


A secure authentication scheme based on elliptic curve cryptography for IoT and cloud servers

Saru Kumari¹ · Marimuthu Karuppiah² ·
Ashok Kumar Das³  · Xiong Li⁴ · Fan Wu⁵ ·
Neeraj Kumar⁶

Published online: 11 April 2017
© Springer Science+Business Media New York 2017

Abstract The Internet of Things (IoT) is now a buzzword for Internet connectivity which extends to embedded devices, sensors and other objects connected to the Internet. Rapid development of this technology has led to the usage of various embedded devices in our daily life. However, for resource sharing and communication among these devices, there is a requirement for connecting these embedded devices to a large

✉ Ashok Kumar Das
iitkgp.akdas@gmail.com; ashok.das@iiit.ac.in

Saru Kumari
saryusiiohi@gmail.com; saru@ccsuniversity.ac.in

Marimuthu Karuppiah
marimuthume@gmail.com; k.marimuthu@vit.ac.in

Xiong Li
lixiongzhu@163.com

Fan Wu
conjurer1981@gmail.com

Neeraj Kumar
neeraj.kumar@thapar.edu

- ¹ Department of Mathematics, Ch. Charan Singh University, Meerut, Uttar Pradesh 250 005, India
- ² School of Computer Science and Engineering, VIT University, Vellore, Tamilnadu 632 014, India
- ³ Center for Security, Theory and Algorithmic Research, International Institute of Information Technology, Hyderabad 500 032, India
- ⁴ School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan 411201, China
- ⁵ Department of Computer Science and Engineering, Xiamen Institute of Technology, Xiamen 361021, China
- ⁶ Department of Computer Science and Engineering, Thapar University, Patiala 147 004, India

pool of resources like a cloud. The promising applications of IoT in Government and commercial sectors are possible by integrating cloud servers with these embedded devices. But such an integration of technologies involves security issues like data privacy and authentication of devices whenever information is exchanged between them. Recently, Kalra and Sood proposed an authentication scheme based on elliptic curve cryptography (ECC) for IoT and cloud servers and claimed that their scheme satisfies all security requirements and is immune to various types of attacks. However, in this paper, we show that Kalra and Sood scheme is susceptible to offline password guessing and insider attacks and it does not achieve device anonymity, session key agreement, and mutual authentication. Keeping in view of the shortcomings of Kalra and Sood's scheme, we have proposed an authentication scheme based on ECC for IoT and cloud servers. In the proposed scheme in this paper, we have formally analyzed the security properties of the designed scheme by the most widely accepted and used Automated Validation of Internet Security Protocols and Applications tool. Security and performance analysis show that when compared with other related schemes, the proposed scheme is more powerful, efficient, and secure with respect to various known attacks.

Keywords Authentication · Embedded device · Internet of Things · Cloud server · Cookies · Security

1 Introduction

An integrated system comprising of computer software, hardware as well as mechanical components and having task-specific processing ability is called as an embedded device. Embedded devices have evolved from a single microcontroller chip having limited abilities to intelligent systems having enhanced connectivity, processors, and operating systems. Such smart systems can be deployed by enterprises to create inter-related and efficient systems that are able to communicate, collect and analyze data.

Nowadays, the embedded device can act as a HTTP (Hyper Text Transfer Protocol) client. For that reason, the embedded device must be configured with TCP/IP protocol stack. HTTP is a request–response protocol in the client/server communication model. In general, all personal computers require the assistance of the HTTP protocol. This status is also appropriate for embedded devices. In the present market, the majorities of the embedded devices are configured with a Web browser and can act as HTTP clients. Such embedded devices are deployed in the field without the need of any user interface by various specialized softwares. This scenario is called as the Internet of Things (IoT). IoT is formally defined as the interconnection of distinctively identifiable embedded computing devices (i.e., things) in the prevailing Internet organization. Basically, IoT involves the connection of various embedded devices to the Internet. IoT includes real world and small things having limited storage and processing capabilities, and subsequent issues like reliability, security, privacy, and performance.

These embedded devices are widely used across the globe for data sharing and communication. The data generated from these devices need to be analyzed using one of the most popular infrastructures called as cloud computing. It possesses virtually infinite storage and processing power and has partially resolved most of the IoT issues. Hence,

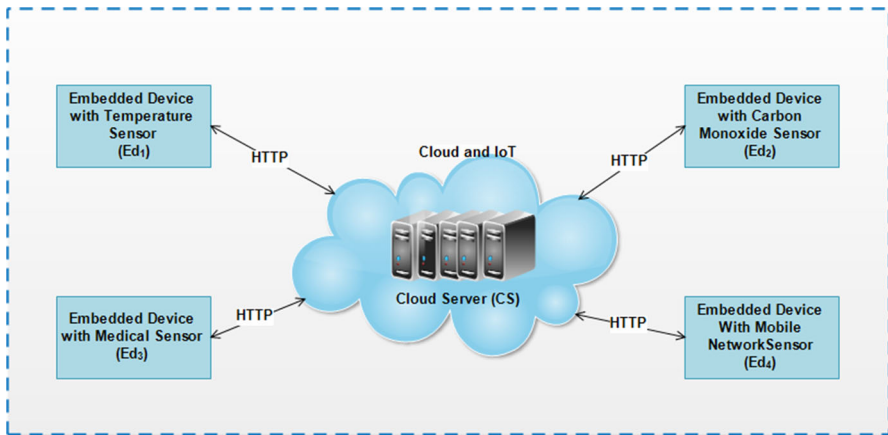


Fig. 1 Cloud-connected embedded devices

a novel IT paradigm where Cloud and IoT, two complementary technologies integrated together as CloudIoT, is expected to bring about an unprecedented growth in current and future Internet [1–3]. CloudIoT solves most problems [4–6], and also provides added features like ease-of-access, ease-of-use, and decreased deployment costs [5]. Security, a concern in all networked environments, is a major problem for CloudIoT as well. Both the IoT (i.e., WSN, RFID) and Cloud sides are susceptible to a number of attacks. In IoT environment, encryption guarantees data confidentiality, authenticity, and integrity. However, it is not able to address insider attacks (e.g., during WSN reprogramming) and is difficult to implement on devices that are computation-constrained.

In cloud IoT environment, with the assistance of specialized software, the embedded devices act as HTTP clients and can communicate with HTTP-enabled cloud server (HTTP server). This implies that machine-to-machine communication is feasible (as shown in Fig. 1). Embedded devices can extract a large amount of data storage and computational facilities from cloud computing infrastructure. Based on the processes in the cloud, networked embedded devices are simply not restrained to their local resources. However, security is a major issue while connecting to a cloud for using resources [7, 8]. Embedded devices (HTTP client) and HTTP-enabled cloud server (HTTP server) should be mutually authenticated before accessing cloud services. In situations where processing power and memory are limited, it is appropriate to use elliptic curve cryptography (ECC), a type of public key cryptography [9, 10].

An authentication plays a pivotal role when embedded devices and cloud computing services are integrated together. Several authentication schemes [11–22] based on ECC have been proposed in the past. Such authentication schemes are appropriate for smart devices proposed by Wu et al. [11], Tian et al. [12], Abichar et al. [13]. But these schemes suffer from several shortcomings. For example, the scheme of Wu et al. only allows the server to authenticate the users. But, this scheme is insecure as an attacker can disguise as a server and extract user information. In contrast, the schemes proposed by Tian et al. [12] and Abichar et al. [13] offer mutual authentication using certificates. Certification schemes increase the cost as extra computations need to be performed by

the server and users to verify each other's identity. Authentication schemes for smart devices based on ECC have also been presented by Yang et al. [14], Hafizul et al. [15] and Debiao et al. [16], Ray et al. [17], Granjal et al. [18], Jiang et al. [19]. The proposed schemes, based on various types of ECC, involve concepts extending from time stamps to certificate-based mutual authentication. An attribute-based light weight encryption scheme for IoT based on ECC has been presented by Yao et al. [20].

1.1 Our contributions

Very recently, in 2015, Kalra and Sood [23] proposed an authentication scheme to connect embedded devices to the cloud server using ECC for IoT and claimed their scheme achieves all security requirements and is resistant to various attacks. But, in this paper, we show that Kalra and Sood's scheme is vulnerable to offline password guessing and insider attacks and that it fails to achieve device anonymity, session key agreement, and mutual authentication. In order to fix the shortcomings of Kalra and Sood's scheme, we have proposed an enhanced authentication scheme based on elliptic curve cryptography (ECC) for IoT and cloud servers which is proficient to mitigate various types of known attacks and achieves various security goals.

1.2 Organization of the paper

The rest of the paper is organized as follows. The preliminaries of elliptic curve cryptography (ECC) have been discussed in Sect. 2. A review of Kalra and Sood's scheme is discussed in Sect. 3. The weaknesses of Kalra and Sood scheme have been described in Sect. 4. The proposed scheme and its analysis have been presented in Sects. 5 and 6. The performance analysis and security requirement comparisons have been discussed in Sect. 7. Finally, conclusions have been presented in Sect. 8.

2 Preliminaries

This section provides brief introduction to the elliptic curve cryptography [24–26]. The security strength of the ECC based on the hardness of solving the elliptic curve discrete logarithm problem (ECDLP) and it can provide same level of security strength with smaller key size [27]. ECC with smaller key size leads to evident cost savings. ECC with smaller key size also facilitates the design of faster cryptographic operations that run on small chips with tiny memory. This is appropriate for resource-constrained systems since it reduce the power consumption and heat production. As a result, ECC is well suitable for smart devices that operate in resource constraint environments [28].

