Pervasive and Mobile Computing 24 (2015) 210–223

Contents lists available at ScienceDirect

Pervasive and Mobile Computing

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

Secure authentication scheme for IoT and cloud servers

Sheetal Kalra [a,*], Sandeep K. Sood [b]

[a] Department of Computer Science and Engineering, Guru Nanak Dev University, Regional Campus, Jalandhar, Punjab, 144001, India
[b] Department of Computer Science and Engineering, Guru Nanak Dev University, Regional Campus, Gurdaspur, Punjab, 143521, India

ARTICLE INFO

Article history:
Available online 11 August 2015

Keywords:
Authentication
Cookies
Cloud computing
Elliptic Curve Cryptography
Internet of Things

ABSTRACT

Internet of Things (IoT) is an upcoming platform where information and communication technology connect multiple embedded devices to the Internet for performing information exchange. Owing to the immense development of this technology, embedded devices are becoming more sophisticated every day and are being deployed in various arenas of life. An important advancement in today's technology is the ability to connect such devices to large resource pools such as cloud. Integration of embedded devices and cloud servers brings wide applicability of IoT in many commercial as well as Government sectors. However, the security concerns such as authentication and data privacy of these devices play a fundamental role in successful integration of these two technologies. Elliptic Curve Cryptography (ECC) based algorithms give better security solutions in comparison to other Public Key Cryptography (PKC) algorithms due to small key sizes and efficient computations. In this paper, a secure ECC based mutual authentication protocol for secure communication of embedded devices and cloud servers using Hyper Text Transfer Protocol (HTTP) cookies has been proposed. The proposed scheme achieves mutual authentication and provides essential security requirements. The security analysis of the proposed protocol proves that it is robust against multiple security attacks. The formal verification of the proposed protocol is performed using AVISPA tool, which confirms its security in the presence of a possible intruder.

1. Introduction

An embedded system is a special purpose system composed of computer hardware, software and additional mechanical components with processing capability dedicated to a specific task. Increased processing power and more sophisticated software has evolved embedded devices from single microcontroller chip with limited capabilities to multi-component intelligent systems. Single-function embedded devices have matured as "smart systems" with powerful processors, operating systems and efficient connectivity. With these smart systems, the enterprises can envision to deploy interconnected complex systems that can collect, analyze and communicate data efficiently. Presently, many organizations are trying to collaborate their embedded systems with cloud. Embedded devices can leverage vast amount of data storage and computing capability from cloud computing. Cloud computing has become increasingly popular over last few years because of its infinite resources and dynamic elasticity. The cloud technology consists of both hardware and software provided by the data center for which customers have to pay only for the resources they consume. The number of Internet connected devices is rapidly increasing and these devices not only include personal computers but also small embedded devices such as Personal

* Corresponding author.
E-mail addresses: sheetal.kalra@gmail.com (S. Kalra), san1198@gmail.com (S.K. Sood).

http://dx.doi.org/10.1016/j.pmcj.2015.08.001
1574-1192/

Digital Assistant (PDA), bank cards in the wallet and similarly many more. This evolution leads to a new scenario where Internet connected devices could benefit from cloud computing abundant resources. A networked embedded device can have capabilities based upon operations carried out in cloud and not simply restricted to its own local resources. Security still remains the major issue while getting connected to cloud for using its resources [1,2]. Embedded devices must be authenticated before getting services of a cloud and also cloud servers should be authenticated by these devices. Elliptic Curve Cryptography (ECC) is a form of public key cryptography best suited for constrained environments of embedded devices where resources like memory and processing power are very limited [3,4].

In this paper, mutual authentication scheme for embedded devices and cloud severs based on ECC has been proposed. The proposed protocol ensures mutual authentication between embedded device and cloud service provider using Hyper Text Transfer Protocol (HTTP) cookies. In Section 2, the operating environment of embedded devices connecting to cloud has been discussed. In Section 3 of the paper, the related work and security issues in collaborating embedded devices with cloud has been discussed. In Section 4, the preliminaries of ECC have been discussed. In Section 5, a novel ECC based mutual authentication protocol between the embedded device and cloud server has been proposed. In Section 6, security analysis based on an attack model has been done. In Section 7, cost and functionality analysis of the protocol has been discussed. The protocol has been formally verified using Automated Validation of Internet Security Protocols and Applications (AVISPA) tool. The results have been presented in Section 8. Lastly, Section 9 concludes the paper.

