT) provides many benefits for our life but it also raises many security threats. The main risk is the security of the transferred data comprising very critical information that its leakage compromises our privacy. In this regard, many security protocols have been introduced in literature, among which multi factor authentication protocols have been received considerable attention. in this paper, in the first step, the first third party security analysis of the newly proposed scheme denoted as ESEAP (designed by Kumari *et al.*) is presented. The provided analysis shows that this protocol has a number of security flaws including vulnerability to off-line password guessing attack, traceability attack, impersonation attack, insider attack and also desynchronization attack. For the second step, an enhanced protocol denoted as *RESEAP* is proposed in which we use physically unclonable function to improve its security. We prove the security of RESEAP informally and also formally in real or random model, which is a widely accepted security model to prove the security of a cryptographic protocol. While the security analysis confirms that RESEAP protocol has better security, its comparison with ESEAP also shows its higher efficiency.

INDEX TERMS Internet of Energy, smart grid, authentication, security analysis, Elliptic Curve Cryptography, Physical Unclonable Function.

I. INTRODUCTION

The Internet of Things (IoT), as a system of interrelated computing devices, allows devices to communicate with each other to transfer data over a network without the need for human-to-human or human-to-computer interaction. This technology enhances our daily life by providing infrastructure for other concepts such as Internet of Vehicles (IoV), Internet of Energy (IoE), Internet of Sensors (IoS) and Machine to Machine Communications (M2M). Combining IoT with the new advances in artificial intelligence and machine/deep learning provides many opportunities. However, that interconnected system of data flow also raises many challenges. More precisely, the devices that are connected through IoT, can capture and transfer many sensitive data that may easily compromise our privacy. Hence, there should be a mechanism to control the access to the captured and transferred data by any devices in an IoT system. To deal with this demand, security protocols can be employed to control the access to data. Such protocols are regularly used in our daily use of the internet, e.g. Secure Sockets Layer (SSL) [1] protocol. However, IoT is a heterogeneous network and it may not be possible to use those common solution because of the very constrained devices that are connected in this network, e.g. RFID tags and WSN nodes. To cope with these restrictions, researchers attempt to develop lightweight protocols for IoT system. Among them, authentication and session key agreement protocols receive more attention.

On the other hand, the edge device is widely distributed, can be compromised or stolen by the adversary to retrieve their entire sensitive data. To overcome such problems, Multi-Factor Authentication (MFA) protocols are proposed

The associate editor coordinating the review of this manuscript and approving it for publication was Bin Zhou.

in literature. In such protocols, for example to authenticate a protocol's instance, more than one factor should be provided by that instance. Secret keys, smart cards, smartphones and biometric features are among those factors that are widely used to design a secure protocol for different applications. A survey of MFA protocols is provided by Ometov *et al.* [2], in which they have provided a detailed analysis of factors that are currently utilized for MFA protocols with their corresponding operational requirements.

Although multi factor based schemes could provide better security, however, they also impose more complexity to the underlying scheme. Hence, it may not be possible to use them in constrained applications. Hence, two-factor based security protocols received more attention in literature. Among them, Haq *et al.* [3] recently proposed a two-factor lightweight authentication protocol using self-certified public key cryptography for multi-server 5G networks. Besides, they have analyzed a proposed protocol by Ying and Nayak [4], which is also a two factor authentication protocol, and have shown that it has important security flaws. Sinigaglia *et al.* [5] also provided a survey on the adoption of MFA and the design choices made by the banking sector in different countries. Qiu *et al.* [6] also proposed a lightweight two-factor authentication and key agreement protocol with dynamic identity based on Elliptic Curve Cryptography (ECC). In this paper, they also showed that the proposed protocol by Nikooghadam *et al.* [7] does not provide desired security against key-compromise impersonation attack and it also suffers from the lack of forward secrecy. It should be noted Nikooghadam *et al.* also have shown that Kumari *et al.*'s protocol [8] cannot resist off-line guessing attack or preserve user anonymity. It should be noted the Kumari *et al.*'s protocol has also analyzed by Kaul and Awasthi [9], where they have shown that the adversary can obtain the common session key of future communication between user and the server. Wu *et al.* [10] proposed a scheme to provide strong user anonymity and strong key agreement property. However, later Gupta and Chaudhariab [11] showed that it fails to provide those properties. In addition, they also proposed a two factor authentication protocol for roaming service in global mobility network and claimed security beyond traditional limit [11]. However, the proposed protocol is very costly and may not be useful for constrained environments, because it uses ECC and also quadratic residue, beside hash function. Xie *et al.* proposed a claimed to be provably secure dynamic ID-based two-factor authentication key exchange protocol [12]. However, later analysis by Li *et al.* [13] showed that it suffers from off-line dictionary attack. Fotouhi *et al.* [14] recently proposed a lightweight and secure two-factor authentication scheme for wireless body area networks in health-care IoT. In [14], they use hash function as the source of the security. Byun and Jeong [15] also analyzed the security of two password-based Physically Unclonable Function (PUF) embedded authentications protocols [16], [17] against off-line password guessing attack.

The above studies, beside many other studies, show that it is not possible to trust any security solution without extensive third party security analysis, that confirmed its security. In this regard, Kumari *et al.* [18] have recently analyzed the security of two-factor protocol proposed by Wang *et al.* [19] and shown that it suffers from off-line password guessing attack and impersonation attack. Besides, they also proposed an improved protocol denoted as ESEAP, which is an ECC based mutual authentication protocol using smart card and claimed to be secure and efficient. However, their claim cannot be trusted before thorough investigation. Hence, in this paper we want to evaluate the security of this protocol against the known attacks in the context. The contribution of this paper is as follows:

1) We show that ESEAP does not provides better security against ESEAP off-line password guessing attack and the adversary can guess the password, assuming that it has low entropy, i.e. selected from a finite set. Besides, we show that this protocol suffers from traceability attack, impersonation attack, insider attack and also desynchronization attack. The proposed attacks are in the security model used by the designers and contradict their claims.
2) We propose a revised protocol and name it RESEAP, in which we are using PUF to provide desired security against the proposed attacks.
3) We heuristically prove the security of the proposed protocol against different attacks and also formally in real or random model. Our security analysis shows that it provides desired semantic security in this model.
4) We compare the performance of RESEAP versus ESEAP and show that the revised protocol also outperforms ESEAP, beside its enhanced security.

In the rest of this paper, in section II we introduce the required backgrounds and also briefly explain ESEAP protocol. Next, in section III we analyze the security of this protocol by proposing several attacks. In section IV, we proposed the revised protocol, RESEAP and provide its security proofs in section V. Performance evaluation of the proposed protocol is in section VI. Finally, section VII concludes the paper.

II. PRELIMINARIES

In this section, we introduce required notations, a brief description of elliptic curve-based cryptography, and also represent ESEAP protocol. Table 1 lists notations used in this paper.

A. ELLIPTIC CURVE CRYPTOGRAPHY

Elliptic Curve Cryptography (ECC) is a public-key cryptography approach based on a group G, which is defined over an elliptic curve. The elliptic curve E_{F_q} is defined as the set of all $(x, y) \in F_q \times F_q$ such that $\lambda^2 = \mu^3 + a\mu + b$, where $a, b \in F_q$ and $4a^3 + 27b^2 \ mod \ q \neq 0$, along with a distinguished point at infinity which is denoted by $\mathcal{O}$.

TABLE 1. Notations.

Symbol	Description
E_{F_q}	Elliptic curve over the finite filed F_q
$\mathbb{G}$	A Prime group over E_{F_q}
q	A prime number
F_q^*	The set of integers $\{1, \ldots, q-1\}$
$a.P$	Multiplying P by scalar a, i.e. $a.P = P + P + \ldots + P$
S	The server
U	user/client
a, b, r	Random numbers
TS	Timestamp
ID	Identifier
P	Generator point of the group G
sk	Secret key
PK	Public key
PW	Password
$h(.)$	One-way hash function
$ES_K(.)/DS_K(.)$	Symmetric Encryption/Decryption using the key K
$\|$	Concatenation
$\oplus$	Bitwise XOR operation

Then $G = \{(\lambda, \mu) \in E_{F_q} \cup \mathcal{O}, +\}$ is a group. If $P \in G$ generates all elements of the group(,) then it is called a generator of the group. The order of an element $Q \in G$ is denoted as the smallest positive number n such that $nQ = \mathcal{O}$. When n is enough large, given any natural (?) scalar $a \in F_q$ and $P = \{(\lambda, \mu) \in E_{F_q}\}$ of order n, it is easy to calculate $y = a \times P$. However, given y, E_{F_q} and P, it is computationally infeasible to determine a, which is known as elliptic curve discrete logarithm problem (EC-DLP). Similarly, for $a, b \in F_q$, given $a \times P$, $b \times P$, E_{F_q} and P, it is computationally hard to determine $a \times b \times P$, which is known as Elliptic Curve Computational Diffie-Hellman Problem (EC-CDHP).