2.1 Theory of elliptic curve

The equation of the elliptic curve $E_p(a, b)$ over F_p ($p > 3$ and is a large prime number) is defined as $y^2 \bmod p \equiv (x^3 + ax + b) \bmod p$ where $(4a^3 + 27b^2) \bmod p \neq 0$; $x, y, a, b \in [0, p - 1]$. Any points $(x, y) \in E_p(a, b)$ are denoted as $E(F_p) = \{(x, y) :$

$x, y \in F_p$ satisfy $y^2 \equiv (x^3 + ax + b) \cup \{\mathcal{O}\}$ where \mathcal{O} is a point at infinity. The point multiplication is computed by repeated addition as $k \cdot P = \overbrace{P + P \dots P}^{k \text{ times}}$. The detailed elliptic curve group properties can be found in [24].

Definition 1 (*Elliptic Curve Discrete Logarithm Problem (ECDLP)*) Given P and Q in an elliptic curve $E_p(a, b)$, find an integer $k \in Z_p^*$ such that $Q = kP$.

3 Review of Kalra and Sood’s scheme

In this section, we review Kalra and Sood’s scheme [23] based on ECC for embedded devices which act as HTTP clients. They used the idea of HTTP cookies in their scheme. Kalra and Sood’s scheme comprises three phases as follows: the registration phase, the pre-computation and login phase and authentication phase. They are explained as follows and also in Fig. 2.

3.1 Notations

The list of notations used in this paper is described in Table 1.

3.2 Initialization

Initially, CS chooses an elliptic curve (EC) equation $y^2 = x^3 + ax + b$ over Z_p where $Z_p(p > 2^{160})$ is the finite field group. Then, two field elements are selected by CS, namely $a, b \in Z_p$, where a and b satisfy the condition $4a^3 + 27b^2 \pmod{p} \neq 0$. Let the EC base point be G which has a prime order of n ($n > 2^{160}$). Let \mathcal{O} be the point at infinity satisfying the equation $n \cdot G = \mathcal{O}$. A random nonce X_{cs} is selected by CS as its secret key.

3.3 Registration phase

To register with CS, Ed_i sends a distinct Id_i to CS. Upon receiving this request, CS generates a distinct password Pw_i for each device Ed_i .

Step R1. $Ed_i \rightarrow CS: Id_i$, CS generates Pw_i

CS selects a random number r_s for each Ed_i and computes a cookie $C_k = h(r_s || X_{cs} || E_t || Id_i)$ and stores the cookie on Ed_i as EC point $C'_k = C_k \cdot G$. Then, CS computes $T_i = r_s \oplus h(X_{cs})$, $A_i = h(r_s \oplus h(X_{cs}) \oplus Pw_i \oplus C'_k)$ and stores $T_i, A'_i = h(r_s \oplus h(X_{cs}) \oplus Pw_i \oplus C'_k) \cdot G$ along with Id_i of Ed_i in its database. The expiration time of the cookie E_t that corresponds to Id_i of Ed_i is stored by CS itself. As well, the expiration time is updated to E'_t and the cookie is restructured as $C_k = h(r_s || X_{cs} || E'_t || Id_i)$ when the cookie expires.

Step R2. $CS \rightarrow Ed_i: \text{Cookie } C'_k$

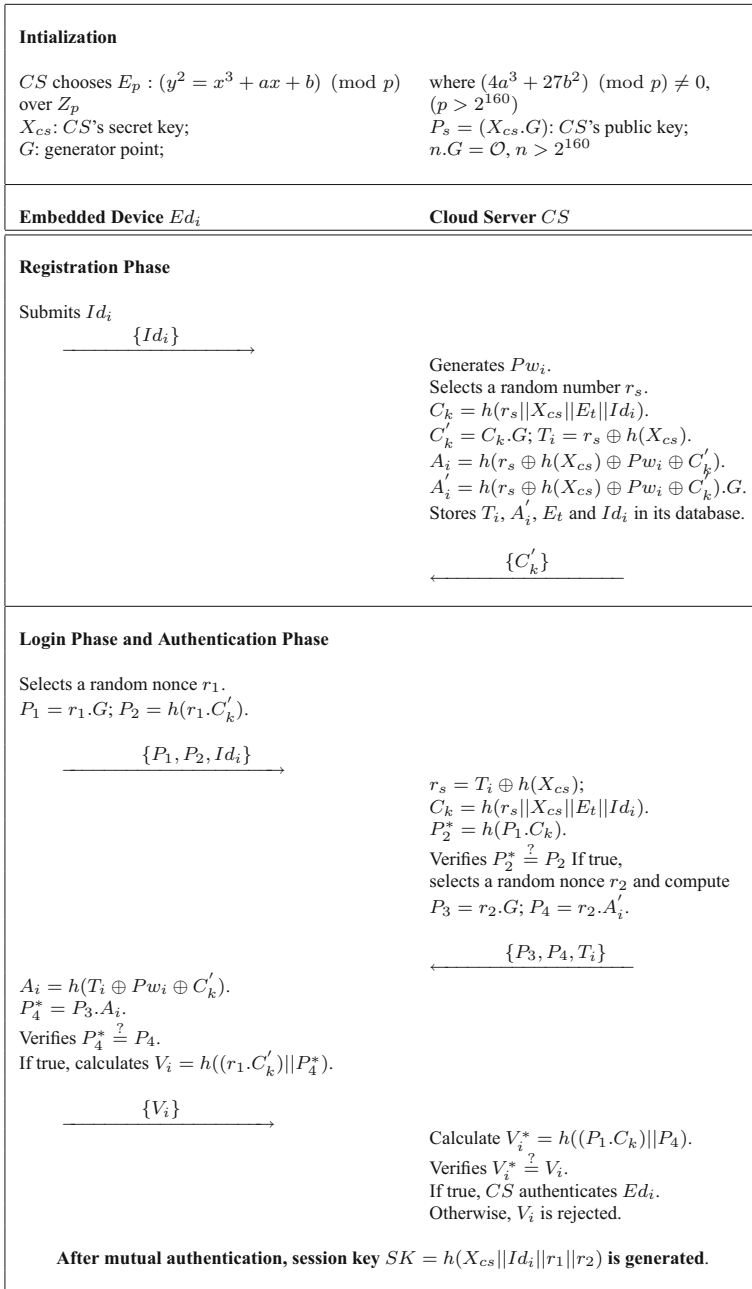


Fig. 2 Summary of the Kalra–Sood’s scheme

Table 1 Notations used in this paper

Notation	Description
Ed_i	Embedded device
Id_i	Identity of Ed_i
Pw_i	Password of Ed_i
CS	Cloud server
Id_{CS}	Identity of CS
X_{CS}	Secret key of CS based on ECC
Z_p	Finite field group
p	Large prime number of the order $>2^{160}$
r_1, r_2	Random numbers generated for ECC parameters
r_s	Random number generated by CS
G	Generator point of a large order n
C_k	Cookie information
E_t	Expiration time of the cookie
$h(\cdot)$	Cryptographic one-way hash function
\oplus	Bitwise XORed
\parallel	Concatenation

3.4 Pre-computation and login phase

Before each login, the Ed_i selects a random nonce r_1 , computes the ECC point $P_1 = r_1 \cdot G$ and then stores it in its memory.

Step L1. Ed_i **computes ECC point** P_1

To login with the cloud server(CS), Ed_i computes the ECC point $P_2 = h(r_1 \cdot C'_k)$ for login with CS and sends $\{P_1, P_2, Id_i\}$ to CS .

Step L2. $Ed_i \rightarrow CS: \{P_1, P_2, Id_i\}$

3.5 Authentication phase

Upon receiving the login request, CS computes cookie information $C_k = h(r_s \parallel X_{CS} \parallel E_t \parallel Id_i)$ by computing r_s from T_i using its secret key X_{CS} as $r_s = T_i \oplus h(X_{CS})$ and using Id_i of Ed_i , E_t and its secret key. Then, it computes the point $P_2^* = h(P_1 \cdot C_k)$ and verifies $P_2^* \stackrel{?}{=} P_2$. If the verification holds, then CS continues to the next step; Otherwise, CS rejects the login request of Ed_i .

Step A1. CS **verifies** $P_2^* \stackrel{?}{=} P_2$

CS selects a random nonce r_2 , computes the ECC point $P_3 = r_2 \cdot G$, $P_4 = r_2 \cdot A'_i$ and sends $\{P_3, P_4, T_i\}$ to Ed_i .

Step A2. $CS \rightarrow Ed_i: \{P_3, P_4, T_i\}$

Upon receiving $\{P_3, P_4, T_i\}$, Ed_i computes $A_i = h(T_i \oplus Pw_i \oplus C'_k)$ and the ECC point $P_4^* = P_3 \cdot A_i$. Then, it verifies $P_4^* \stackrel{?}{=} P_4$.