2. Embedded cloud computing: operating environment

Embedded systems have become an integral and indispensable part of everyone's daily life. Embedded systems range from portable devices such as digital watches and MP3 players to complex systems like traffic lights, factory controllers, hybrid vehicles and avionics. Unlike a general-purpose personal computer, an embedded system performs one or few predefined tasks that have specific requirements and limited field configuration capability. Since the system is dedicated to specific tasks, design engineers are liable to optimize it in order to reduce the size and cost of the product. Therefore, embedded systems have limited resources available in terms of memory, CPU, screen size, limited set (or absence) of key inputs and diskless operations. These parameters play a crucial role in the design, development and testing of such systems so that it can be bound to a relatively static and simple functionality device. Cloud computing is a computing paradigm that uses Internet and central remote servers to maintain and compute multiple data applications. The latest innovations in cloud computing is to make all business applications more mobile and collaborative. Embedded devices can leverage cloud computing to expand their functionalities. Many applications in embedded systems require huge memory and processing power necessary to run complex algorithms that can generate certain results. When cloud connectivity is provided to embedded systems, the later can use resources of cloud to remotely resolve complex algorithms which reduces power consumption in embedded devices. In this way, with few resources great results can be obtained using "external intelligence" stored in cloud. The demand for Internet connected products is growing as Internet is becoming the most cost effective way of remotely monitoring and controlling embedded systems. Internet of Things (IoT) is the name used to depict a scenario where many devices are using the resources of a network without human intervention [5]. As Internet has grown rapidly, it has become the world's low cost network allowing data to be passed easily across continents. Though the embedded system applications are still growing, Internet connected embedded systems is the next step in near future. Embedded systems are generally at remote locations from people that operate them at far and distant places. In such cases, tasks like monitoring their operation, checking their performance, collecting data or upgrading application software can be a costly and time consuming process. In such a scenario, functionalities of embedded systems can be extended with cloud based data storage and computing capabilities. Also, some applications could get great benefits if they could remotely report their status, get remote data to process or even send remote messages to have their administrator informed about some incidents. However, easier said than done, security undoubtedly is the major concern while getting connected to cloud. Cloud security refers to a broad set of policies, technologies and controls deployed to protect data, applications and associated infrastructure of cloud computing. Unauthorized access raises privacy and confidentiality concerns for embedded systems using cloud computing. Security issues related to embedded devices connecting with cloud have discussed in the next section.

3. Security issues and related work

Authentication is the process of identifying legitimate entity of a particular web application. Authentication plays the most important role in successful integration of embedded devices and cloud computing services. Multitenant architecture of cloud encourages the hackers for cybercrime. Survey conducted by International Data Corporation (IDC) in 2008–2009 showed that many organizations were adopting cloud computing as it provides low cost solutions for its users [6]. Security of the information in cloud computing paradigm is still a major concern for them. There have been many cases of security attacks on well known cloud computing providers such as Amazon Web Services (Amazon S3), Google (Gmail, App Engine) and Salesforce.com [7]. In general, the major security parameters in cloud computing are authentication, confidentiality, availability, integrity and non repudiation. Researchers are continuously making efforts to develop such solutions that cater to the security needs of cloud. Recently, cloud computing services have gained a lot of popularity worldwide among business enterprises. Amazon Elastic Compute Cloud (EC2) [8] and Elastic Block Store (EBS) [9] are used to provide both storage

services and cloud computing to their users. A remote user password is used to login to a Window instance in EC2. Microsoft Window Azure [10] also provides cloud computing services in a similar manner. Researchers are trying to shift applications on cloud platform without compromising on security of these applications. He et al. [11] migrated NASA climate prediction application to Amazon EC2. An application called Nearby Supernova Factory was migrated by Jackson et al. [12] to a cluster on Amazon EC2. GoGrid cloud and IBM cloud are other examples providing cloud computing services. Security turns out to be the biggest challenge in successful implementation of any cloud service. The main aspect of security in cloud computing services is to provide authorized access to legitimate entities of the cloud. Researchers need to develop such protocols that only legitimate entities should be given access to the services and data of cloud whereas illegal access should be denied at any cost.

Cookie technology is one of the most innovative features that have made the web stateful. Recently many authentication schemes have been proposed using encrypted cookies. In 2008, Lei et al. [13] proposed a virtual password concept based on randomized linear functions that involve a small amount of human computing to secure user's password in an online transaction. They analyzed that their scheme defends phishing, key logger and social engineering attacks. In 2008, Wu et al. [14] proposed a Single Sign-On (SSO) anti-phishing technique based on encrypted cookie that defeats phishing attacks. It encrypts the sensitive data with server's public key and stores this cookie on the user's machine. This Encrypted Cookie Scheme (ECS) has an advantage that the user can ignore SSL indicator in online transaction procedure. Microsoft's Passport (Window Live ID) initiative [15] is a cookie-based password management system. This service authenticates the user to different websites that are under the control of this centralized system. The main limitation of this approach is that the users have to trust centralized server. Also, it requires Web administration changes on sites that use this system for its authentication. In 2009, Sood et al. [16] proposed a cookie based single password anti-phishing protocol that is secure against different possible attacks. In this protocol, the user's machine browser generates a dynamic identity and a dynamic password for each login request to the server. In 2011, Sood et al. [17] proposed an inverse cookie-based virtual authentication protocol in which the cookies are not stored in trustworthy machines; instead they are stored on those machines from where the user failed to log in.