B. SEMANTIC SECURITY IN THE REAL-OR-RANDOM MODEL

In a password-authenticated key agreement schemes, the scheme's parties use their password in order to share a common session key SK (sk), that will be used to build secure channels [20], [21]. In such schemes, we consider a party as either a client $U \in \mathcal{U}$ or a trusted server $S \in \mathcal{S}$, where U could be either honest or malicious. It also holds a long-lived key password PW_U. The server S holds a vector $pw_S = \langle pw_S[U] \rangle_{U \in \mathcal{U}}$, contains an entry for each client U. $pw_S[U]$ defines a transformation of PW_U. If U is malicious client, then through security analysis one should assume that PW_U could be known by the adversary. If two clients U_i and U_j share the same session identifications, they are called *partners*.

To determine the adversary's advantage to distinguish a real session key from an ideal one, a bit b is defined which is chosen uniformly at random at the beginning of the semantic security game. In general terms, the adversary ($\mathcal{A}$) controls any communications over public channel passively or actively and it can run the following queries:

- Execute (U, S) query as a passive adversary $\mathcal{A}$ to eavesdrop the exchanged messages between U_i and S.
- Send $(U/S, m)$ query. This query models an active adversary that may intercept a message and then either modify it, create a new one (?), or simply forward it to U/S.
- Reveal (U) query to access the content of the client U_i.
- Test (U_i) is used to verify its guess for b. If no session key for instance U is defined or if a Reveal query was asked to either U or to its partner, then an undefined symbol $\perp$ is returned (?). Otherwise, return the session key for instance U if $b = 1$ or a random key of the same size if $b = 0$ (?).

Considering an execution of a password-authenticated key agreement protocol $\mathcal{P}$, in the presence of an adversary $\mathcal{A}$ with access to the Execute, Send, and Test oracles which outputs a guess bit b_0. The adversary's advantage to win the semantic security game in the Real-Or-Random (RoR) sense, by guessing the correct $b_0 = b$, is denoted by (as) $Adv^{RoR}_{\mathcal{D},\mathcal{P}}(t, R)$ and defined as follows [21] (following relation parentheses ?):

$$Adv^{RoR}_{\mathcal{D},\mathcal{P}}(t, R) = \big((Pr(\mathcal{A} \rightarrow b_0 = 1 : b = 1) - (Pr(\mathcal{A} \rightarrow b_0 = 1 : b = 0))\big)$$

$\mathcal{P}$ offers RoR semantic security if:

$$Adv^{RoR}_{\mathcal{D},\mathcal{P}(t,R)} < \varepsilon(.)$$

where $\varepsilon(.)$ is a negligible function (negligible value).

C. ESEAP PROTOCOL

We describe a brief description of ESEAP [18] in this section. It is a mutual authentication and key agreement protocol between a user and a server over public channel (?). The protocol includes several phases as initialization phase, registration phase and login and key agreement phase, password change phase, revocation phase and re-registration phase.

In the initialization phase of the scheme (), S chooses a random number x as its secret key and an elliptic curve E_{F_q} and a generator g over the group G and a hash function $h(.)$.

In the next phase, the user U is registered to the server S, over a secure channel as follows:

1) The user U chooses a password PW_U and an identifier ID_U, generates a random integer $a \in Z_q^*$ and calculates $PWU = h(ID_U \| PW_U \| a)$ and sends the message $\{PWU, ID_U\}$ to S.
2) Next, S generates a random number $b \in Z_q^*$ and sets $User\_List := \{ID_U, b, Honey\_List\}$. Besides (then), it computes $c = ID_U \oplus b$, $B_1 = h(ID_U \| x \| c \| b)$ and $L_1 = B_1 \oplus PWU$. In addition, S issues an smart card SC and sends it to U which includes $\{G, g, h(.), L_1, c\}$.
3) After receiving the message, U calculates $P_1 = a \oplus h(ID_U \| PW_U)$, $P_2 = h(PWU \| P_1)$ and $P_3 = a \oplus ID_U \oplus c$, deletes c from SC and stores $\{P_1, P_2, P_3\}$ in SC (,respectively).

Assuming that U registered to S successfully (After successfully registration of U in S), the login and authentication phase of ESEAP, over a public channel, is as follows:

1) The user U inserts its SC into the card reader along with its ID_U and PW_U.
2) SC calculates $a^* = P_1 \oplus h(ID_u \| PW_U)$ and $PWU^* = h(ID_U \| PW_U \| a^*)$ and verifies whether $P_2 \stackrel{?}{=} h(PWU^* \| P_1)$ to accept the login. Next, SC computes $B_1^* = L_1 \oplus PWU^*$, $C_1 = h(ID_U \| c)$, $C_2 = h(ID_U \| a \| C1)$, $K_{U1} = h(ID_U \| a \| P_3)$, $N = h(ID_U \| C_1 \| C_2 \| B_1 \| TS_{U1})$, generates a random number $u \in Z_q^*$ and calculates $E_1 = Es_{K_{U1}}(P_3, ID_U, C_1, C_2, u.g, N, TS_{U1})$ and sends the message $\{E_1, P_3\}$ to S.
3) Next, S computes $K_{S1} = h(ID_U \| P_3 \oplus P_3 \| P_3)$, $(P_3^*, ID_U^*, C_1^*, C_2^*, u.g^*, N^*, TS_{U1}^*) = Ds_{K_{U1}}(E_1)$ and checks $P_3 \stackrel{?}{=} P_3^*$. If that (the) checking does not pass, S searches its *User_List* to find a match for ID_U^*. Then, S verifies the received TS_{U1}, computes $B_1 = h(ID_U \| x \| c \| b)$ and verifies N^* to authenticate U. Then, S calculates $B_2 = h(ID_U^* \| x \| (P_3 \oplus b) \| b \| TS_{U1})$, chooses a random number $s \in Z_q^*$, computes a $K_{S2} = h(C_1^* \| C_2^* \| B_1 \| N^* \| TS_{S1})$, the session key $SK_S = h(ID_U^* \| ID_S \| K_{S2} \| N^*.g \| u.s.g \| B_2 \| TS_{S1})$, $V = h(u.g \| u.s.g \| s.g \| TS_{S1})$, $B_2' = B_2 \oplus h(B_1 \| ID_U^* \| ID_S)$, $ID_{S1} = ID_S \oplus h(b \| N^* \| B_1 \| B_2)$ and $E_2 = Es_{K_{S2}}(ID_{S1}, s.g, V, B_2')$. Then it sends the message $M_2 = \{E_2, TS_{S1}\}$ to U.
4) After receiving M_2, U verifies TS_{S1} and computes $K_{U2} = h(C_1 \| C_2 \| B_1 \| N \| TS_{S1})$ to decrypt $(ID_{S1}, s.g, V, B_2') = Ds_{K_{S2}}(E_2)$ and computes $B_2^* = B_1^* \oplus h(B_1 \| ID_U^* \| ID_S^*)$. Then, it checks $V \stackrel{?}{=} h(u.g \| u.s.g \| s.g \| TS_{S1})$ to authenticate S, computes $ID_S^* = ID_{S1} \oplus h((P_3 \oplus a) \| N \| B_1^* \| B_2^*)$ and the session key $SK_U = h(ID_U \| ID_S^* \| K_{U2} \| N.g \| u.s.g \| B_2^* \| TS_{S1})$.

At the end of this process, the shared session key is $SK = SK_U = SK_S$.

In the password change phase, U inserts SC along with its ID_U and PW_U into the card reader. SC computes $a^* = P_1 \oplus h(ID_u \| PW_U)$ and $PWU^* = h(ID_U \| PW_U \| a^*)$ and verifies whether $P_2 \stackrel{?}{=} h(PWU^* \| P_1)$ to accept the login. Then, U chooses the new password PW_U^{new} and computes $a_U^{new} = P_1 \oplus h(IDU \| PW_U^{new})$, $PWU^{new} = h(ID_U \| PW_U^{new} \| a^{new})$, $P_1^{new} = a^{new} \oplus h(ID_U \| PW_U^{new})$ and $L_1^{new} = B_1 \oplus PWU^{new}$. Next, PWU, P_1 and L_1 in SC are respectively replaced by PWU^{new}, P_1^{new} and L_1^{new}. It should be noted this phase included some typos which we fixed based on our understanding. For example, they stated "PW_U, P_1 and L_1 in SC are respectively replaced by PW_U^{new}, P_1^{new} and L_1^{new}" which could not be correct because PW_U is not stored in SC instead it includes PWU. Hence, we revised it as *PWU is replaced by* PWU^{new}.