Step A3. Ed_i verifies $P_4^* \stackrel{?}{=} P_4$

Then, Ed_i computes $V_i = h((r_1 \cdot C'_k) || P_4^*)$ and sends V_i to CS . After receiving V_i , CS computes $V_i^* = h((P_1 \cdot C_k) || P_4)$ and verifies $V_i^* \stackrel{?}{=} V_i$ to authenticate Ed_i . If the verification holds, CS authenticates Ed_i ; Otherwise, V_i is rejected.

Step A4. CS verifies $V_i^* \stackrel{?}{=} V_i$

Subsequent to successful mutual authentication process between Ed_i and CS , both agree on a common session key $SK = h(X_{cs} || Id_i || r_1 || r_2)$. From then on all the subsequent messages transmitted between Ed_i and CS are XOR ed with the session key.

4 Cryptanalysis of Kalra and Sood’s scheme

In this section, we will discuss the security weaknesses of Kalra and Sood’s scheme. Through careful analysis, we find that Kalra and Sood’s scheme cannot achieve user anonymity, mutual authentication and session key agreement. Moreover, their scheme is vulnerable to offline password guessing and insider attacks. The details of these security weaknesses are described as follows.

4.1 Absence of device anonymity

In cloud environment, the leakage of device-specific information may facilitate an attacker to track the device’s login history and current location. Moreover, anonymity property makes authentication mechanism more strong as an attacker could not track which devices are interacting with the cloud server. A simple way to preserve anonymity is to conceal device’s valid identity throughout the communication. However, in Kalra and Sood’s scheme, Ed_i ’s identity (Id_i) is transmitted in plaintext through login request message $\{P_1, P_2, Id_i\}$. As a result, an attacker can know about the logging device by monitoring the login request message and device’s privacy is not preserved by the scheme. Thus, an attacker can use wrongly the readily available identity (Id_i) of Ed_i to break the security wall of the scheme. Therefore, Kalra and Sood’s scheme fails to preserve device anonymity.

4.2 Offline password guessing attack

Suppose an attacker obtain the cookie information C'_k from the embedded device Ed_i . Let us assume that an attacker intercepts the CS ’s response message as shown in Fig. 3. Through the intercepted message $\{P_3, P_4, T_i\}$ from the open channel, attacker can obtain Ed_i ’s password Pw_i as follows:

1. Guess the Ed_i ’s password as Pw_a .
2. Computes $A_i^* = h(T_i \oplus Pw_a \oplus C'_k)$.
3. Computes $P'_4 = P_3 \cdot A_i^*$.
4. Verifies the correctness of Pw_a by checking if $P'_4 \stackrel{?}{=} P_4$.

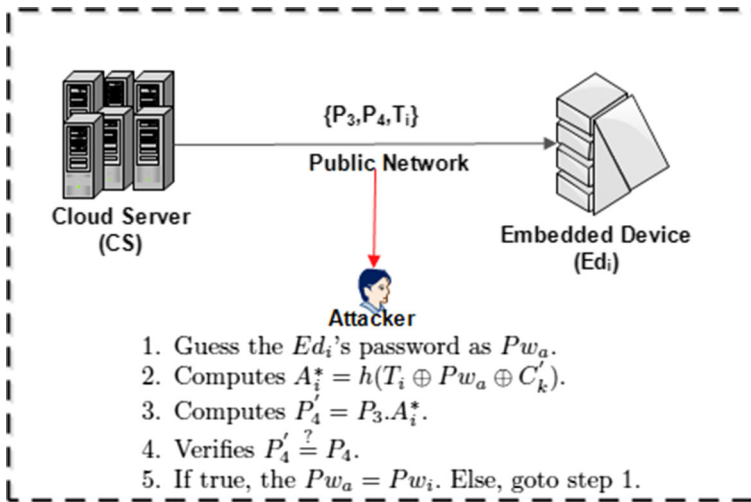


Fig. 3 Illustration of the offline password guessing attack

5. If the verification holds, considers Pw_a as the Ed_i 's password. Otherwise, attacker repeats Steps 1–4 until the exact password Pw_i is found.

The above discussion shows that Kalra and Sood's scheme is vulnerable to offline password guessing attack.

4.3 Insider attack

CS generates a distinct password (Pw_i) for each Ed_i during registration phase, subsequently Ed_i 's password (Pw_i) is known to the insider of CS who can misuse it. Therefore, Kalra and Sood's scheme is vulnerable to insider attack.

4.4 Stolen-verifier attack

In Kalra and Sood's scheme, during the registration phase of the Ed_i , cloud server CS stores T_i , A'_i and E_t with Id_i in its database. An attacker can steal this record from the database of CS and can use it to mount server impersonation attack. When Ed_i transmits the message $\{P_1, P_2, Id_i\}$ to CS, the attacker intercepts and blocks it from reaching to CS. The attacker obtains the values from the stolen record corresponding to the identity Id_i present in the message. Then the attacker selects a random nonce r_a , computes $P_{3a} = r_a \cdot G$, $P_{4a} = r_a \cdot A'_i$ and sends $\{P_{3a}, P_{4a}, T_i\}$ to Ed_i . On receiving this message, Ed_i would compute $P_{4a}^* = P_{3a} \cdot h(T_i \oplus Pw_i \oplus C'_k) = r_a \cdot G \cdot h(T_i \oplus Pw_i \oplus C'_k) = r_a \cdot h(T_i \oplus Pw_i \oplus C'_k) \cdot G = r_a \oplus A'_i = P_{4a}$, that is, $P_{4a}^* = P_{4a}$ meaning thereby that the response message sent by the attacker passes the verification test at Ed_i . Thus, Ed_i is confirmed that it is connected with the legal CS. Hence, the attacker is successful to impersonate as server using stolen verifiers of Ed_i .

4.5 Session key computation is not possible

In Kalra and Sood's scheme, the session key $SK = h(X_{cs} || Id_i || r_1 || r_2)$ is computed by both entities, embedded device Ed_i and the cloud server CS . Considering the computation of session key at the device end, we observe that Ed_i has identity Id_i and its chosen random nonce r_1 . However, Ed_i can neither obtain the random number r_2 from the received ECC point $P_3 = r_2 \cdot G$ nor it can know the value of the secret key X_{cs} of CS . On the other hand, CS has its secret key X_{cs} , its chosen number r_2 and the identity Id_i of Ed_i . However, CS cannot retrieve the random number r_1 from the received ECC point $P_1 = r_1 \cdot G$. Consequently, none of these entities can compute the session key, hence there is no session key agreement in the Kalra and Sood's scheme. For the reason that, in the Kalra and Sood's scheme, the forward secrecy and backward secrecy goals are missed.

4.6 Mutual authentication is not possible

In Step A2 of the authentication phase, Ed_i computes $A_i = h(T_i \oplus Pw_i \oplus C'_k)$. However, Ed_i is not capable to compute A_i since Pw_i is not known to Ed_i . In fact, during the registration phase, CS generates Pw_i after receiving Id_i from Ed_i . But the cloud server CS never provides the password Pw_i to Ed_i , hence the device Ed_i cannot compute A_i . In addition, Ed_i cannot compute P_4^* without the value of A_i and then no verification ($P_4^* \stackrel{?}{=} P_4$) is possible to confirm the legitimacy of CS . Thus, mutual authentication cannot be achieved properly in Kalra and Sood's scheme.

5 The proposed scheme

In this section, we describe various phases related to our proposed scheme. As pointed out in [29,30], we also assume that the tamper-resistant devices are available [31], where even the owner of the device cannot access the protected data stored in the device. This practical assumption is widely adopted in several applications, such as the Pay-TV systems [32,33]. Furthermore, since attacks on tamper-resistant devices require special equipment, it would be much costly than buying an embedded device, and the attacker cannot have economic incentives in order to mount such an attack [29]. Thus, with the use of a tamper-resistant embedded device, the security of the proposed scheme will be strong enough.

5.1 Initialization

Initialization phase is similar to that in Kalra and Sood's scheme.

5.2 Registration phase

This phase consists of the following steps:

Step R1. To register with CS , the tamper proof-embedded device Ed_i computes $I_i = h(Id_i || Pw_i)$ and sends I_i to CS through a secure communication channel.

Step R2. When the registration request received, CS generates a random number r_s and computes $PId_i = h(r_s || Id_{cs} || I_i) \oplus Id_{cs}$ for Ed_i and stores PId_i . Then, CS computes the cookie C_k and other security parameters as follows

$$\begin{aligned} C_k &= h(r_s || X_{cs} || E_t || PId_i) \\ C'_k &= C_k \cdot G \\ T_i &= r_s \oplus h(X_{cs} || PId_i) \\ A_i &= h(r_s \oplus h(X_{cs} || PId_i) \oplus I_i \oplus C'_k) \\ A'_i &= A_i \cdot G \end{aligned}$$

CS stores $t_i = T_i \oplus X_{cs}$, $a'_i = A'_i \oplus X_{cs}$ and $e_t = E_t \oplus X_{cs}$ corresponding to PId_i of Ed_i in its database. The expiration time of the cookie E_t that corresponds to I_i of Ed_i is stored by CS itself. Subsequently, CS delivers $\{PId_i, C'_k\}$ to Ed_i through a secure communication channel. After receiving $\{PId_i, C'_k\}$, the embedded device Ed_i stores PId_i and C'_k in its memory. The expiration time is updated to E'_t and the cookie is restructured as $C_k = h(r_s || X_{cs} || E'_t || PId_i)$ whenever the cookie expires.