The security requirements for large number of connected embedded devices are distinct on account of their limited memory, constrained middleware and low computing power. Unfortunately, many networked embedded systems lack robust encryption to protect sensitive information. This could be due to resource limitations, cost restrictions or design limitations. Even the most encrypted secure protocols have the potential to be hacked using the processing power of cloud computing servers. Regardless of the reason, lack of robust encryption can lead to disastrous consequences. Intruders or malicious insiders could read, intercept, modify or remove communications. Insufficient cryptographic protection can lead to compromises, many of which are not apparent at the time of system design. A prudent embedded system designer must consider the implications of intercepted, deleted, modified and forged information from all components of a network system and take appropriate steps to protect the system against such attacks. Use of cloud and centralized servers can increase the risk of server harboring spying agents, password stealers or other cyber criminals. Therefore, we need protocols that are robust and cannot be forged by the hackers. Elliptic Curve Cryptography (ECC) offers better security at minimum key size and is the most cost-effective solution to implement security in embedded devices having constrained environment [18]. Recently, many authentication protocols have been proposed that are based on ECC. In 2011, Kalra and Sood [19] had done a detailed survey of ECC based protocols. ECC also turns out to be the best choice among other public key cryptographic techniques for achieving mutual authentication between smart devices and servers. ECC based authentication protocols applicable for smart devices proposed by Wu et al. [20], Tian et al. [21], Abichar et al. [22] had various limitations. Protocol proposed by Wu et al. [20] only supports user authentication by the server. This is unsafe as an attacker can impersonate a server to obtain information from the user. On the other hand, protocols proposed by Tian et al. [21] and Abichar et al. [22] provide mutual authentication using certificates. Certification schemes lead to increase in cost as the server and users have to perform additional computations to verify each other's identity. ECC based authentication protocols for smart devices have also been proposed by Yang et al. [23], Hafizul et al. [24] and Debiao et al. [25], Ray et al. [26], Granjal et al. [27], Jiang et al. [28]. These proposed protocols are based on different forms of ECC, using concepts ranging from time stamps to certificate based mutual authentication. A light weight attribute based encryption scheme using ECC for IoT has been proposed by Yao et al. [29] in 2014. In 2014, an authentication scheme based on ECC for RFID systems for IoT networks, was developed by Moosavi et al. [30]. Another scheme for IoT based on ECC was proposed by Liao et al. [31] using ID verifier transfer protocol in 2014. A survey of RFID authentication protocols for IoT in health care environment based on ECC has been done by Debiao et al. [2] in 2014, where they have compared the computational and communication cost of both the tag and the server side. In 2014, an authentication and key agreement protocol for heterogeneous ad hoc wireless sensor networks and IoT was proposed by Muhamed et al. [32]. Recently, in 2015, Nguyen et al. [33] conducted an extensive survey of protocols proposed for IoT networks. A detailed comparison, ranging from protocols based on symmetric cryptography to protocols based on asymmetric cryptography has been done by the authors. Researchers have proposed novel authentication schemes where wireless sensor networks are integrated in IoT networks. A top-down utility paradigm for IoT and cloud by using sensor networks and mobile devices was proposed by Salvotore et al. [34] in 2015. In 2015, Persson et al. [35] proposed a framework for merging IoT and cloud in a unified programming model that would make the communication efficient between the device and cloud. In this paper, we propose an ECC based mutual authentication protocol for IoT devices and cloud servers using encrypted cookies. A comparison of the proposed protocol with related protocols in Section 7.4, shows that the proposed protocol is very efficient and has least computation cost among other protocols.

Table 1
Comparison of ECC and RSA based on key size for same security levels, supplied by NIST (National Institute of Standards and Technology) [36].

ECC key size (bits)	RSA key size (bits)	Key size ratio
163	1024	1:16
256	3072	1:12
384	7680	1:20
512	15360	1:30

4. Preliminaries of Elliptic Curve Cryptography

For providing the same level of security, ECC uses much smaller key sizes and ensures higher levels of security compared to other asymmetric techniques. The benefits are more substantial for larger key sizes, i.e. a 256-bit symmetric key must be protected by more than 15,000-bit RSA, while an ECC asymmetric key size of only 512 bits provides equivalent security [18]. The reduced key size of ECC leads to obvious cost savings. The use of smaller key size also enables the design of more compact implementations and faster cryptographic operations that run on smaller chips. This leads to less heat production and reduced power consumption which is beneficial for resource-constrained systems. Thus, ECC is an emerging cryptographic technique and embedded implementations of ECC are now being designed and incorporated into systems. Table 1 shows the comparison between ECC and RSA based on key size for same security level [36].