Revocation phase of the protocol is very similar to the first two steps of the login and authentication phase. More precisely, in the user's side, $E_1 = Es_{K_{U1}}(P_3, ID_U, C_1, C_2, u.g, N, TS_{U1})$ is calculated in the login and authentication phase and the message $\{E_1, P_3, Revoke_{Request}, TS_{Rev1}\}$ is sent to S. Next, S verifies TS_{Rev1} and follows the approach similar to login and authentication phase to authenticate U. Once U has been authenticated, S sets b to *NULL* and afterward U cannot login, because S assumes its SC has been breached or stolen.

Re-registration phase is the last phase of ESEAP, which is used if the registered account of U does not work properly. In this phase, U sends $M_{RR} = \{PWU, ID_U, TS_{RR1}\}$ to S. Then, S verifies TS_{RR1}. If ID_U exists in the *User_List*, the previous SC is suspended and S follows the process of the registration phase.

III. SECURITY ANALYSIS OF ESEAP PROTOCOL

In this section, we review ESEAP in more dept. The attacks that are considered are those that the designers have explicit claim of security against, e.g. forward secrecy contradiction attack and insider attack.

A. OFF-LINE PASSWORD GUESSING ATTACK

The designers claimed that if the adversary $\mathcal{A}$ extracts the SC's data and eavesdrops the transferred messages between U and S, she/he cannot guess the password efficiently.

Let assume the user's password PW_U and identifier ID_U are in a finite set denoted by $\{S_1 : PW_U\}$ and $\{S_2 : ID_U\}$ but a is a random value. In addition, following the designers' assumption, assume $\mathcal{A}$ also extracted the content of SC that are $L_1 = PWU \oplus B_1$, $P_1 = a \oplus h(ID_U \| PW_U)$, $P_2 = h(PWU \| P_1)$ and $P_3 = a \oplus ID_U \oplus c$. To do offline password guessing attack, $\mathcal{A}$ does as follows:

1) For any $PW_U^* \| ID_U^* \in \{S_1 \| S_2\}$:
   a) $a^* \leftarrow P_1 \oplus h(ID_U^* \| PW_U^*)$;
   b) $PWU^* \leftarrow PWU = h(ID_U^* \| PW_U^* \| a)$;
   c) if $P_2 = h(PWU^* \| P_1)$ returns a^*, PW_U^* and ID_U^* respectively as a, the user's password PW_U and identifier ID_U.
2) Given a, PW_U and ID_U, retrieves $B_1 = PWU \oplus L_1$, $c = P_3 \oplus ID_U \oplus a$, $b = P_3 \oplus c$, $C_1 = h(ID_U \| c)$, $C_2 = h(ID_U \| a \| C1)$ and $K_{U1} = h(ID_U \| a \| P_3)$.

Following the above attack, the adversary can determine the secret parameters, including the user's password PW_U and identifier ID_U with the complexity of $|\{S1 \| S_2\}|$, independent of $|a|$.

B. SMART CARD LOSS ATTACK

The designers claimed (claims) (According to... designers claims) that since it is not possible to do off-line password guessing attack, the adversary also cannot do SC loss attack, assuming that $\mathcal{A}$ stolen SC and obtained its content. However, following the given scenario in subsection III-A, obtaining the content of SC, $\mathcal{A}$ can determine the user's password PW_U and identifier ID_U, assuming that they have been selected from a finite set. Hence, similar to the explained attack, assuming that $\mathcal{A}$ has stolen the U's SC, she/he can retrieve all the parameters playing important role in

the protocol, i.e. ID_U, PW_U, PWU, B_1, c, b, C_1, C_2 and $K_{U1} = h(ID_U \| a \| P_3)$, as shown in subsection III-A. All these retrieved parameters are constant values as long as the user has not participated in a new registration or password change phase.

C. USER'S ANONYMITY

In a protocol, if the adversary can link two sessions in which the user U has participated with a non-negligible probability, that protocol is a target for the user's traceability. If a user could be traced then its anonymity will be compromised. In ESEAP, in each session, U sends $P_3 = a \oplus ID_U \oplus c$ to S. Given it is not updated from a session to another session (between two consecutive sessions), hence it could be the source of traceability of a target U with a high probability. It is worth noting that given two smart cards SC and SC' contain P_3 and P'_3, the probability of $P_3 = P'_3$ is $2^{-max(|a|,|b|)}$. Hence, the success probability of connecting two sessions in which SC is involved would be $1 - 2^{-|a|}$.

D. INSIDER ATTACK

Another attack, that designers claimed ESEAP is secure against, is insider attack. In this attack, we can assume that the server's administrator has access to any information related to the user that are stored at the server, the registration request sent by the user to S in the registration phase, the messages sent by the user to S in the login and authentication phase; and the insider attacker's target could be to retrieve PW_U. Following this assumption, the insider attacker has access to ID_U, b and c. On the other hand, P_3 is also transferred over public channel to S. Hence, the insider attacker can determine $a = P_3 \oplus b$. Given $PWU = h(ID_U \| PW_U \| a)$, the only unknown value to the insider attacker is PW_U, which is selected from a finite set. Hence, she/he can guess all possible values for PW_U and verify its correctness by verifying it in PWU to filter the wrong guesses. The complexity of determining PW_U is $|S_1|$ is expected to be a small space. Therefore, ESEAP suffers from insider attack.

E. USER IMPERSONATION ATTACK

To impersonate U, the adversary does as follows:

1) Assume $\mathcal{A}$ has access to SC, by stealing it, and also retrieved ID_U and PW_U following the attacks described in subsection III-A, subsection III-B or subsection III-D.
2) $\mathcal{A}$ inserts the stolen SC into the card reader along with the retrieved ID_U and PW_U.
3) SC calculates $a = P_1 \oplus h(ID_u \| PW_U)$ and $PWU = h(ID_U \| PW_U \| a)$, verifies whether $P_2 \stackrel{?}{=} h(PWU^* \| P_1)$ and accepts the login. It then computes $B_1 = L_1 \oplus PWU$, $C_1 = h(ID_U \| c)$, $C_2 = h(ID_U \| a \| C_1)$, $K_{U1} = h(ID_U \| a \| P_3)$, $N = h(ID_U \| C_1 \| C_2 \| B_1 \| TS_{U1})$, generates a random number $u \in Z_q^*$ and calculates $E_1 = Es_{K_{U1}}(P_3, ID_U, C_1, C_2, u.g, N, TS_{U1})$ and sends the message $\{E_1, P_3\}$ to S.
4) Next, S computes (?) $K_{S1} = h(ID_U \| P_3 \oplus b \| P_3)$, $(P_3, ID_U, C_1, C_2, u.g, N, TS_{U1}) = Ds_{K_{U1}}(E_1)$ and checks P_3. It also verifies the received TS_{U1}, computes $B_1 = h(ID_U \| x \| c \| b)$ and verifies N and authenticates $\mathcal{A}$ as the legitimate U. Then, S calculates $B_2 = h(ID_U \| x \| (P_3 \oplus b) \| b \| TS_{U1})$, chooses a random number $s \in Z_q^*$, computes $K_{S2} = h(C_1 \| C_2 \| B_1 \| N \| TS_{S1})$, the session key $SK_S = h(ID_U \| ID_S \| K_{S2} \| N.g \| u.s.g \| B_2 \| TS_{S1})$, $V = h(u.g \| u.s.g \| s.g \| TS_{S1})$, $B'_2 = B_2 \oplus h(B_1 \| ID_U \| ID_S)$, $ID_{S1} = ID_S \oplus h(b \| N \| B_1 \| B_2)$ and $E_2 = Es_{K_{S2}}(ID_{S1}, s.g, V, B'_2)$ (?). Then it sends the message $M_2 = \{E_2, TS_{S1}\}$ to U.
5) After receiving M_2, $\mathcal{A}$ computes $K_{U2} = h(C_1 \| C_2 \| B_1 \| N \| TS_{S1})$ decrypts $(ID_{S1}, s.g, V, B'_2) = Ds_{K_{S2}}(E_2)$ and computes $B_2 = B_1 \oplus h(B_1 \| ID_U^* \| ID_S^*)$. Then, it computes $ID_S = ID_{S1} \oplus h((P_3 \oplus a) \| N \| B_1 \| B_2)$ and the session key: $SK_U = h(ID_U \| ID_S \| K_{U2} \| N.g \| u.s.g \| B_2 \| TS_{S1})$.