5.3 Login and authentication phase

- Step LA1. Before each login, the Ed_i selects a random nonce r_1 , compute the ECC point $P_1 = r_1 \cdot G$ and $P_2 = h(r_1 \cdot C'_k)$. Then, it stores P_1 in its memory and sends the login request $\{P_1, P_2, PId_i\}$ to CS .
- Step LA2. Upon receiving the login request, CS obtains data corresponding to the received PId_i and computes $r_s = T_i \oplus h(X_{cs} || PId_i)$. Next, CS compute the cookie information $C_k = h(r_s || X_{cs} || E_t || PId_i)$ and $P_2^* = h(P_1 \cdot C_k)$. Then, it verifies $P_2^* \stackrel{?}{=} P_2$. If the verification holds, then CS continues to the next step; Otherwise, CS rejects the login request of Ed_i .
- Step LA3. CS selects a random nonce r_2 , computes the ECC point $P_3 = r_2 \cdot G$; $P_4 = r_2 \cdot A'_i$ and sends $\{P_3, P_4, T_i\}$ to Ed_i .
- Step LA4. Upon receiving $\{P_3, P_4, T_i\}$, Ed_i compute $A_i = h(T_i \oplus I_i \oplus C'_k)$ and the ECC point $P_4^* = P_3 \cdot A_i$. Then, it verifies $P_4^* \stackrel{?}{=} P_4$ to authenticate CS . If the verification holds, then Ed_i authenticates CS and continues the next step; Otherwise, Ed_i rejects the message $\{P_3, P_4, T_i\}$ of CS and resume the login.
- Step LA5. Ed_i computes the session key $SK = r_1 \cdot P_3 = r_1 \cdot r_2 \cdot G$ and sends $V_i = h((r_1 \cdot C'_k) || SK)$ to CS .
- Step LA6. After receiving V_i , CS compute the session key $SK^* = r_2 \cdot P_1 = r_2 \cdot r_1 \cdot G$ and $V_i^* = h((P_1 \cdot C_k) || SK^*)$. Then, CS verifies $V_i^* \stackrel{?}{=} V_i$ to authenticate Ed_i . If the verification holds, CS authenticates Ed_i ; Otherwise, V_i is rejected. From then on all the subsequent messages transmitted between Ed_i and CS are XOR ed with the session key $SK = r_1 \cdot P_3 = r_1 \cdot r_2 \cdot G = r_2 \cdot P_1 = SK^*$.

The summary of the login and authentication phase is given in Fig. 4.

6 Security analysis

6.1 Formal security verification using AVISPA tool

This section carries out the simulation of our proposed scheme using the most widely accepted and used Automated Validation of Internet Security Protocols and Applications (AVISPA) tool to show our scheme is secure against replay and man-in-the-middle attacks.

AVISPA is a modular and expressive formal language for specifying protocols and their security properties. It integrates different backends that implement a variety of state-of-the-art automatic analysis techniques [34]. It is a push-button tool for the automated validation of Internet security-sensitive protocols and applications, which becomes a widely accepted tool for our formal security verification in recent years [35–41]. AVISPA contains four backends: On-the-fly Model-Checker (OFMC), Constraint Logic-based Attack Searcher (CL-AtSe), SAT-based Model-Checker (SATMC) and Tree Automata based on Automatic Approximations for the Analysis of Security Protocols (TA4SP). The detailed descriptions of these backends are available in [34].

The security protocols need to be implemented in the HLPSL (High Level Protocols Specification Language), which is a role-oriented language [34], and contains the basic roles representing each participant role, and composition roles representing the scenarios of basic roles. An intruder is always represented by i and modeled using the Dolev–Yao model [42]. As a result, the intruder (i) is allowed to have a legitimate role in a protocol run. In HLPSL, a number of sessions, and a number of principals and some basic roles are defined. Using HLPSL2IF translator, HLPSL is converted to the intermediate format (IF), which then produces the output format (OF) with one of the four backends. OF has the following sections [43]:

- Summary indicates whether the tested protocol is safe, unsafe, or whether the analysis is inconclusive.
- Details either explains under what condition the tested protocol is declared safe, or what conditions have been used for finding an attack, or finally why the analysis was inconclusive.
- Protocol indicates the name of the protocol.
- Goal is the goal of the analysis.
- Backend tells the name of the backend used.
- Finally, after some comments and statistics, the trace of an attack (if any) is also printed in the standard Alice–Bob format.

Several basic types are supported in HLPSL and the details of these types are available in [34].

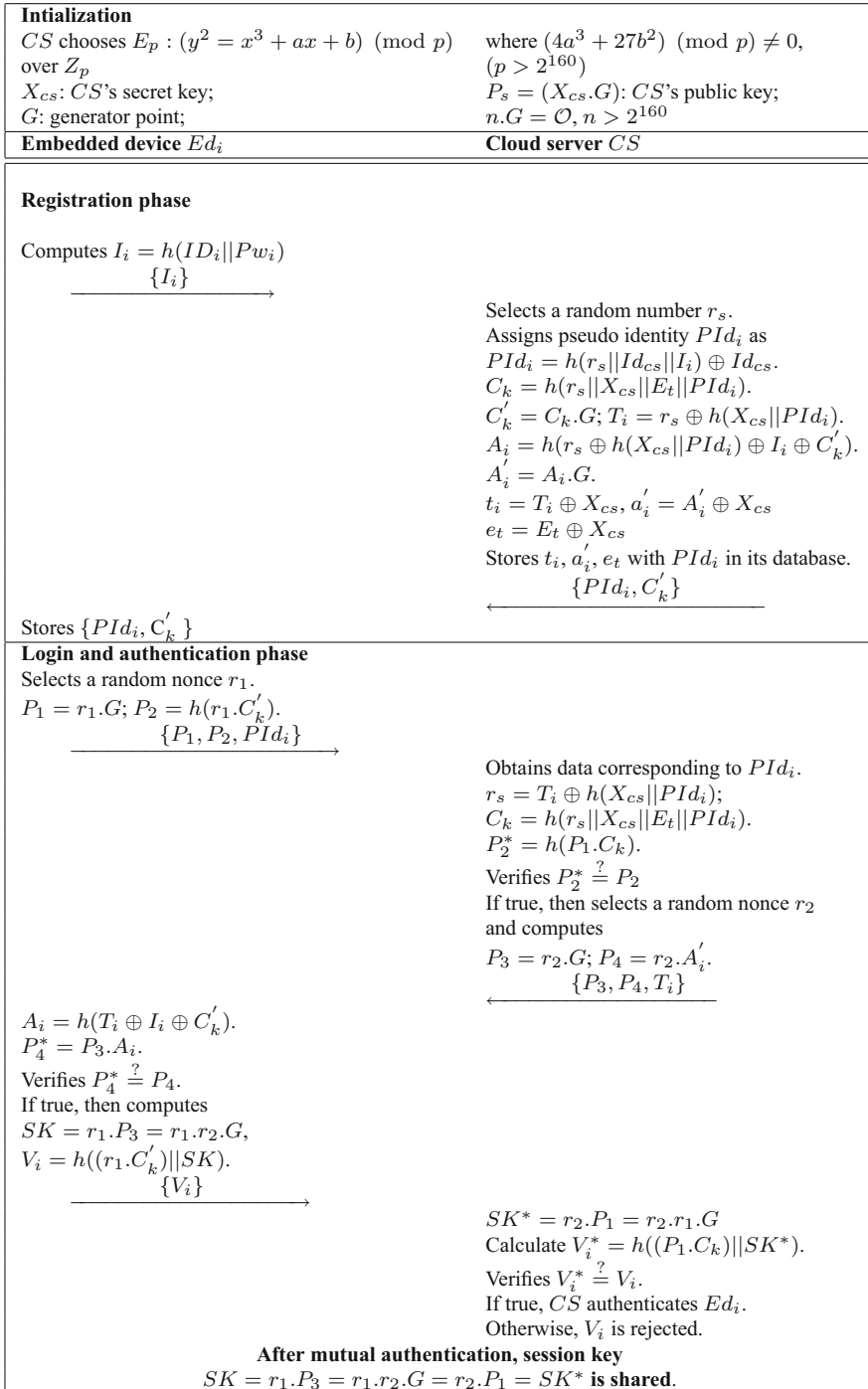


Fig. 4 Summary of the proposed scheme

6.2 Specification of the protocol

We have implemented two basic roles for the embedded device Ed_i and the cloud server CS in HLPSSL related to the registration phase and login and authentication phase of our scheme. Besides these two roles, the roles for the session, goal and environment in HLPSSL must be specified for our scheme.