The security of any cryptographic system is directly proportional to the relative complexity of underlying mathematical problem. An algorithm is said to be a polynomial time algorithm if its time complexity is upper bounded by a polynomial expression of the form $T(n) = O(n^k)$, where n is the size of input and k is a non negative integer. These algorithms are mathematically easier to solve in comparison to exponential time algorithms. The security of ECC depends on the difficulty of solving discrete logarithm problem over the points on an elliptic curve, i.e. Elliptic Curve Discrete Logarithm Problem (ECDLP). The best known method to solve ECDLP (Pollard's rho algorithm) [37] is fully exponential and substantially smaller key sizes are used to obtain equivalent security. Therefore, ECC is better suited for smart devices that operate in constraint environments [37].

4.1. ECDLP: the hard problem

Elliptic Curve Discrete Logarithm Problem can be stated as follows. P and Q are two points on an elliptic curve and kP represents the point added to itself k times, where k is a scalar, such that $kP = Q$. For given P and Q, it is computationally infeasible to obtain k, if k is sufficiently large. k is the discrete logarithm of Q to the base P.

The mathematical operations of ECC are defined over the elliptic curve equation:

$$\mathbf{y^2 = x^3 + ax + b}, \quad \text{where } \mathbf{4a^3 + 27b^2 \neq 0}.$$

The elliptic curve is set of solutions (x, y) which satisfy the above equation. Each value of 'a' and 'b' gives a different elliptic curve. All points (x, y) which satisfies the above equation plus a point O at infinity lies on the elliptic curve.

4.2. Mathematical operations involved in ECC

Point multiplication.

In point multiplication, a point P on the elliptic curve is multiplied with a scalar k using elliptic curve equations to obtain another point Q on the same elliptic curve, i.e. $kP = Q$. Point multiplication is achieved by two basic elliptic curve operations:

- *Point addition*, adding two points J and K on the curve to obtain another point L on curve, i.e., $L = J + K$.
- *Point doubling*, adding a point J on the curve to itself to obtain another point L on the curve, i.e. $L = 2J$.

Here is a simple example of point multiplication. Let P be a point on an elliptic curve. Let k be a scalar that is multiplied with the point P to obtain another point Q on the curve, i.e. to find $Q = kP$.

If $k = 23$ then $kP = 23.P = 2(2(2(2P) + P) + P) + P$.

Thus, point multiplication uses point addition and point doubling repeatedly to give the result. The above method is called 'double and add' method for point multiplication [37]. The order of the curve is the number of points lying on the curve and is represented by $|E|$ where E represents the elliptic curve. A point O is said to be a point at infinity and is the identity element in addition over elliptic curve as for a point P we have $P + O = P$. A point on an elliptic curve, if repeatedly added to itself will eventually reach O, the point at infinity. The number of times a point can be repeatedly added to itself until it reaches infinity is called order of the point. The order of every point on the curve is co-factor of order of the curve.

5. Proposed protocol

In this section, we propose an ECC based authentication protocol for embedded devices that are HTTP clients. There are various authentication protocols for smart devices, but the idea of using HTTP cookies for smart device authentication is

Table 2
Notations used in the protocol.

D_i	Embedded device
S	Cloud server
ID_i	Identity of the device D_i
P_i	Password of the device D_i
R_i	Random number generated by the server
N_1, N_2	Random numbers generated for ECC parameters
$H()$	One-way hash function
X	Private key of the server based on ECC
Z_p	Finite field group
p	Large prime number of the order $>2^{160}$
G	Generator point of a large order n
CK	Cookie information
EXP_TIME	Expiration time of the cookie
$\|$	Concatenation
$\oplus$	XOR operation

novel. The embedded device needs to be configured with TCP/IP protocol stack in order to act as a HTTP client. HTTP is based on a simple client/server protocol where the HTTP server and client communicate via a TCP connection. Today, almost every personal computer offers necessary assistance for this protocol and this status is becoming valid for embedded devices also. Most of the embedded devices in market are configured with a Web browser and can act as HTTP clients. For those embedded devices which do not have any user interface and are deployed in field, organizations are providing specialized software. Using the specialized software, the devices can communicate as HTTP clients to a corresponding cloud server which is HTTP enabled (HTTP server). This implies that machine-to-machine (M2M) communication is also possible using the proposed protocol with no human intervention. This is a novel authentication protocol, based on HTTP cookies, designed for embedded devices working in constraint environments and cloud servers. Table 2 denotes the notations used in proposed protocol.

Our protocol consists of three phases.

a. **Registration Phase**: In this phase, the embedded device registers itself with the cloud server and server stores a cookie on embedded device.
b. **Pre-computation and Login Phase**: When device wants to connect with the server, it sends a login request in this phase.
c. **Authentication Phase**: In this phase, the embedded device and cloud server mutually authenticate each other using ECC parameters.

Before the system begins, server S selects an elliptic curve equation $y^2 = x^3 + ax + b$ over Z_p where Z_p ($p > 2^{160}$) is the finite field group. Server selects two field elements $a, b \in Z_p$, where a and b satisfy the condition $4a^3 + 27b^2 \pmod p \neq 0$. Let G be the base point of the elliptic curve with a prime order n ($n > 2^{160}$) and O be the point at infinity such that $n \times G = O$. The server selects a random number X as its private key. Fig. 1 shows the workflow of the protocol.