Following this attack, $\mathcal{A}$ could impersonate U successfully and share the session key with the server. Hence, this protocol does not provide security against user impersonation attack.

F. DESYNCHRONIZATION ATTACK

The designers argued that it is not possible to do a desynchronization attack on ESEAP because no shared parameter is updated through login and authentication phase. However, suppose the attacker $\mathcal{A}$ gets access to the SC of the user and $\mathcal{A}$ retrieved (?) secret parameters based on the attacks given in subsection III-A or subsection III-B, (then ?) it can participate in a later (next) phase of the protocol, e.g. password change phase, revocation phase or re-registration phase, to desynchronize the legitimate user from S. Hence, this protocol is not secure against desynchronization attack. It worth noting (?) that if the SC is not replaced and the user comes to know that his/her SC is missing/lost then he/she can go for the revocation phase as stated in ESEAP.

IV. RESEAP: REVISED VERSION OF ESEAP

To remedy ESEAP, we make some modifications to fix its security flaws. The revised protocol is called RESEAP for simplicity. In RESEAP, to simplify the protocol, we omit the last two phases ESEAP, i.e. revocation phase and re-registration phase, but keep the sequence of other phases identical to ESEAP. Besides, we also equip each smart card with a secure and reliable Physical Unclonable Function $PUF(.)$. A secure and reliable $PUF(.)$ returns completely different responses $PUF(\mathcal{C})$ and $PUF(\mathcal{C}')$, given the challenges $\mathcal{C} \neq \mathcal{C}'$. Besides, it returns the same response for the same challenge, even if it is tested again and again. In addition, different PUFs should return completely different responses for the same challenges. Although designing such a PUF function is a challenging task but its research area has had many progresses already [22]–[24]. However, it is out of the scope of our study in this paper.

U	$Message$	S
	$\xleftarrow{SC}$	Stores $\{E_q(c,d), q, g, h(.), Es(.), ID_S, PK_S\}$ in a SC
Chooses a password PW_U and an identifier ID_U, generates $a \in Z_q^*$, calculates $PWU = h(PUF(ID_U\|PW_U) \oplus a)$, $sk_U = (PUF(IDUkPWU) + a)$, $PK_U = sk_U.g$, $M = h(PK_U, ID_U, T_U^{reg})$		
	$\xrightarrow{\{ID_U, PK_U, T_U^{reg}, M\}}$	Verifies T_U^{reg} and M, generates $b \in Z_q^*$, calculates $K_U = h(b\|sk_S\|ID_U)$ and sets $User\_List := \{ID_U, PK_U, K_U\}$
Sets $Honey\_List = 0$ and stores $\{a, PWU, K_U\}$ in the SC	$\xleftarrow{K_U}$	

FIGURE 1. RESEAP's Registration phase between *U* and *S* over secure channel.

We also use ECC and one-way hash function in the proposed protocol. Following this modification, the revised protocol is explained in the rest of this section.

A. INITIALIZATION PHASE

In the initialization phase of the scheme, S chooses an elliptic curve E_{F_q} and a generator g over the group G and a hash function $h(.)$, a random number sk_S as its secret key and computes its public key $PK_S = sk_S.g$. Next, S stores the system parameters, i.e. $\{E_q(c, d), q, g, h(.), Es(.), ID_S, PK_S\}$ in each SC which has been already equipped with a secure and reliable $PUF(.)$ in factory.

B. REGISTRATION PHASE

In this phase, following Figure 1, U receives a fresh SC and over a secure channel, it is registered to the server S as follows:

1) The user U chooses a password PW_U and an identifier ID_U, generates a random integer $a \in Z_q^*$, calculates $PWU = h(PUF(ID_U\|PW_U) \oplus a)$, $PK_U = (PUF(ID_U\|PW_U) + a).g$, $M = h(PK_U, ID_U, TS_U^{reg})$ and sends the message $\{ID_U, PK_U, TS_U^{reg}, M\}$ to S, where TS_U^{reg} is its current timestamp.
2) Next, S verifies the timestamp TS_U^{reg} and M to accept the U's registration. Next, it generates a random integer $b \in Z_q^*$, calculates $K_U = h(b\|sk_S\|ID_U)$ and sets $User\_List := \{ID_U, PK_U, K_U\}$ and stores it in an encrypted database. Besides, S sends K_U to U in order to confirm the successful registration.
3) After receiving K_U message, U sets $Honey\_List = 0$ and stores $\{a, PWU, K_U, Honey\_List\}$ in the SC.

C. LOGIN AND AUTHENTICATION PHASE

After successfully registration of U in S, the login and authentication phase of ESEAP, over a public channel, is as follows:

1) The user U inserts its SC into the card reader along with its ID_U and PW_U.
2) SC verifies whether $PWU \stackrel{?}{=} h(PUF(ID_U\|PW_U) \oplus a)$ to accept the login; otherwise increases $Honey\_List$ and if $Honey\_List = 4$ does not accept any new passwords for following 10 minutes. Next, SC generates a random number $r_U \in Z_q^*$, computes $M_1 = (PUF(ID_U\|PW_U) + a).r_U.PK_S$ and $M_2 = (PUF(ID_U\|PW_U) + a).r_U.g$, extracts its current timestamp TS_U, calculates $M_3 = h(M_1\|ID_S\|ID_U\|TS_U\|K_U)$ and $M_4 = (TS_U\|ID_S\|ID_U\|M_3) \oplus M_1$ and sends the message $\{M_4, M_2\}$ to S.
3) Next, S computes $M_1^* = sk_S.M_2$, $(TS_U^*\|ID_S^*\|ID_U^*\|M_3^*) = M_4 \oplus M_1^*$, verifies TS_u^* and ID_S^*, retrieves K_U from its encrypted database from the $User\_List$ using ID_U^* and also verifies whether $M_3^* \stackrel{?}{=} h(M_1^*\|ID_S\|ID_U^*\|TS_U\|K_U)$ to accept the login. Then, S generates a random number $r_S \in Z_q^*$, computes $M_5 = r_S.M_1^*$ and $M_6 = sk_s.r_S.g$, extracts its current timestamp TS_S, calculates $M_7 = h(M_5\|ID_U\|ID_S\|TS_S\|K_U)$ and $M_8 = (TS_S\|ID_U\|ID_S\|M_7) \oplus M_5$. Next, S sends the message $\{M_8, M_6\}$ to U.
4) After receiving the message, U computes $M_5^* = r_u.sk_U.M_6$, $(TS_S^*\|ID_U^*\|ID_S^*\|M_7^*) = M_8 \oplus M_5^*$ and verifies TS_S^* and ID_U^* and also verifies whether $M_7^* \stackrel{?}{=} h(M_5^*\|ID_U^*\|ID_S^*\|TS_S\|K_U)$ to authenticate S. Assuming S successfully authenticated, U computes the session key as $Sk_U = h(M_5\|TS_S^*\|TS_U\|ID_U\|ID_S\|K_U)$ and $M_9 = h(Sk_U\|M_7^*\|TS_S)$ and sends M_9 to S.
5) In order to authenticate U, S verifies whether $M_9 \stackrel{?}{=} h(Sk_U\|M_7^*\|TS_S)$.

At the end of this process (Figure 2), the shared session key is $SK = SK_U = SK_S$.