The role of the initiator, Ed_i in HLPSSL is given in Fig. 5. Ed_i first receives the start signal, updates its state from 0 to 1, which is maintained by the variable *State*, and then sends the registration request $\{I_i\}$ securely to the CS during the registration phase with the help of $TX(\)$ channel. The type declaration channel (*dy*) declares that the channel is for the Dolev–Yao threat model [42], which indicates that the intruder (*i*) has the ability to intercept, analyze, and/or modify messages transmitted over a insecure public channel. After that Ed_i receives the message $\{PId_i, C'_k\}$ from CS securely by the help of $RX(\)$ channel. In this role, the *played_by A* declaration indicates that the agent named in variable *A* plays in the role. By the declaration *secret(Id_i, Pw_i, sec1, ED_i)*, it indicates that the information Id_i and Pw_i are only known to Ed_i . During the login and authentication phase, Ed_i sends the message $\{P_1, P_2, PId_i\}$ to CS through open channel. Ed_i then receives the message $\{P_3, P_4, T_i\}$ from CS via public channel. Finally, Ed_i replies with the message $\{V_i\}$ to CS via open channel. A knowledge declaration (generally in the top-level *Environment* role) is used to specify the intruder's initial knowledge. Immediate reaction transitions are of the form $X = | > Y$, which relate an event *X* and an action *Y*. By the declaration *witness(ED_i, CS, alice_bob_r1, R1')*, we mean that Ed_i has generated the random nonce r_1 freshly for the CS . On the other hand, by the declaration *request(CS, ED_i, bob_alice_r2, R2')*, it indicates that Ed_i 's acceptance of the random nonce r_2 generated for Ed_i by CS . In a similar way, we have implemented the role for the CS during the registration as well as login and authentication phases in Fig. 6.

Finally, the roles for the session, goal and environment of our proposed scheme are provided in Fig. 7. All the basic roles: *embeddeddevice* and *cloudserver* are the instances with concrete arguments in the role of the session. The top-level role (environment) is always defined in HLPSSL specification. In HLPSSL, the intruder (*i*) also participates in the execution of protocol as a concrete session as shown in Fig. 7. In our implementation, there are two secrecy goals and two authentication goals. For example, the secrecy goal: *secrecy_of sec1* reveals that Id_i and Pw_i are kept secret to Ed_i only. The authentication goal: *authentication_on alice_bob_r1* means that Ed_i generates a random nonce r_1 , where r_1 is only known to Ed_i . When CS receives r_1 from other messages from Ed_i , CS performs a strong authentication for Ed_i based on r_1 .

6.3 Simulation results

Our scheme is simulated under the widely used OFMC backend using the Security Protocol ANimator for AVISPA (SPAN) [34]. The following verifications are executed while simulating our scheme [35–41]:

- *Executability check on non-trivial HLPSSL specifications* Due to some modeling mistakes, the protocol model can not sometimes execute to completion. It may

```

role embeddeddevice (EDi, CS: agent,
    SKedics: symmetric_key,
    H : hash_func,
    TX, RX: channel(dy))
played_by EDi
def=
local State: nat,
    Idi, Pwi, Ii, PIdi, Ckk, Idcs, Rs, Xcs, Et: text,
    G, R1, P1, P2, R2, SK, Vi: text,
    F : hash_func
const sec1, sec2, alice_bob_r1, bob_alice_r2 : protocol_id
init State := 0
transition
% Registration phase
1. State = 0  $\wedge$  RX(start)  $\Rightarrow$ 
State' := 1  $\wedge$  Ii' := H(Idi.Pwi)
 $\wedge$  secret({Idi,Pwi}, sec1, EDi)
% Send the registration request {Ii} to CS securely
% via secure channel
 $\wedge$  TX({Ii'}_SKedics)
% Receive registration reply {PIdi, Ck'} from CS securely
2. State = 1  $\wedge$  RX({xor(H(Rs'.Idcs.H(Idi.Pwi)), Idcs).
F(H(Rs'.Xcs.Et.xor(H(Rs'.Idcs.
H(Idi.Pwi)), Idcs)).G)}_SKedics)  $\Rightarrow$ 
State' := 3  $\wedge$  secret({Idcs,Xcs,Et}, sec2, EDi)
% Login and authentication phase
 $\wedge$  R1' := new()  $\wedge$  P1' := F(R1'.G)
 $\wedge$  P2' := H(R1'.H(Rs'.Xcs.Et.xor(H(Rs'.
Idcs.H(Idi.Pwi)), Idcs)))
% Send login request {P1, P2, PIdi} to CS via open channel
 $\wedge$  TX(P1'.P2'.xor(H(Rs'.Idcs.H(Idi.Pwi)), Idcs))
%% Edi has generated r1 freshly for CS
 $\wedge$  witness(EDi, CS, alice_bob_r1, R1')
% Receive message {P3, P4, Ti} from CS via open channel
3. State = 3  $\wedge$  RX(F(R2'.G).F(R2'.F(H(xor(xor(Rs',H(Xcs.
xor(H(Rs'.Idcs.H(Idi.Pwi)), Idcs))),
xor(H(Idi.Pwi),F(H(Rs'.Xcs.Et.
xor(H(Rs'.Idcs.H(Idi.Pwi)), Idcs)).G))))).G)).
xor(Rs',H(Xcs.xor(H(Rs'.Idcs.
H(Idi.Pwi)), Idcs))))  $\Rightarrow$ 
State' := 5  $\wedge$  SK' := F(R1'.F(R2'.G))
 $\wedge$  Vi' := H(F(R1'.F(H(Rs'.Xcs.Et.PIdi')).G)).SK')
% Send message {Vi} to CS via open channel
 $\wedge$  TX(Vi')
% Edi's acceptance of random nonce r2 generated for Edi by CS
 $\wedge$  request(CS, EDi, bob_alice_r2, R2')
end role

```

Fig. 5 Role specification in HLPSSL for *Ed_i*

```

role cloudserver (EDi, CS: agent,
  SKedics: symmetric_key,
  H : hash_func,
  TX, RX: channel(dy))
played_by CS
def=
  local State: nat,
    Idi, Pwi, Idcs, Rs, Ck, Ckk : text,
    PIdi, Xcs, Et, G, R1, R2, P3, P4 : text,
    Ai, Ai1, Ti : text,
    F : hash_func
  const sec1, sec2, alice_bob_r1, bob_alice_r2 : protocol_id
  init State := 0
  transition
  % Registration phase
  % Receive registration request <Ii> from EdI securely
  % embedded device securely
  1. State = 0  $\wedge$  RX( $\{H(\text{Idi.Pwi})\}_{\text{SKedics}}$ )  $\Rightarrow$ 
  State' := 2  $\wedge$  secret( $\{\text{Idi.Pwi}\}$ , sec1, EDi)
     $\wedge$  Rs' := new()
     $\wedge$  PIdi' := xor(H(Rs'.Idcs.H(Idi.Pwi)), Idcs)
     $\wedge$  Ck' := H(Rs'.Xcs.Et.PIdi')
     $\wedge$  Ckk' := F(Ck'.G)
     $\wedge$  secret( $\{\text{Idcs.Xcs.Et}\}$ , sec2, EDi)
  % Send registration reply {PIdi, Ck'} to EdI securely
     $\wedge$  TX( $\{\text{PIdi'.Ckk'}\}_{\text{SKedics}}$ )
  % Login and authentication phase
  % Receive login request {P1, P2, PIdi} from EdI via open channel
  2. State = 2  $\wedge$  RX(F(R1'.G).H(R1'.H(Rs'.Xcs.Et.xor(H(Rs'.
    Idcs.H(Idi.Pwi)), Idcs))))
    xor(H(Rs'.Idcs.H(Idi.Pwi)), Idcs))  $\Rightarrow$ 
  State' := 4  $\wedge$  R2' := new()  $\wedge$  P3' := F(R2'.G)
     $\wedge$  Ai' := H(xor(xor(Rs', H(Xcs.xor(H(Rs'.Idcs.
    H(Idi.Pwi)), Idcs))), xor(H(Idi.Pwi),
    F(H(Rs'.Xcs.Et.xor(H(Rs'.Idcs.
    H(Idi.Pwi)), Idcs)).G))))
     $\wedge$  Ai1' := F(Ai'.G)
     $\wedge$  P4' := F(R2'.Ai1')
     $\wedge$  Ti' := xor(Rs', H(Xcs.xor(H(Rs'.Idcs.
    H(Idi.Pwi)), Idcs)))
  % Send message {P3, P4, Ti} to EdI via open channel
     $\wedge$  TX(P3'.P4'.Ti')
  %% CS has generated r2 freshly for EdI
     $\wedge$  witness(CS, EDi, bob_alice_r2, R2')
  % Receive message {Vi} from EdI via open channel
  3. State = 4  $\wedge$  RX(H(F(R1'.F(H(Rs'.Xcs.Et.PIdi').G)).
    F(R1'.F(R2'.G))))  $\Rightarrow$ 
  % CS's acceptance of random nonce r1 generated for CS by EdI
  State' := 6  $\wedge$  request(EDi, CS, alice_bob_r1, R1')
end role

```

Fig. 6 Role specification in HPLSL for CS


```

role session (EDi, CS: agent,
              SKedics: symmetric_key,
              H: hash_func)
def=
local TX1, RX1, TX2, RX2: channel(dy)
composition
  embeddeddevice (EDi, CS, SKedics, H, TX1, RX1)
  ∧ cloudserver (EDi, CS, SKedics, H, TX2, RX2)
end role

role environment()
def=
const edi, cs: agent,
      skedics: symmetric_key,
      h, f: hash_func,
      p1, p2, pidi, p3, p4, ti, vi: text,
      sec1, sec2, alice_bob_r1, bob_alice_r2: protocol_id
intruder_knowledge = {edi, cs, h, f,
                      p1, p2, pidi, p3, p4, ti, vi}
composition
  session(edi, cs, skedics, h)
  ∧ session (i, cs, skedics, h)
  ∧ session (edi, i, skedics, h)
end role
goal
  secrecy_of sec1
  secrecy_of sec2
  authentication_on alice_bob_r1
  authentication_on bob_alice_r2
end goal
environment()

```

Fig. 7 Role specification in HLPSL for the session, goal and environment

happen that the AVISPA backends cannot find an attack, if the protocol model can not reach to a state where that attack can happen. An executability test is then very essential [43]. Our scheme shows that the protocol description is well matched with the designed goals as specified in Figs. 5, 6 and 7 for the executability test.