5.1. Registration phase

In order to register with the cloud server S, the embedded device D_i sends a unique ID_i to the server. On receiving this request, the cloud server generates a unique password P_i for every device D_i.

Step 1: Embedded Device $D_i \rightarrow$ Server S: ID_i, Server S generates P_i

The server selects a unique random number R_i for every device and generates a cookie $\text{CK} = H(R_i|X|\text{EXP\_TIME}|ID_i)$ where X is the private key of the server and stores the cookie on the embedded device as ECC point $\text{CK}' = \text{CK} \times G$. The server also calculates the security parameters $T_i = R_i \oplus H(X)$, $A_i = H(R_i \oplus H(X) \oplus P_i \oplus \text{CK}')$ and stores $A_i' = H(R_i \oplus H(X) \oplus P_i \oplus \text{CK}') \times G$, T_i corresponding to the identity ID_i of the device D_i in its database. The server itself stores the expiration time of the cookie EXP_TIME corresponding to a particular embedded device's identity. When the cookie expires, the expiration time is updated to $\text{EXP\_TIME}'$ and cookie is updates as $\text{CK} = H(R_i|X|\text{EXP\_TIME}'|ID_i)$.

Step 2: Server $\rightarrow$ Embedded Device D_i: Cookie CK'

5.2. Pre-computation and login phase

Before every login, the device selects a random number N_1 and calculates an ECC point $P_1 = N_1 \times G$ and stores it in its memory.

Step 1: Embedded Device Calculates ECC point P_1.

In order to login with the cloud server, the device calculates the ECC point $P_2 = H(N_1 \times \text{CK}')$ sends the P_1, P_2 and its ID_i to the server.

Step 2: Embedded Device $\rightarrow$ Server: ID_i, P_1, P_2.

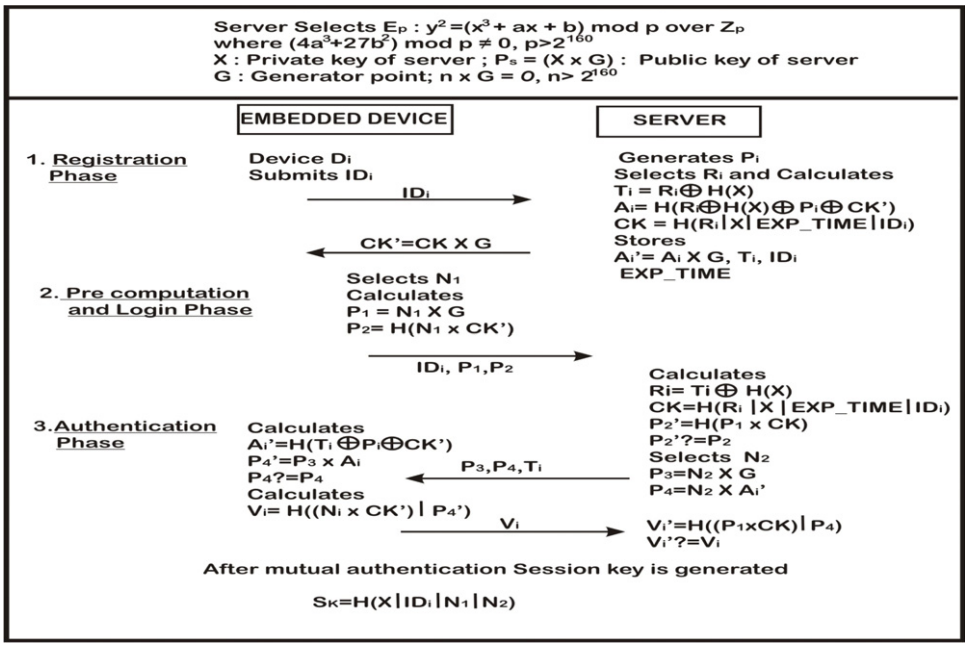

Fig. 1. Workflow of the protocol.

5.3. Authentication phase

After receiving the parameters on login request, the cloud server calculates cookie information $\mathrm{CK} = H(R_i|X|\mathrm{EXP\_TIME}|ID_i)$ by calculating the random number R_i from T_i using its private key X as $R_i = T_i \oplus H(X)$ and using its private key, identity of the device ID_i and expiration time EXP_TIME. It then calculates the point $P_2' = H(P_1 \times \mathrm{CK})$. The cloud server then checks whether the value of P_2' is equal to the received value of P_2. In case they are equal, the server proceeds to the next step otherwise it terminates the session.

Step 1: S checks $P_2'? = P_2$.

Then the server selects a random number N_2 and calculates the ECC point $P_3 = N_2 \times G$, $P_4 = N_2 \times A_i'$ and sends P_3, P_4 and T_i to the embedded device.

Step 2: $S \rightarrow$ Embedded Device D_i: P_3, P_4 and T_i.