D. PASSWORD RENOVATION PHASE

The password change phase is as follows:

1) The user U inserts its SC into the card reader along with its ID_U and PW_U.
2) SC verifies whether $PWU \stackrel{?}{=} h(PUF(ID_U\|PW_U) \oplus a)$ to accept the login. Next, U inserts a new password PW_U^{new} and SC does as follows:
   a) computes $a^{new} = a + PUF(ID_U\|PW_U) - PUF(ID_U^{new}\|PW_U^{new})$

U	$Message$	S
Inserts SC into the card reader along with its ID_U and PW_U. SC verifies $PWU \stackrel{?}{=} h(PUF(ID_U\|PW_U) \oplus a)$, generates $r_U \in Z_q^*$ computes $M_1 = (PUF(ID_U\|PW_U) + a).r_U.PK_S$ and $M_2 = (PUF(ID_U\|PW_U) + a).r_U.g$, extracts TS_U, calculates $M_3 = h(M_1\|ID_S\|ID_U\|TS_U\|K_U)$ and $M_4 = (TS_U\|ID_S\|ID_U\|M_3) \oplus M_1$		
	$\xrightarrow{\{M_4,M_2\}}$	Computes $M_1^* = sk_S.M_2$, $TS_U^*\|(ID_S^*\|ID_U^*\|M_3^*) = M_4 \oplus M_1^*$ and verifies TS_U^*, ID_S^*, retrieve K_U from the $User\_List$ using ID_U^* and verifies $M_3^* \stackrel{?}{=} h(M_1^*\|ID_S\|ID_U^*\|TS_U\|K_U)$, generates $r_S \in Z_q^*$, computes $M_5 = r_S.M_1^*$ and $M_6 = sk_S.r_S.g$, extracts TS_S, calculates the session key $SK_S = h(M_5\|TS_S\|TS_U\|ID_U\|ID_S\|K_U)$, $M_7 = h(SK_S\|ID_U^*\|ID_S\|TS_S\|K_U)$, and $M_8 = (TS_S\|ID_U^*\|ID_S\|M_7) \oplus M_5$
Computes $M_5^* = r_U.sk_U.M_6$, $(TS_S^*\|ID_U^*\|ID_S^*\|M_7^*) = M_8 \oplus M_5^*$ and verifies TS_S^* and ID_U^*, computes $SK_U = h(M_5^*\|TS_S^*\|TS_U\|ID_U\|ID_S\|K_U)$ and checks $M_7^* \stackrel{?}{=} h(SK_U\|ID_U^*\|ID_S^*\|TS_S^*\|K_U)$ to authenticate S and $M_9 = h(Sk_U\|M_7^*\|TS_S^*)$	$\xleftarrow{\{M_8,M_6\}}$	
	$\xrightarrow{M_9}$	Verifies $M_9 \stackrel{?}{=} h(Sk_S\|M_7\|TS_S)$ to authenticate U

FIGURE 2. RESEAP's Login and authentication phase between *U* and *S* over public channel.

b) computes $PWU^{new} = h(PUF(ID_U\|PW_U^{new}) \oplus a^{new})$
c) overwrites $\{a^{new}, PWU^{new}\}$ in SC.

3) After successful password change, SC returns a message containing new password.

Following this process, the public key of U remains unchanged:

$$\begin{aligned} sk_U^{new} &= PUF(ID_U^{new}\|PW_U^{new}) + a^{new} \\ &= PUF(ID_U^{new}\|PW_U^{new}) + a + PUF(ID_U\|PW_U) \\ &\quad - PUF(ID_U^{new}\|PW_U^{new}) \\ &= PUF(ID_U\|PW_U) + a \\ &= sk_U \end{aligned}$$

V. SECURITY ANALYSIS OF RESEAP

Given any user is registered over a secure channel and received his/her SC securely, it is enough to evaluate the security of the other two phases, i.e. login and authentication phase and password change phase.

TABLE 2. Security comparison of RESEAP versus ESEAP, where MA, SKA, RA, ImA, TA, SDA, DOS, FS, FSC, InA, PG, U2UP, MIMA and DC respectively denote mutual authentication, secret key agreement, replay attack, impersonation attack, traceability attack, secret disclosure attack, desynchronization attack, forward secrecy, forward secrecy with compromise device, insider attack, password guessing and data confidentiality.

	MA	SKA	RA	ImA	TA	SDA
ESEAP	✓	✓	✓	×	×	✓
RESEAP	✓	✓	✓	✓	✓	✓
	DOS	FS	FSC	InA	PG	DC
ESEAP	×	✓	✓	×	×	✓
RESEAP	✓	✓	✓	✓	✓	✓

A. HEURISTIC SECURITY ANALYSIS

In this section, we show that RESEAP provides desired security against different attacks in the context, including the proposed attacks against ESEAP. Table 2 represents a summary of the security comparison of RESEAP versus ESEAP.

1) MUTUAL AUTHENTICATION

To accept the U's login, S verifies whether $M_3^* \stackrel{?}{=} h(M_1^*\|ID_S^*\|ID_U^*\|TS_U\|K_U)$, where $M_1^* = sk_S.M_2$, $M_1 = (PUF(ID_U\|PW_U)+a).r_U.PK_S$, $M_2 = (PUF(ID_U\|PW_U)+a).r_U.g$, $PK_S = sk_S.g$ and $PK_U = sk_U.g = (PUF(ID_U\|PW_U)+a).g$. After the following computations, if S receives the message correctly, then it will accept the login request:

$$\begin{aligned} M_1^* &= sk_S.M_2 \\ &= sk_S.(PUF(ID_U\|PW_U)+a).r_U.g \\ &= (PUF(ID_U\|PW_U)+a).r_U.PK_S \\ &= M_1 \end{aligned}$$

Then, U verifies whether $M_7^* \stackrel{?}{=} h(M_5^*\|ID_U^*\|ID_S^*\|TS_S^*\|K_U)$ to authenticate S, where $M_5 = r_S.M_1^*$, $M_6 = sk_s.r_S.g$, and $M_7 = h(M_5\|ID_U^*\|ID_S\|TS_S\|K_U)$. After the following computations, if U receives the message correctly, then it will authenticate S:

$$\begin{aligned} M_5^* &= sk_U.r_U.M_6 \\ &= sk_u.sk_S.r_S.r_U.g \\ &= sk_S.r_U.r_S.PK_U \\ &= M_5 \end{aligned}$$

To authenticate U, S computes $SK_S = h(M_5\|TS_S\|TS_U\|ID_U\|ID_S\|K_U)$ and verifies whether $M_9 \stackrel{?}{=} h(Sk_U\|M_7^*\|TS_S^*)$, where $SK_U = h(M_5^*\|TS_S^*\|TS_U\|ID_U\|ID_S\|K_U)$. Given $M_5 = M_5^*$ hence $SK_U = SK_S$ and S will authenticate U correctly. All in all, legitimate entities in this protocol are successfully authenticated.

2) SESSION KEY AGREEMENT

U computes the session key as $SK_U = h(M_5^*\|TS_S^*\|TS_U\|ID_U\|ID_S\|K_U)$ and S calculates it as $SK_S = h(M_5\|TS_S\|TS_U\|ID_U\|ID_S\|K_U)$. Given $M_5 = M_5^*$, then $SK_U = SK_S$ and S and U drive identical value for the session key.

3) REPLAY ATTACK

In RESEAP, any session is refreshed by random values r_U, r_S and timestamps TS_U and TS_S which are verified by U and S. Besides, the integrity of the timestamps and other values are guaranteed by one-way hash function. Hence, the adversary cannot replay a message belongs to previous sessions without being detected. It shows that RESEAP provides security against replay attack.

4) IMPERSONATION ATTACK

To do an impersonate attack, the adversary should either do a replay attack or generate a valid message. However, following subsection V-A3, the adversary has no chance to do replay attack. On the other hand, to impersonate S, the adversary should generate a valid M_5 which is not feasible. Impersonation of S, without the knowledge of sk_S and K_U and impersonation of U, without the knowledge of sk_u and K_U is not feasible and the adversary also has no chance to generate a valid set of M_1, M_2, M_3 and M_4, because it has no access to K_U and sk_i. The same argument can be deduced for impersonation of U. Hence, RESEAP is secure against impersonation attack.

5) TRACEABILITY AND ANONYMITY

The transferred messages over public channel, in the proposed protocol, are M_2, M_4, M_6, M_8 and M_9. Among them, M_4, M_8 and M_9 are produced using one-way hash functions and their inputs are randomized for each session, e.g. by timestamp and random numbers. Hence, the adversary cannot use them to trace the user or the server. On the other hand, $M_4 = (TS_U\|ID_S\|ID_U\|M_3) \oplus M_1$ and $M_1 = (PUF(ID_U\|PW_U)+a).r_U.PK_S$ and r_u is a random value generated by U. Hence, assuming that the used ECC has enough security, the adversary cannot use M_4 to trace U or S. A similar argument can be conducted for M_8. It shows that RESEAP is secure against traceability.

6) SECRET DISCLOSURE ATTACK

Given M_4, M_8 and M_9 are produced using one-way hash functions, the adversary cannot extract any information from them. M_4 and M_8 are also masked by ECC. Hence, the adversary cannot reveal any secret information from the transferred messages over public channel.

7) DESYNCHRONIZATION ATTACK

In RESEAP, the protocol participants do not update their shared secrets. Hence, the adversary cannot desynchronize them.