- *Replay attack check* For the replay attack check, the OFMC backend verifies whether the legitimate agents can execute the specified protocol by performing a search of a passive intruder. This backend provides the intruder the knowledge of some normal sessions between the legitimate agents. The test results shown in Fig. 8 indicate that our scheme is secure against the replay attack.
- *Dolev–Yao model check* For the Dolev–Yao model check, the OFMC backend also verifies whether there is any man-in-the-middle attack possible by an intruder. It is clear from the results reported in Fig. 8 that our scheme fulfills the design properties and is also secure under this backend.

Fig. 8 Result of the analysis using OFMC backend

```
% OFMC
% Version of 2006/02/13
SUMMARY
SAFE
DETAILS
BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
C:\progra~1\SPAN\testsuite
  \results\auth.if
GOAL
as_specified
BACKEND
OFMC
COMMENTS
STATISTICS
parseTime: 0.00s
searchTime: 0.03s
visitedNodes: 12 nodes
depth: 4 plies
```

6.4 Informal security analysis

The security of the proposed scheme is based on the difficulty of the high entropy ECC and the secure cryptographic hash function. This section analyzes the security features of the proposed scheme under the thought of tamper proof-embedded device and shows that all security requirements stated in Table 4 are achieved in the proposed scheme.

6.4.1 Replay attack

A replay attack involves retransmitting previously intercepted messages. In the proposed scheme, an attacker may intercept the message $\{P_1, P_2, PId_i\}$ which is transmitted from Ed_i to CS . Using this intercepted message the attacker may try to launch the replay attack. To login as a legal device, an attacker may retransmit the intercepted message $\{P_1, P_2, PId_i\}$ to CS . Upon receiving the login request, CS obtains data corresponding to PId_i and derives r_s as $r_s = T_i \oplus h(X_{cs} || PId_i)$. Then, it compute the parameters C_k and P_2^* and verifies $P_2^* \stackrel{?}{=} P_2$. The verification holds as $P_2^* = h(P_1 \cdot C_k) = P_2$. Next, it sends $\{P_3, P_4, T_i\}$ to Ed_i . However, after receiving the message $\{P_3, P_4, T_i\}$, the attacker cannot compute $A_i = h(T_i \oplus I_i \oplus C'_k)$ without knowing I_i of Ed_i since I_i of Ed_i is neither sent through any messages over public channel nor can be obtained from the embedded device Ed_i due to its tamper proof design. Thus, an attacker cannot compute valid session key $SK = r_1 \cdot P_3 = r_1 \cdot r_2 \cdot G$ and the security parameter $V_i = h((r_1 \cdot C'_k) || SK)$. Hence, an attacker cannot launch the replay attack using the intercepted message $\{P_1, P_2, PId_i\}$.

6.4.2 Man-in-the-middle attack

In the proposed scheme, the man-in-the-middle attack is disallowed by mutual authentication between Ed_i and CS . As a result, man-in-the-middle attacks are thwarted since we show that the proposed scheme achieves mutual authentication in Sect. 6.4.10.

6.4.3 Insider attack

In the proposed scheme, Ed_i sends $\{I_i = h(Id_i || Pw_i)\}$ instead of Pw_i to CS securely during registration phase. The privileged insider of CS cannot obtain the password Pw_i since it is protected by Ed_i 's identity Id_i and collision-resistant one-way hash function $h(\cdot)$. Therefore, the proposed scheme can resist the insider attack.

6.4.4 Stolen-verifier attack

In the proposed scheme, during the registration phase of the Ed_i , cloud server CS stores $t_i = T_i \oplus X_{cs}$, $a'_i = A'_i \oplus X_{cs}$ and $e_t = E_t \oplus X_{cs}$ against PId_i in its database. Even if an attacker steals this record from the database of CS , he cannot use it to apply any malicious activity such as server impersonation attack. The reason is that the values $\{T_i, A'_i, E_t\}$ are not accessible to the attacker in plaintext from the stolen record because these values are protected with the secret key X_{cs} of CS . Moreover, an attacker cannot create a valid login request to pass the authentication steps without the knowledge of C'_k since it is not stored in CS 's database. In addition, the computation of cookie $C'_k = C_k \cdot G$ relies on the correct computation of $C_k = h(r_s || X_{cs} || E_t || PId_i)$. Without the knowledge of CS 's secret key X_{cs} the attacker cannot compute a valid cookie C_k . Thus, the attacker cannot create a valid login request. Therefore, the proposed scheme can withstand the stolen-verifier attack.

6.4.5 Impersonation attack

As discussed in Sect. 6.4.1, we can see that the attacker cannot impersonate the embedded device Ed_i and the cloud server CS through replaying the previously intercepted messages. Therefore, the attacker has to create a fresh login request or a response message if he/she wishes to impersonate Ed_i or CS , respectively. Since the embedded device Ed_i is tamper proof, it is not possible to extract values stored in it. Without knowing the cookie C'_k and PId_i , the attacker cannot build a valid login request message $\{P_1, P_2, PId_i\}$, where $P_1 = r_1 \cdot G$ and $P_2 = h(r_1 \cdot C'_k)$. Further, without knowing CS 's secret key X_{cs} and $A_i = h(r_s \oplus h(X_{cs} || PId_i)) \oplus I_i \oplus C'_k$, the attacker cannot create the message $\{P_3, P_4, T_i\}$, where $P_3 = r_2 \cdot G$ and $P_4 = r_2 \cdot A'_i$. Therefore, the proposed scheme can resist the impersonation attack.

6.4.6 Brute force attack

For deploying a brute force attack, the attacker needs to extract the security parameters $\{P_1, P_2, P_3, P_4$ and $T_i\}$ from the transmitted messages between Ed_i and CS . Even he obtains these security parameters, he still cannot find the password Pw_i as the CS 's

secret key X_{cs} is unknown to him and there is no way of guessing random numbers r_1 and r_2 . Therefore, the proposed scheme can resist the brute force attack.

6.4.7 Cookie theft attack

In the proposed scheme, the cookie $C'_k = h(r_s || X_{cs} || E_r || PId_i) \cdot G$ is stored as an ECC point in the embedded device Ed_i . But, it is not possible to extract the cookie from Ed_i due to its tamper proof property. Moreover, the cookie C'_k is not sent through any of the communication messages over public channel. As a result, under any situation, the attacker cannot obtain the cookie C'_k . Therefore, the proposed scheme can resist the cookie theft attack.

6.4.8 Offline password guessing attack

As we illustrated above, throughout the proposed scheme, Ed_i 's password Pw_i only makes one presences as $I_i = h(ID_i || Pw_i)$. Manifestly, the attacker cannot toss an offline password guessing attack without knowing Ed_i 's identity Id_i since Id_i is neither sent through any message nor stored in the Ed_i and CS . In addition, we have proved that the improved scheme achieves user anonymity in Sect. 6.4.9. Therefore, the proposed scheme can resist the offline password guessing attack.

6.4.9 Device anonymity

Device anonymity means that an attacker cannot dig up the device's masked identity I_i from the transmitted messages during login and authentication phase. In the proposed scheme, Ed_i sends $\{P_1, P_2, PId_i\}$ to CS , where $P_1 = r_1 \cdot G$, $P_2 = h(r_1 \cdot C'_k)$ and $PId_i = h(r_s || Id_{cs} || I_i) \oplus Id_{cs}$. Here, PId_i is related with the Ed_i 's masked identity I_i . However, I_i is well protected by Id_{cs} and the random number r_s with help of hash function. I_i , Id_{cs} and r_s are neither sent through any message nor stored in the Ed_i and CS in plaintext. This shows that the attacker cannot obtain the masked identity I_i of Ed_i . Therefore, the proposed scheme preserves device anonymity.

6.4.10 Mutual authentication

The proposed scheme ensures the unbeaten mutual authentication between Ed_i and CS . CS authenticates Ed_i by verifying $P_2^* \stackrel{?}{=} P_2$ and $V_i^* \stackrel{?}{=} V_i$. A valid P_2 can be computed by only a valid Ed_i because $P_2 = h(r_1 \cdot C'_k)$ where $C'_k = C_k \cdot G$. In the proposed scheme, the attacker cannot compute valid P_2 without knowing the cookie C'_k and he has no way to access the cookie C'_k . The reason is that C'_k is neither sent in messages transmitted over public nor can be extracted from the tamper proof device Ed_i . For similar reasons, the attacker cannot compute $V_i = h((r_1 \cdot C'_k) || SK)$. Thus, CS authenticates Ed_i . Also, CS can be authenticated by Ed_i by verifying $P_4^* \stackrel{?}{=} P_4$ where $P_4^* = P_3 \cdot A_i$. Therefore, the proposed scheme achieves proper mutual authentication.