The device then calculates $A_i = H(T_i \oplus P_i \oplus \mathrm{CK}')$ and calculate ECC point $P_4' = P_3 \times A_i$ and compares the value of P_4' with the received value of P_4.

Step 3: Embedded Device D_i checks $P_4'? = P_4$.

Then, the embedded device calculates $V_i = H((N_1 \times \mathrm{CK}')|P_4')$ and sends V_i to the server. The server calculates $V_i' = H((P_1 \times \mathrm{CK})|P_4)$ and compares the value to the received value of V_i to authenticate the device.

Step 4: Server checks $V_i'? = V_i$.

After mutual authentication between the embedded device and the cloud server, both the entities agree on a common session key $SK = H(X|ID_i|N_1|N_2)$. Afterwards all the subsequent messages communicated between the device and cloud server are XOR^{ed} with this session key.

6. Security analysis

In this never ending race between the developers and hackers, the later will always try to find out ways to intrude a system seeking an unauthorized access. An attacker may try to get access to the cloud server in place of a legitimate embedded device. In this section, the security of the proposed scheme by considering a formal attack model has been discussed. An attack model or attack types, in general, specify how much information a cryptanalyst or a hacker has access to while cracking an encrypted message. For correct analysis, an efficient and convincing formal methodology is required to evaluate the proposed scheme.

6.1. Attack model

The adversary can engage in various illegal ways to acquire sensitive information of a user. The following is the definition of attack model used by an adversary to acquire information from the system.

- Eavesdropping: The communication messages between the server and embedded device travel through insecure channel and the adversary can acquire the user's secret information via eavesdropping or initiate another attack by using the eavesdropped message.
- Traffic analysis: Traffic analysis is a method to analyze the messages eavesdropped during communication between the device and the server. By analyzing this information, the attacker can acquire information needed to authenticate the device to the server. An attacker can perform brute-force attacks using traffic analysis tools such as Flowd [38] and pcNetFlow [38]. This type of attack is also called brute force attack.
- Replay attack: A very common attack in which an adversary transmits a message obtained by eavesdropping on a regular communication between server and a device during authentication process.
- Man-in-the-middle attack: This is similar to replay attack. It this attack an adversary impersonates a legitimate device by transmitting response message that is obtained from a device by impersonating a legitimate server.
- Cookie theft attack: After obtaining a smart device and physically acquiring the cookie stored on it, the adversary can counterfeit or alter it.
- Offline dictionary attack: The attacker can record the transmitted messages and may try to gather security parameters from the previously transmitted messages.
- Leak of verifier attack: The attacker can break into the server and steal information stored in it. The adversary can use this information to calculate user's information and impersonate as a legitimate user.

6.2. System security requirements

In order to strengthen the security of the system, the system requirements that need to be considered while designing an authentication protocol has been discussed. The system requirements are defined in terms of mutual authentication, confidentiality, anonymity and forward secrecy.

- Mutual authentication: This is the most essential requirement as the device and the cloud server must authenticate each other for secure communication.
- Confidentiality: Confidentiality requires that the secret information is securely transmitted during all communications. Therefore, to ensure confidentiality, the device and server transmit encrypted information so that only they can recognize it.
- Anonymity: Anonymity is another important security requirement for privacy. Anonymity means that adversary cannot trace the device's information in place of a legitimate server. If the transmitted information cannot satisfy anonymity, an attacker can continuously trace the messages of a specific device and may get authenticated to the cloud server.
- Forward secrecy: It is essential that the previously transmitted information does not get traced using present transmission information. If the previous information of a specific device can be compromised, it constitutes a serious privacy issue.

6.3. Security analysis and system requirement analysis

6.3.1. Resistance to replay attack

Having intercepted previous communication, the attacker can replay the same message of receiver or sender to pass the verification of system. The malicious user tries to login as a legitimate user by listening and replaying the communication messages between client and the server. In our protocol, the elliptic curve point P_1 is generated using a unique nonce N_1 which is further used to generate P_2 using a one way hash function. When malicious device will try to login with a nonce N', the login will fail in the very step of login phase when the server authenticates the message. In case the attacker tries to use the previous values of P_1 and P_2 to calculate points $P_1'' = N' \times P_1$ and $P_2'' = N' \times P_2$, he will fail in the first step authentication phase of the protocol. This is because when the server will calculate the point P_2'' using the nonce as $P_2'' = N' \times H(P_1'' \times \mathrm{CK})$ and compare it with the expected value of $P_2' = H(N' \times G \times \mathrm{CK})$, they come out to be unequal. Hence, the protocol is secure against replay attack. Therefore, the malicious user cannot replay the login message. Further, if the attacker keeps a record of all messages passed during an authentication between a device and a server, it still cannot replay those messages to be authenticated by the server. It cannot generate $V_i = H((N_1 \times \mathrm{CK}')|P_4')$ because it does not have access to the value of N_1 or CK′. Moreover, it cannot reuse the same value of P_4 for replay as P_4 depends on the nonce N_3 generated randomly by the server for each authentication attempt.