8) PROVIDING MESSAGE CONFIDENTIALITY

In RESEAP, the session key is computed as:

$$SK_S = h(M_5\|TS_S\|TS_U\|ID_U\|ID_S\|K_U)$$

where, $M_5 = sk_u.sk_S.r_S.r_U.g$. It is clear the adversary cannot extract the session key without solving EC-DLP or EC-CDHP.

9) SECURITY AGAINST COMPROMISED SMART CARD

Since the user's smart cart can be stolen, we should assume that the adversary has access to the SC and compromise it. With compromising SC, the adversary has access to its content and her/his aim could be to impersonate the user or to compromise the security of the previous session keys. The content of SC is $\{E_q(c,d), q, g, h(.), Es(.), ID_S, PK_S, a, PWU, K_U\}$. To impersonate the user or extract a previous session key, the adversary at least needs $sk_U = PUF(ID_U\|PW_U) + a$. However, the adversary has no access to the $PUF(.)$ function and by compromising the smart card $PUF(.)$ will not be reconstructable. In addition, the session key is calculated as $SK_S = h(sk_u.sk_S.r_S.r_U.g\|TS_S\|TS_U\|ID_U\|ID_S\|K_U)$ and the adversary has access to $M_2 = sk_U.r_U.g$ and $M_6 = sk_S.r_S.g$.

However, to compute $sk_u.sk_S.r_S.r_U.g$ using those information, the adversary should deal with EC-DLP or EC-CDHP which is not feasible in polynomial time. Hence, RESEAP is secure against attacks related to compromised smart card.

10) INSIDER ATTACK

An insider adversary has access to the content of the SC sent by S, i.e. $\{E_q(c, d), q, g, h(.), ID_S, PK_S\}$, and the transferred messages from U, during the registration phase or later, i.e. ID_U, PWU, M_2, M_4, M_9 and also the extracted $TS_U\|ID_S\|ID_U\|M_3$. However, to extract the user's password, it should overcome the embedded $PUF(.)$ which is not feasible. Hence, the proposed protocol is secure against insider attack. It is worth nothing that even having sk_S does not help the adversary to recover the user's password in polynomial time.

11) PASSWORD GUESSING

If the insider attacker cannot guess the user's password, then any other adversaries, which have less access, will not be able to recover the user's password.

B. FORMAL SECURITY ANALYSIS IN RoR MODEL

In this section, acco5rding to [20], [21], we formally prove the semantic security of RESEAP in real or random model (RoR).

Theorem 1: Let PUF and $h(.)$ be a secure and reliable PUF and a secure hash function respectively and q_{exe}, q_{send} and q_{test} represent the number of queries to Execute, Send and Test oracles on RESEAP, respectively. Then:

$$Adv^{RoR}_{\mathcal{D},RESEAP}(t; q_{exe}; q_{test}; q_{send}) \leq 4.q.\varepsilon_h + 4.q.\varepsilon_{ECC} + q.\varepsilon_{PUF}$$

where ε_{ECC} denotes the maximum advantage of solving EC-DLP or EC-CDHP by the adversary on each query, ε_h denotes the maximum advantage of contradicting collision resistance security of $h(.)$ and ε_{PUF} denotes the maximum advantage of contradicting the indistinguishability property of PUF on each query and $q = q_{exe} + q_{test} + q_{send}$.

Proof: We are assuming U is communicating with S to share a session key and the adversary $\mathcal{A}$ is an adversary against the semantic security of RESEAP in the Real-or-Random model. Similar to [21], we define a series of games $\mathcal{G}$, starting with random world *Ran* and ending with real world RESEAP, denoted as *Real*. On each game $\mathcal{G}_n$, we will determine $Adv^{RoR-\mathcal{G}_n}_{\mathcal{D},\mathcal{P}}(t, R)$ as the $\mathcal{A}$'s advantage to guess the hidden bit b involved in the Test queries of RoR model. It should be noted that in order to rule out trivial advantages, through the proof, we assume that the structure of the transferred messages, e.g. the block size, in *Ran* and *Real* worlds are identical.

Game $\mathcal{G}_0$. This game exactly defines *Ran* and $Adv^{RoR-\mathcal{G}0}_{\mathcal{D},Ran}(t, R) = 0$.

Game $\mathcal{G}_1$. It is similar to $\mathcal{G}_0$. The only difference is that M_2 and M_6 are replaced with ECC point multiplications. Given $M_2 = (PUF(ID_U\|PW_U) + a).r_U.g$, $M_6 = sk_s.r_S.g$ and r_U and r_S are fresh nonces that are generated by U and S respectively. hence, to distinguish this modification, the adversary should deal with solving EC-DLP or EC-CDHP which is expected to be hard. Following this argument (?):

$$Adv^{RoR-\mathcal{G}_1}_{\mathcal{D},Ran}(t, R) \leq Adv^{RoR-\mathcal{G}_0}_{\mathcal{D},Ran}(t, R) + 2.q.\varepsilon_{ECC}$$

where $q = q_{exe} + q_{send} + q_{test}$.

Game $\mathcal{G}_2$. In this game, M_4 and M_8 are respectively replaced by $(TS_U\|ID_S\|ID_U\|M_3) \oplus M_1$ and $M_8 = (TS_S\|ID_U\|ID_S\|M_7) \oplus M_5$ but yet M_1 and M_5 are random strings. It is clear this modification has no affect on the adversary's advantage as long as M_1 is not distinguishable from a random string. Hence, $Adv^{RoR-\mathcal{G}_2}_{\mathcal{D},Ran}(t, R) - Adv^{RoR-\mathcal{G}_1}_{\mathcal{D},Ran}(t, R) = 0$.

Game $\mathcal{G}_3$. This game is similar to $\mathcal{G}_2$. The only difference is that M_1 and M_5 are calculated using ECC as $M_1 = (PUF(ID_U\|PW_U) + a).r_U.PK_S$ and $M_5 = r_S.M_1^*$ respectively. However, the adversary has already access to $M_2 = (PUF(ID_U\|PW_U) + a).r_U.g$ and $M_6 = sk_s.r_S.g$ from $\mathcal{G}_1$ and if it can solve EC-DLP or EC-CDHP then it should be able to distinguish M_4 or M_8 from a random string which means that M_4 or M_8 leak information. Therefore:

$$Adv^{RoR-\mathcal{G}_3}_{\mathcal{D},Ran}(t, R) \leq Adv^{RoR-\mathcal{G}_2}_{\mathcal{D},Ran}(t, R) + 2.q.\varepsilon_{ECC}$$

Game $\mathcal{G}_4$. In this game the values that are calculated by hash function in *Real* are computed by $h(.)$. More precisely, we compute $M_3 = h(M_1\|ID_S\|ID_U\|TS_U\|K_U)$, $M_7 = h(M_5\|ID_U\|ID_S\|TS_S\|K_U)$ and $M_9 = h(Sk_U\|M_7^*\|TS_S)$. They are all randomized by timestamp and implicitly by the random values that are generated in each session. Hence, to distinguish them from random strings, the adversary should deal with the hash function's security. Therefore:

$$Adv^{RoR-\mathcal{G}_4}_{\mathcal{D},Ran}(t, R) \leq Adv^{RoR-\mathcal{G}_3}_{\mathcal{D},Ran}(t, R) + 3.q.\varepsilon_h.$$

Game $\mathcal{G}_5$. Compared to $\mathcal{G}_4$, in this game we use $PUF(.)$ to compute sk_U. Hence:

$$Adv^{RoR-\mathcal{G}_5}_{\mathcal{D},Ran}(t, R) \leq Adv^{RoR-\mathcal{G}_4}_{\mathcal{D},Ran}(t, R) + q.\varepsilon_{PUF}.$$

Game $\mathcal{G}_6$. This game is similar to $\mathcal{G}_5$. The only difference is that the session key is calculated using hash function as $h(M_5\|TS_S^*\|TS_U\|ID_U\|ID_S\|K_U)$. However, if the input string to the hash function is randomized by timestamp and implicitly by the random values that are generated in each session, then:

$$Adv^{RoR-\mathcal{G}_6}_{\mathcal{D},Ran}(t, R) \leq Adv^{RoR-\mathcal{G}_5}_{\mathcal{D},Ran}(t, R) + q.\varepsilon_h$$

where $q' = q_{exe} + q_{send} + q_{test}$. It is clear that $\mathcal{G}_6$ exactly represents the implementation of RESEAP and we can conclude that:

$$\begin{aligned} &Adv^{RoR}_{\mathcal{D},RESEAP}(t; q_{exe}; q_{test}; q_{send}) \\ &= Adv^{RoR}_{\mathcal{D},RESEAP}(t, R) - Adv^{RoR}_{\mathcal{D},Ran}(t, R) \\ &= Adv^{RoR-\mathcal{G}_6}_{\mathcal{D},Ran}(t, R) - Adv^{RoR-\mathcal{G}_0}_{\mathcal{D},Ran}(t, R) \\ &\leq 4.q.\varepsilon_h + 4.q.\varepsilon_{ECC} + q.\varepsilon_{PUF} \\ &= 4.q.\varepsilon_h + q.\varepsilon_{ES} + 2.q.\varepsilon_{ECC} \end{aligned}$$

Hence, the proof has been completed in this way. ∎ ■

VI. PERFORMANCE EVALUATION

A. SIMULATION METRICS

To provide a fair security comparison between RESEAP and other related protocols, through our analysis, bit lengths of timestamp, random number, identifier, hash value and ECC point are respectively considered as 32, 128, 128, 160 and 320 bits. It should be noted that we are considering SHA-256 as one-way hash function but truncating its output to 160-bit. The reason is the recently reported security concerns of SHA-1 [25].