6.5 Session key agreement

Subsequent to the authentication process, the Ed_i and CS share a session key $SK = r_1 \cdot P_3 = r_1 \cdot r_2 \cdot G = r_2 \cdot P_1 = SK^*$. Since the attacker has no knowledge of the random number r_1 and r_2 , the session key cannot be directly computed, as it is protected by a high entropy ECC point. Hence, the proposed scheme guarantees the secrecy of future session keys.

6.5.1 Forward secrecy

Forward secrecy means that the attacker cannot find session keys created in past sessions even if he/she discovers the Ed_i 's password P_{w_i} and the CS 's secret key X_{CS} . In the proposed scheme, Ed_i and CS compute an incomparable session key $SK = r_1 \cdot P_3 = r_1 \cdot r_2 \cdot G = r_2 \cdot P_1 = SK^*$ in every run of the proposed scheme. The attacker cannot compute SK (or SK^*) from $P_1 = r_1 \cdot G$ and $P_3 = r_2 \cdot G$ even if he/she find out Ed_i 's password P_{w_i} and CS 's secret key X_{CS} due to the hardness of high entropy ECC points. Therefore, the proposed scheme can provide the forward secrecy.

6.5.2 Provides confidentiality

In the proposed scheme, the required device authentication parameter is secured by using hash function and ECC points. It allows only authenticated devices to gain access to the legitimate cloud server. The proposed scheme is also resilient to eavesdropping and traffic analysis and assures confidentiality by making sure that the difficulty of brute force attack is high.

7 Performance analysis

In this section, we compare the security requirements and performance of the proposed scheme with the Kalra and Sood's scheme [23] to manifest the merits of the proposed scheme. In order to carry out the performance analysis, the following notations have been defined:

- T_h is the execution time of a hash operation.
- T_{ecm} is the execution time of an ECC point multiplication operation.

7.1 Computation cost

The computational costs of the login and authentication phases have been considered since these two phases are executed more often as compared to the other phases in an authentication scheme. An experiment results of [44] demonstrate that an execution time (computation costs) of T_h and T_{ecm} are $2.3 \mu s$ and $22.26 \times 10^2 \mu s$. Note that since very inexpensive computation is associated with lightweight operations (i.e., concatenation, comparison and XOR), their computational costs have been ignored.

Table 2 Computational cost comparison

Schemes	Login and authentication phase		Total
	Embedded device	Cloud server	
Kalra and Sood [23]	$4T_h + 3T_{ecm}$	$5T_h + 4T_{ecm}$	$9T_h + 7T_{ecm} \approx 15.603 \times 10^3 \mu s$
Proposed scheme	$3T_h + 4T_{ecm}$	$4T_h + 4T_{ecm}$	$7T_h + 8T_{ecm} \approx 17.824 \times 10^3 \mu s$

T_h the computational cost of a hash operation; T_{ecm} the computational cost of a ECC point multiplication operation

Table 3 Communication and storage cost comparison

Schemes	Communication cost		Storage cost (bits)
	Number of messages	Number of bits	
Kalra and Sood [23]	3	1760	320
Proposed scheme	3	1760	480

In Table 2, we compare the computational cost of the proposed scheme with the Kalra and Sood’s scheme [23] in the login and authentication phase. Based on Table 2, the computational cost of Kalra and Sood’s scheme and the proposed scheme are $9T_h + 7T_{ecm} \approx 15.603 \times 10^3 \mu s$ and $7T_h + 8T_{ecm} \approx 17.824 \times 10^3 \mu s$. Here, we see that the proposed scheme has little increased computation cost as compared to Kalra and Sood’s scheme. For the reason that, in the proposed scheme, the forward secrecy and backward secrecy is achieved through the session key agreement between Ed_i and CS while Kalra and Sood’s scheme does not.

7.2 Communication cost

Table 3 compares the communication cost of the proposed scheme with the Kalra and Sood’s scheme [23]. Though calculating the communication cost of the schemes, we assumed that the length of the random number(r_1, r_2, r_s) is 160 bits, device identity (Id_i) is 160 bits, pseudo-identity (PId_i) is 160 bits and the security parameter(C'_k, T_i, V_i) is 160 bits. Also, we assumed that the output (digest) of hash function(for SHA-1 [45]) is 160 bits. In proposed scheme, we consider the elliptic curve cryptosystem (ECC) with 160-bit as its security strength is equivalent as RSA cryptosystem of 1024-bit. Hence, for an elliptic curve $E_p(a, b)$, all parameters (p, a and b) are 160-bits each. Next, an ECC point $P = (x_p, y_p) \in E_p(a, b)$ needs $(160 + 160) = 320$ bits. In proposed scheme, the login request message $\{P_1, P_2, PId_i\}$ requires $(320 + 320 + 160) = 800$ bits, the message $\{P_3, P_4, T_i\}$ requires $(320 + 320 + 160) = 800$ bits and the message $\{V_i\}$ requires 160 bits. Therefore, the proposed scheme requires $(800 + 800 + 160) = 1760$ bits for the communication cost of three messages transmitted between Ed_i and CS . However, the communication cost of the Kalra and Sood’s scheme also 1760 bits. Here, we see that the proposed scheme has same communication cost as compared to Kalra and Sood’s scheme.

Table 4 Security requirements comparison

Security requirements	Schemes	
	Kalra and Sood [23]	Proposed scheme
Replay attack	✓	✓
an-in-the-middle attack	✓	✓
Insider attack	×	✓
Stolen-verifier attack	✓	✓
Impersonation attack	✓	✓
Brute force attack	✓	✓
Cookie theft attack	✓	✓
Offline password guessing attack	×	✓
Device anonymity	×	✓
Mutual authentication	×	✓
Session key agreement	×	✓
Forward secrecy	×	✓
Confidentiality	✓	✓

✓ achieved; × not achieved

7.3 Storage cost

We scale the storage cost of the proposed scheme with respect to Kalra and Sood's scheme. Here, we consider the storage cost of the embedded device since it has tiny memory. Toward the end of the registration phase in Kalra and Sood's scheme, the embedded device (Ed_i) needs to store $\{C'_k\} = 320$ bits of data in its memory. However, in the proposed scheme, embedded device contains $\{C'_k, PId_i\} = 320 + 160 = 480$ bits of data in its memory, which is little greater than the Kalra and Sood's scheme. In the proposed scheme, the additional occupied memory space for PId_i provides device anonymity while Kalra and Sood's scheme does not. Table 3 shows that the storage cost required by the embedded device Ed_i in Kalra and Sood's scheme and the proposed scheme.

7.4 Security requirements comparison

Table 4 lists the security requirements comparison between the proposed scheme and the Kalra and Sood's. From Table 4, it is manifest that the proposed scheme can resist various attacks. In contrast, the scheme of Kalra and Sood is vulnerable to offline password guessing and insider attacks. Additionally, the scheme of Kalra and Sood does not achieve device anonymity, mutual authentication and session key agreement. Thus, the proposed scheme provides a greater result over the scheme of Kalra and Sood with respect to security strength.

In Fig. 9, we summarize the performance of the proposed scheme and Kalra and Sood's scheme. In Fig. 9, let C_1 be the communication cost(in bits), C_2 be the com-

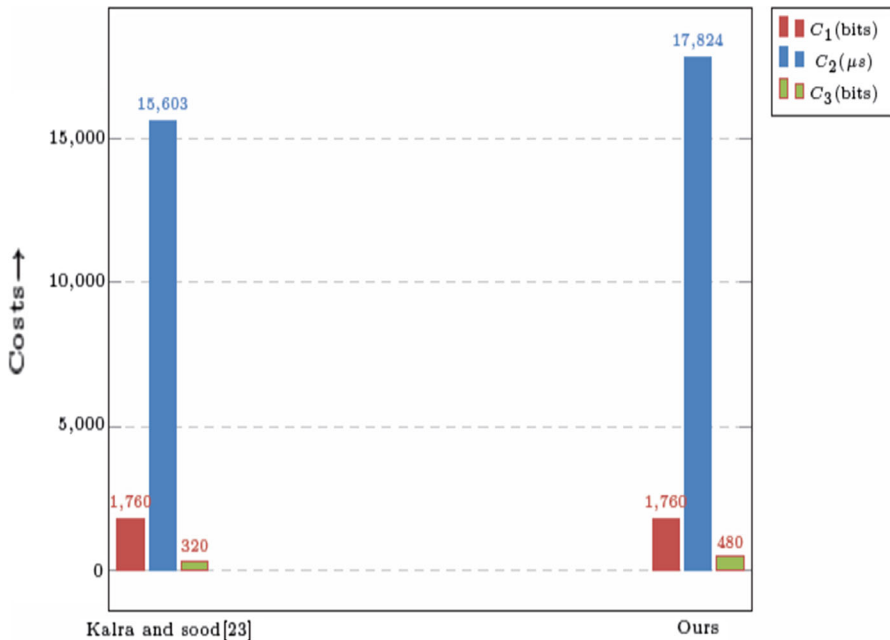


Fig. 9 Performance comparison

putation cost (in μs) and C_3 be the storage cost (in bits). Compared to the Kalra and Sood's scheme, computation cost and storage cost in the proposed scheme are slightly increased. This is justified, because the proposed scheme achieves all the security requirements while Kalra and Sood's scheme does not. Conclusively, the proposed scheme sustains reasonable efficiency and is well suited to authenticate the embedded device in cloud and IoT environment.