6.3.2. Resistance to man-in-the-middle attack

In this type of attack, the malicious user may gather the communication parameters such as CK′, P_1, P_2, P_3, P_4. The malicious user cannot compute these ECC points as they have high entropy. It is impossible to solve ECDLP in real polynomial time [27]. Moreover, it is required for the malicious user to know the random nonce values of N_1 and N_2 that cannot be forged. Hence, proposed protocol is secure against man-in-the-middle attack.

6.3.3. Resistance to cookie theft attack

A malicious user may steal the cookie information from an embedded device and then try to login to the cloud server S using this information. But in our protocol, the cookie information $\mathrm{CK} = H(R_i|X|\mathrm{EXP\_TIME}|ID_i) \times G$ is stored as an ECC point

and therefore cannot be manipulated to get the correct parameters necessary to login. Moreover, our protocol does not send the cookie CK to the server in any communication message. So even if the malicious user is able to get the cookie CK, it is practically of no use to him. The reason is that under any circumstances the malicious user cannot log onto the server with the help of stolen cookie.

6.3.4. Resistance to eavesdropping

Impersonation is when the attacker forges the authentication messages of a legitimate user and then tries to modify the login request into ID_i^*, P_1^*, P_2^* from ID_i, P_1, P_2. But in our proposed scheme, the malicious user cannot obtain random numbers N_1 and N_2. Moreover, the entropies of P_1 and P_2 are very high and therefore the ECC points cannot be forged. Therefore, this type of attack fails in very first step of authentication phase.

6.3.5. Resistance to brute force attack

In order to launch a brute force attack, the malicious user first obtains the parameters from communication messages P_1, P_2, P_3, P_4 and T_i. Even if the malicious user successfully records this information, he will still not be able to find the correct password P_i using the brute force attack as he cannot know the server's secret key X and there is no way to guess the random numbers N_1 and N_2. Hence, our scheme is secure against brute force attack.

6.3.6. Resistance to offline dictionary attack

In this type of attack, the malicious user first records the communication messages and then attempts to guess the legitimate user's password from them. But in the proposed protocol, it is impossible to calculate the password in real polynomial time from the recorded communication messages that are high entropy ECC points. Therefore, the protocol is secure against offline dictionary attack.

6.3.7. Resistance to leak of verifier attack

In this type of attack, the malicious user breaks into the system and steals the server's database information. Then he may use that information to calculate the vital information such as legitimate user's password. In our protocol, the information stored in device's database is $\text{CK} = H(R_i|X|\text{EXP\_TIME}'|ID_i)$, $A_i' = H(R_i \oplus H(X) \oplus P_i) \times G$ which is an ECC point. So the malicious user cannot guess server's private key X or ECC point A_i'. Therefore, leak of verifier attack is not possible in our protocol.

6.3.8. Provides mutual authentication

The server checks the authenticity of the smart device by comparing the received value P_2 and the calculated value P_2'. In step 5.2 of the protocol, the device calculates P_2 using the equation $P_2 = H(N_1 \times \text{CK}')$, where $\text{CK}' = \text{CK} \times G$, and sends it to the server. The server then calculates value P_2' using the equation $P_2' = H(P_1 \times \text{CK})$, where $P_1 = N_1 \times G$. Thus, if the values are equal, the server successfully authenticates the user. The user authenticates the server by comparing the received value P_4 and calculated value P_4'. In step 5.3 of the protocol, server calculates P_4 using the equation $P_4 = N_2 \times A_i'$, where $A_i' = A_i \times G$ and sends it to the device. The device then calculates P_4' using the equation $P_4' = P_3' \times A_i$ where $P_3' = N_2 \times G$. If both the values are equal, the device successfully authenticates the server.

6.3.9. Provides confidentiality

The proposed protocol protects the information necessary for device authentication by using ECC points and hash functions. It ensures that only the authenticated device gets access to legitimate server. Further, the proposed protocol is secure against traffic analysis and eavesdropping and guarantees confidentiality by ensuring that the complexity of brute force attack is high.

6.3.10. Provides anonymity

The device can send messages for authentication to any server in its vicinity. If an attacker impersonating as a server comes in contact with it, the device will exchange initial messages with the attacker. According to the proposed protocol, attacker has no access to the cookie (CK') related to the device and cannot generate the correct value of $A_i' = H(T_i \oplus P_i \oplus \text{CK}') \times G$ and consequently generating an incorrect value of $P_4 = N_2 \times A_i'$. The received value of P_4 will not be verified with the expected value of $P_4' = P_3 \times A_i$, where $A_i = H(T_i \oplus P_i \oplus \text{CK}')$. Therefore, anonymity is maintained, as the device will not authenticate the attacker and the session will be dropped.