To compute the energy consumption of a scheme, we use the known relation $Ec = V.I.TC$, in which Ec is the consumed energy during the computation time TC, assuming I is the consumed current and V is the working voltage of the device.

B. RESULTS

The user side of ESEAP requires computational modules for ECC, hash function and symmetric encryption/decryption. On the other hand, in RESEAP, the user does not need symmetric encryption/decryption but it should support PUF. Given PUF is a hardware friendly, we can assume required hardware resources for RESEAP and ESEAP are almost similar. However, the server of ESEAP requires computational modules for ECC, hash function and symmetric encryption/decryption while the server of RESEAP only needs ECC and hash function. On the other hand, the user of ESEAP performs 12 calls to the hash function (T_h), 2 calls to symmetric cipher (T_{Es}) and 3 calls to ECC point-multiplications (T_{ECC}) and its server performs $9T_h + 2T_{Es} + 3T_{ECC}$. The user of RESEAP performs a PUF invocation (T_{PUF}), 3 calls to ECC point-multiplications (T_{ECC}), 5 calls to the hash function and 3 calls to ECC point-multiplications and its server performs $5T_h + 3T_{ECC}$. This comparison shows that RESEAP is more efficient compared to ESEAP, in terms of computational complexity. A comparison between computation complexity of RESEAP and related protocols are presented in Table 3. Through experimental evaluation, we used an Arduino UNO R3 board having microcontroller ATmega328P for each client. Using the mentioned platform, we achieved $T_{ECC} \approx 21\ ms$, $T_{ECC} \approx 26\ ms$, $T_h \approx 3\ ms$ for SHA-256 and $T_{Es} = 3.7\ ms$. We also considered the time of a PUF invocation (T_{PUFn}) equal to T_h. Following this experiment, the execution time in the user side for ESEAP and RESEAP are respectively 106.4 *ms* and 81 *ms*. The execution time in the server side for ESEAP and RESEAP are respectively 97.4 *ms* and 78 *ms*. It shows that RESEAP is much faster than ESEAP in this platform, almost 24% in the user side and almost 20% in the server side and 22% for whole session. It should be noted that in ESEAP each call to the symmetric encryption/decryption requires more than one call to the block cipher, because the input message is much larger than the block length. If we consider it, the computational cost of ESEAP even will be higher.

Based on our parameter setting that are given in subsection VI-A, the communication cost of RESEAP for M_2, M_4, M_6 and M_8 will be 320 bits and the cost of M_9 will be 160 bits. On the other hand, the communication cost of ESEAP for P_3 and E_1 are 64 bits and 928 bits respectively and its cost for T_S and E_2 are 32 bits and 704 bits respectively. It shows that the communication cost of RESEAP is less than ESEAP. It is worth noting that E_1 and E_2 are calculated using symmetric encryption and their length should be at least the same as the encrypted values. Hence, the bit-length of E_1 and E_2 are considered equal to the bit length of their input values. Based on Table 3 and also Figure 3, RESEAP requires less cost than ESEAP in different aspects.

TABLE 3. Comparison of Cost for RESEAP and ESEAP.

	Time (ms)		Communication (bit)		Energy (mJ)	
Protocol	U	S	U	S	U	S
ESEAP	106.4	97	992	736	8.2	6.2
RESEAP	81	78	640	640	7.5	6

According to ATmega328P datasheet [26], the maximum power, i.e. $(V.I)$, of ATmega328P is less than $14mA \times 5.5\ V = 77\ mW$. Following this, the comparison of energy consumption of RESEAP and ESEAP is provided in Table 3.

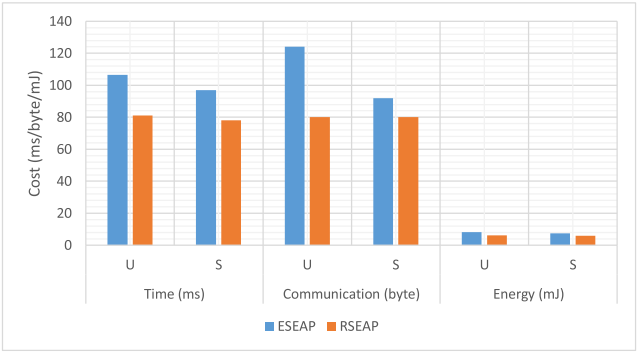

FIGURE 3. Comparison of overhear for RESEAP and ESEAP.

VII. CONCLUSION

In this paper, we provided the first third party security analysis of ESEAP, an ECC based mutual authentication protocol using smart card which has been proposed by Kumari *et al.*. Our detailed security analysis of this protocol demonstrated several crucial flaws on it, including vulnerability to off-line password guessing attack, traceability attack, and desynchronization attack. Besides, as a revised version, we proposed RESEAP which provides better efficiency compared to ESEAP and has provable semantic security in RoR model.

In ESEAP, to overcome off-line password guessing attack, the designers stored the hash image of the password, randomized by a nonce. They claimed that nonce prevents mentioned types of attack even if the password is selected from a finite set. However, given the nonce is constant and stored in a permanent memory and assuming adversary has access to the hash image, then he/she has access to the nonce. Hence, this approach could not provide desired security. In this regard,

future works could be investigating our strategy versus other protocols that may have used the approach. To overcome this problem, we have used PUF which adversary cannot reconstruct it.

REFERENCES

[1] W. Chou, "Inside SSL: The secure sockets layer protocol," *IT Prof.*, vol. 4, no. 4, pp. 47–52, 2002.

[2] A. Ometov, S. Bezzateev, N. Mäkitalo, S. Andreev, T. Mikkonen, and Y. Koucheryavy, "Multi-factor authentication: A survey," *Cryptography*, vol. 2, no. 1, p. 1, Jan. 2018.

[3] I. Ul Haq, J. Wang, and Y. Zhu, "Secure two-factor lightweight authentication protocol using self-certified public key cryptography for multi-server 5G networks," *J. Netw. Comput. Appl.*, vol. 161, Jul. 2020, Art. no. 102660.

[4] B. Ying and A. Nayak, "Lightweight remote user authentication protocol for multi-server 5G networks using self-certified public key cryptography," *J. Netw. Comput. Appl.*, vol. 131, pp. 66–74, Apr. 2019.

[5] F. Sinigaglia, R. Carbone, G. Costa, and N. Zannone, "A survey on multi-factor authentication for online banking in the wild," *Comput. Secur.*, vol. 95, Aug. 2020, Art. no. 101745.

[6] S. Qiu, G. Xu, H. Ahmad, G. Xu, X. Qiu, and H. Xu, "An improved lightweight two-factor authentication and key agreement protocol with dynamic identity based on elliptic curve cryptography," *KSII Trans. Internet Inf. Syst.*, vol. 13, no. 2, pp. 978–1002, 2019.

[7] M. Nikooghadam, R. Jahantigh, and H. Arshad, "A lightweight authentication and key agreement protocol preserving user anonymity," *Multimedia Tools Appl.*, vol. 76, no. 11, pp. 13401–13423, Jun. 2017.

[8] S. Kumari, M. K. Khan, and X. Li, "An improved remote user authentication scheme with key agreement," *Comput. Electr. Eng.*, vol. 40, no. 6, pp. 1997–2012, Aug. 2014.

[9] S. D. Kaul and A. K. Awasthi, "Security enhancement of an improved remote user authentication scheme with key agreement," *Wireless Pers. Commun.*, vol. 89, no. 2, pp. 621–637, Jul. 2016.