8 Conclusion

Kalra and Sood proposed an authentication scheme based on ECC for IoT and cloud servers and proved its immunity against various attacks. However, after reviewing their scheme and carrying out security analysis, two attacks, i.e., offline password guessing and insider attacks have been presented in different circumstances. Moreover, it has been shown that their scheme fails to achieve device anonymity, session key agreement and mutual authentication. The analyses show that their scheme is not suitable for practical applications. In order to fix the shortcomings of Kalra and Sood's scheme, we have proposed an enhanced authentication scheme based on ECC. Performance and security analyses show that the proposed scheme is invincible to various attacks and is well crafted for IoT and cloud server environment.

In future, we would like to extend our scheme for non-tamper-resistant embedded devices.

Acknowledgements This work was supported by the National Natural Science Foundation of China under Grant No. 61300220. Fan Wu is supported by Fujian Education and Scientific Research Program for Young and Middle-aged Teachers under Grant No. JA14369 and University Distinguished Young Research Talent Training Program of Fujian Province (Year 2016). This work was partially supported by the Information Security Education and Awareness (ISEA) Phase II Project, Department of Electronics and Information Technology (DeitY), India.

References

1. Zhou J, Leppanen T, Harjula E, Ylianttila M, Ojala T, Yu C, Jin H, Yang LT (2013) Cloudthings: a common architecture for integrating the internet of things with cloud computing. In: 2013 IEEE 17th International Conference on Computer Supported Cooperative Work in Design (CSCWD). IEEE, pp 651–657
2. Chang KD, Chen CY, Chen JL, Chao HC (2011) Internet of things and cloud computing for future internet. In: Security-enriched urban computing and smart grid. Springer, pp 1–10
3. Botta A, de Donato W, Persico V, Pescapé A (2016) Integration of cloud computing and internet of things: a survey. *Future Gener Comput Syst* 56:684–700
4. Fox GC, Kamburugamuve S, Hartman RD (2012) Architecture and measured characteristics of a cloud based internet of things. In: 2012 International Conference on Collaboration Technologies and Systems (CTS). IEEE, pp 6–12
5. Dash SK, Mohapatra S, Pattnaik PK (2010) A survey on applications of wireless sensor network using cloud computing. *Int J Comput Sci Eng Technol* 1(4):50–55
6. Suciú G, Vulpe A, Halunga S, Fratu O, Todoran G, Suciú V (2013) Smart cities built on resilient cloud computing and secure internet of things. In: 2013 19th International Conference on Control Systems and Computer Science (CSCS). IEEE, pp 513–518
7. Mühlbach S, Wallner S (2008) Secure communication in microcomputer bus systems for embedded devices. *J Syst Archit* 54(11):1065–1076
8. He D, Zeadally S (2015) An analysis of rfid authentication schemes for internet of things in healthcare environment using elliptic curve cryptography. *IEEE Internet Things J* 2(1):72–83
9. Afreen R, Mehrotra SC (2011) A review on elliptic curve cryptography for embedded systems. *J Comput Sci Inf Technol* 3(3):84–103
10. Salas M (2013) A secure framework for OTA smart device ecosystems using ECC encryption and biometrics. In: *Advances in Security of Information and Communication Networks*. Springer, pp 204–218
11. Wu ST, Chiu JH, Chieu BC (2005) ID-based remote authentication with smart cards on open distributed system from elliptic curve cryptography. In: *IEEE International Conference on Electro Information Technology*. IEEE, pp 5
12. Tian X, Wong DS, Zhu RW (2005) Analysis and improvement of an authenticated key exchange protocol for sensor networks. *IEEE Commun Lett* 9(11):970–972
13. Abi-Char PE, Mhamed A, El-Hassan B (2007) A fast and secure elliptic curve based authenticated key agreement protocol for low power mobile communications. In: *The 2007 International Conference on Next Generation Mobile Applications, Services and Technologies (NGMAST'07)*. IEEE, pp 235–240
14. Yang JH, Chang CC (2009) An id-based remote mutual authentication with key agreement scheme for mobile devices on elliptic curve cryptosystem. *Comput Secur* 28(3):138–143
15. Islam SH, Biswas GP (2011) A more efficient and secure id-based remote mutual authentication with key agreement scheme for mobile devices on elliptic curve cryptosystem. *J Syst Softw* 84(11):1892–1898
16. He D, Chen J, Hu J (2012) An ID-based client authentication with key agreement protocol for mobile client–server environment on ECC with provable security. *Inf Fusion* 13(3):223–230
17. Ray S, Biswas GP (2012) Establishment of ECC-based initial secrecy usable for IKE implementation. In: *Proceedings of the World Congress on Engineering*, vol 1
18. Granjal J, Monteiro E, Silva JS (2013) End-to-end transport-layer security for internet-integrated sensing applications with mutual and delegated ECC public-key authentication. In: *IFIP Networking Conference, 2013*. Brooklyn, NY, pp 1–9
19. Jiang R, Lai C, Luo J, Wang X, Wang H (2013) EAP-based group authentication and key agreement protocol for machine-type communications. *Int J Distrib Sens Netw* 9(11):1–14

20. Yao X, Chen Z, Tian Y (2015) A lightweight attribute-based encryption scheme for the internet of things. *Future Gener Comput Syst* 49:104–112
21. Moosavi SR, Nigussie E, Virtanen S, Isoaho J (2014) An elliptic curve-based mutual authentication scheme for RFID implant systems. *Procedia Comput Sci* 32:198–206
22. Liao YP, Hsiao CM (2014) A secure ECC-based RFID authentication scheme integrated with ID-verifier transfer protocol. *Ad Hoc Netw* 18:133–146
23. Kalra S, Sood SK (2015) Secure authentication scheme for IoT and cloud servers. *Pervasive Mob Comput* 24:210–223
24. Koblitz N (1987) Elliptic curve cryptosystems. *Math Comput* 48(177):203–209
25. Menezes AJ, Van Oorschot PC, Vanstone SA (1996) *Handbook of applied cryptography*. CRC Press, Boca Raton
26. Miller VS (1985) Use of elliptic curves in cryptography. In: *Advances in Cryptology—CRYPTO’85 Proceedings*. Springer, pp 417–426
27. Hancock B (1999) Security views. *Comput Secur* 18(7):553–564
28. Caelli WJ, Dawson EP, Rea SA (1999) Pki, elliptic curve cryptography, and digital signatures. *Comput Secur* 18(1):47–66
29. Bertino E, Shang N, Wagstaff SS Jr (2008) An efficient time-bound hierarchical key management scheme for secure broadcasting. *IEEE Trans Dependable Secure Comput* 5(2):65–70
30. Odelu V, Das AK, Goswami A (2016) A secure and efficient time-bound hierarchical access control scheme for secure broadcasting. *Int J Ad Hoc Ubiquitous Comput* 22(4):236–248
31. Lassus M (1997) Smart-cards-a cost-effective solution against electronic fraud. In: *European Conference on Security and Detection (ECOS 1997)*, pp 81–85
32. Macq BM, Quisquater JJ (1995) Cryptology for digital TV broadcasting. *Proc IEEE* 83(6):944–957
33. Wan Z, Liu J, Zhang R, Deng RH (2013) A collusion-resistant conditional access system for flexible-pay-per-channel pay-TV broadcasting. *IEEE Trans Multimed* 15(6):1353–1364
34. AVISPA Automated Validation of Internet Security Protocols and Applications. <http://www.avispa-project.org/>. Accessed on February (2016)
35. Odelu V, Das AK, Goswami A (2015) A secure biometrics-based multi-server authentication protocol using smart cards. *IEEE Trans Inf Forensics Secur* 10(9):1953–1966
36. Odelu V, Das AK, Goswami A (2015) A secure and scalable group access control scheme for wireless sensor networks. *Wirel Pers Commun* 85(4):1765–1788
37. Odelu V, Das AK, Goswami A (2015) An effective and robust secure remote user authenticated key agreement scheme using smart cards in wireless communication systems. *Wirel Pers Commun* 84(4):2571–2598
38. Odelu V, Das AK, Goswami A (2015) DMAMA: dynamic migration access control mechanism for mobile agents in distributed networks. *Wirel Pers Commun* 84(1):207–230
39. Das AK (2015) A secure and efficient user anonymity-preserving three-factor authentication protocol for large-scale distributed wireless sensor networks. *Wirel Pers Commun* 82(3):1377–1404
40. Das AK (2016) A secure and robust temporal credential-based three-factor user authentication scheme for wireless sensor networks. *Peer-to-peer Netw Appl* 9(1):223–244
41. Mishra D, Das AK, Mukhopadhyay S (2016) A secure and efficient ECC-based user anonymity-preserving session initiation authentication protocol using smart card. *Peer-to-peer Netw Appl* 9(1):171–192
42. Dolev D, Yao AC (1983) On the security of public key protocols. *IEEE Trans Inf Theory* 29(2):198–208
43. von Oheimb D (2005) The high-level protocol specification language HLPSSL developed in the EU project AVISPA. In: *Proceedings of APPSEM 2005 Workshop*
44. Kilinc HH, Yanik T (2014) A survey of SIP authentication and key agreement schemes. *IEEE Commun Surv Tutor* 16(2):1005–1023
45. PUB FIPS (1995) 180-1. Secure hash standard. *Natl Inst Stand Technol* 17:45