[10] F. Wu, L. Xu, S. Kumari, X. Li, M. K. Khan, and A. K. Das, "An enhanced mutual authentication and key agreement scheme for mobile user roaming service in global mobility networks," *Ann. des Télécommun.*, vol. 72, nos. 3–4, pp. 131–144, Apr. 2017.

[11] M. Gupta and N. S. Chaudhari, "Anonymous two factor authentication protocol for roaming service in global mobility network with security beyond traditional limit," *Ad Hoc Netw.*, vol. 84, pp. 56–67, Mar. 2019.

[12] Q. Xie, D. S. Wong, G. Wang, X. Tan, K. Chen, and L. Fang, "Provably secure dynamic ID-based anonymous two-factor authenticated key exchange protocol with extended security model," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 6, pp. 1382–1392, Jun. 2017.

[13] X. Li, D. Yang, X. Zeng, B. Chen, and Y. Zhang, "Comments on 'Provably secure dynamic id-based anonymous two-factor authenticated key exchange protocol with extended security model,'" *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 12, pp. 3344–3345, Dec. 2019.

[14] M. Fotouhi, M. Bayat, A. K. Das, H. A. N. Far, S. M. Pournaghi, and M. A. Doostari, "A lightweight and secure two-factor authentication scheme for wireless body area networks in health-care IoT," *Comput. Netw.*, vol. 177, Aug. 2020, Art. no. 107333.

[15] J. W. Byun and I. R. Jeong, "Comments on physically unclonable function based two-factor authentication protocols," *Wireless Pers. Commun.*, vol. 106, no. 3, pp. 1243–1252, Jun. 2019.

[16] A. C. D. Resende, K. Mochetti, and D. F. Aranha, "PUF-based mutual multifactor entity and transaction authentication for secure banking," in *Proc. 4th Int. Workshop Lightweight Cryptogr. Secur. Privacy LightSec* in Lecture Notes in Computer Science, vol. 9542, T. Güneysu, G. Leander, and A. Moradi, Eds. Bochum, Germany: Springer, Sep. 2015, pp. 77–96.

[17] K. B. Frikken, M. Blanton, and M. J. Atallah, "Robust authentication using physically unclonable functions," in *Proc. 12th Int. Conf. Inf. Secur. (ISC)*, in Lecture Notes in Computer Science, vol. 5735, P. Samarati, M. Yung, F. Martinelli, and C. A. Ardagna, Eds. Pisa, Italy: Springer, Sep. 2009, pp. 262–277.

[18] A. Kumari, S. Jangirala, M. Y. Abbasi, V. Kumar, and M. Alam, "ESEAP: ECC based secure and efficient mutual authentication protocol using smart card," *J. Inf. Secur. Appl.*, vol. 51, Apr. 2020, Art. no. 102443.

[19] C. Wang, D. Wang, G. Xu, and Y. Guo, "A lightweight password-based authentication protocol using smart card," *Int. J. Commun. Syst.*, vol. 30, no. 16, p. e3336, Nov. 2017.

[20] M. Abdalla, P. Fouque, and D. Pointcheval, "Password-based authenticated key exchange in the three-party setting," in *Proc. 8th Int. Workshop Theory Pract. Public Key Cryptogr. Public Key Cryptogr.*, vol. 3386, S. Vaudenay, Ed. Les Diablerets, Switzerland: Springer, Jan. 2005, pp. 65–84.

[21] M. Hosseinzadeh, O. H. Ahmed, S. H. Ahmed, C. Trinh, N. Bagheri, S. Kumari, J. Lansky, and B. Huynh, "An enhanced authentication protocol for RFID systems," *IEEE Access*, vol. 8, pp. 126977–126987, 2020.

[22] M. S. Alkatheiri, A. R. Sangi, and S. Anamalamudi, "Physical unclonable function (PUF)-based security in Internet of Things (IoT): Key challenges and solutions," in *Handbook of Computer Networks and Cyber Security, Principles and Paradigms*, B. B. Gupta, G. M. Pérez, D. P. Agrawal, and D. Gupta, Eds. Cham, Switzerland: Springer, 2020, pp. 461–473.

[23] K.-U. Choi, S. Baek, J. Heo, and J.-P. Hong, "A 100% stable sense-amplifier-based physically unclonable function with individually embedded non-volatile memory," *IEEE Access*, vol. 8, pp. 21857–21865, 2020.

[24] D. Jeon, J. H. Baek, Y.-D. Kim, J. Lee, D. K. Kim, and B.-D. Choi, "A physical unclonable function with bit error rate $< 2.3 \times 10^{-8}$ based on contact formation probability without error correction code," *IEEE J. Solid-State Circuits*, vol. 55, no. 3, pp. 805–816, Mar. 2020.

[25] G. Leurent and T. Peyrin, "From collisions to chosen-prefix collisions application to full SHA-1," in *Proc. 38th Annu. Int. Conf. Theory Appl. Cryptograph. Techn. Adv. Cryptol. EUROCRYPT*, in Lecture Notes in Computer Science, vol. 11478, Y. Ishai and V. Rijmen, Eds. Darmstadt, Germany: Springer, May 2019, pp. 527–555.

[26] Atmel. *8-Bit AVR Microcontroller With 32k BYTES in-System Programmable Flash*. Accessed: Jun. 10, 2020. [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf

MASOUMEH SAFKHANI received the Ph.D. degree in electrical engineering from the Iran University of Science and Technology, in 2012, with the security analysis of RFID protocols as her major field. She is currently an Assistant Professor with the Computer Engineering Department, Shahid Rajaee Teacher Training University, Tehran, Iran. She is the author/coauthor of over 50 technical articles in information security and cryptology in major international journals and conferences. Her current research interests include the security analysis of lightweight and ultra-lightweight protocols, targeting constrained environments, such as RFID, the IoT, VANET, and WSN.

NASOUR BAGHERI received the M.S. and Ph.D. degrees in electrical engineering from the Iran University of Science and Technology (IUST), Tehran, Iran, in 2002 and 2010, respectively. He is currently an Associate Professor with the Electrical Engineering Department, Shahid Rajaee Teacher Training University, Tehran. He is also a part-time Researcher with the Institute for Research in Fundamental Sciences. He is the author of more than 100 articles in information security and cryptology. His research interests include cryptology, more precisely, designing and analysis of symmetric schemes, such as lightweight ciphers, e.g., block ciphers, hash functions, and authenticated encryption schemes, cryptographic protocols for constrained environment, such as RFID tags and the IoT edge devices, and hardware security, e.g., the security of symmetric schemes against side-channel attacks, such as fault injection and power analysis.

SARU KUMARI received the Ph.D. degree in mathematics from Chaudhary Charan Singh University, Meerut, India, in 2012. She is currently an Assistant Professor with the Department of Mathematics, Chaudhary Charan Singh University. She has published more than 133 research articles in reputed international journals and conferences, including 115 publications in SCI-indexed journals. Her current research interests include information security and applied cryptography. She is a Technical Program Committee member for many international conferences. She has served as a Lead/Guest Editor of four special issues in SCI journals of Elsevier, Springer, and Wiley. She is on the Editorial Board for more than 12 journals of international repute, including seven SCI journals.

HAMIDREZA TAVAKOLI received the M.S. and Ph.D. degrees in electrical engineering from the Iran University of Science and Technology (IUST), Tehran, Iran, in 2002 and 2012, respectively. He is currently a Professor of Electrical Engineering with Hakim Sabzevari University, Sabzevar, Iran. His research interests include performance evaluation of wireless networks and network security.

SACHIN KUMAR (Member, IEEE) received the Ph.D. degree in computer science from CCS University, Meerut, in 2007. He has been working as a Professor with the Department of Computer Science and Engineering, Ajay Kumar Garg Engineering College (AKGEC), Ghaziabad, since October 2011. Prior to joining AKGEC, he worked with the Raj Kumar Goel Institute of Technology (RKGIT) Ghaziabad, the Krishna Institute of Engineering Technology (KIET), Ghaziabad, and CCS University, Meerut. He has more than 18 years of academic experience. He has guided four Ph.D. and ten M.Tech. students. He has published/presented several articles in journals/conferences of repute. He is the author/coauthor of three books of computer science.

JIAHUI CHEN (Member, IEEE) received the B.S. degree from South China Normal University, China, in 2009, and the M.S. and Ph.D. degrees from South China University of Technology, China, in 2012 and 2016, respectively. He joined the National University of Singapore, as a Research Scientist, from March 2017 to May 2018. He is currently an Associate Professor with the School of Computers, Guangdong University of Technology. His research interests include public key cryptography, post-quantum cryptography, and network security.