6.3.11. Provides forward secrecy

It is essential that the previously transmitted information cannot be traced using the present transmitted information of device. The proposed protocol prevents a malicious user from acquiring device information by providing confidentiality based on fresh value of nonce in every session. As the protocol prevents replay attack, the malicious user has no means to know the random numbers generated inside the device. Therefore, the protocol ensures forward secrecy by providing unpredictable variations in the past communication messages.

Table 3
Comparison of proposed protocol with related work.

Attributes	Protocols									
	Abichar et al. [22]	Hafizul et al. [24]	Debiao et al. [25]	Ray et al. [26]	Granjal et al. [27]	Jiang et al. [28]	Yao et al. [29]	Moosavi et al. [30]	Liao et al. [31]	Proposed protocol
Computation cost (server)	$2T_{ECM}+2T_{ECA}$	$3T_{ECM}+2T_{ECA}$	$6T_{ECM}+4T_{ECA}$	$14T_{ECM}$	$8T_{ECM}$	$6T_{ECM}$	$3T_{ECM}+1T_{SYM}$ [a]	$3T_{ECM}+2T_{ECA}$	$3T_{ECM}+5T_{ECA}$	$4T_{ECM}$
Computation cost (device)	$3T_{ECM}+2T_{ECA}$	$5T_{ECM}+2T_{ECA}$	$3T_{ECM}+2T_{ECA}$	$8T_{ECM}$	$4T_{ECM}$	$4T_{ECM}$	$3T_{ECM}+1T_{SYM}$ [a]	$3T_{ECM}+2T_{ECA}$	$3T_{ECM}+2T_{ECA}$	$3T_{ECM}$
Communication cost (bits)	800	1280	1216	1280	1216	1218	1280	336	1680	1280
Memory cost (device)	448	352	224	448	448	352	256	496	652	224
Certificate based authentication	Yes	No	No	Yes	No	Yes	No	Yes	No	No
Key agreement	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes
Clock synchronization problem	No	No	No	Yes	No	No	No	No	No	No
Attack resistance	No	No	No	No	No	No	Yes	Yes	No	Yes

[a] T_{SYM} is the time involved in one symmetric encryption/decryption operation.

7. Computation and communication cost analysis

An efficient authentication protocol must consider computation and communication cost while authenticating entities. Along with ECC, our protocol uses XOR operations and one-way hash functions, both of which are very inexpensive operations in cryptography. Our protocol is very secure and efficient as it is based on random nonce values. The protocol has no time synchronization problem as it does not use timestamps. While calculating the cost of the protocol, the identity ID_i, password P_i, nonce values (N_1, N_2), random number R_i and the security parameters T_i, R_i all are assumed to be 128 bits long. Also, the output of one way hash function is 128 bits and elliptic curve cryptosystem is ECC—224 bits. Let T_H, T_E, T_{ECM}, T_{ECA} be the time for one hashing operation, one exponential operation, one multiplication of a number over elliptic curve and elliptic curve point addition respectively. The comparison of the time complexity associated with these operations can be expressed as $T_E \gg T_{ECM} > T_{ECA} > T_H$. This is because the time taken to perform an exponential operation is much more (approx. 8 times) than the time taken to perform one elliptic point multiplication [20].

7.1. Communication cost of the protocol

Communication cost of a protocol is the cost involved in the transmission of security parameters. Let C1 be the cost of communication parameters involved in the authentication process. In the proposed protocol, the communication parameters are $[P_1, P_2, P_3, P_4, T_i, ID_i, V_i] = [4 \times 224 + 3 \times 128] = 1280$ bits.

7.2. Computation cost of the protocol

Let C2 be the cost of registration that includes the total time of all operations executed by the server S in the registration phase. In the proposed protocol, the cost of registration is $(3T_H + 2T_{ECM})$. Let C3 be the computation cost of the cloud server during the process of authentication. In proposed protocol, the total time spent by the servers is $(4T_H + 4T_{ECM})$. Let C4 be the cost of the time spent by the embedded device for computation during the authentication process. In the proposed protocol, this time is $(3T_H + 3T_{ECM})$.

7.3. Storage cost of the protocol

Let C5 be the memory needed by the embedded device and C6 be the memory needed by the server to store security parameters. In the proposed protocol, the embedded device stores cookie CK′ as ECC point so C5 = 224 bits. The cloud server stores (A'_i, ID_i) so C6 = 224 + 128 = 352 bits.

7.4. Functionality comparisons

A comparative study of various security and cost attributes of the proposed protocol is performed with related ECC based authentication protocols in Table 3. Comparisons show that the proposed protocol is very efficient and practical for embedded devices.

Hence, with a low computation and communication cost, the proposed protocol can be efficiently used for embedded devices that opt for using cloud computing services. This protocol provides mutual authentication between the device and cloud server at low cost.

```
role device(U,S          : agent,
            K,Ea         : symmetric_key,
            Hash         : hash_func,
            SND,RCV      : channel(dy))

played_by U
def=

local
State                                : nat,
Un1,Pass1,Ck,N1,Ga,Ck2               : text,
P1,P2,Ti,P3,P4,Da,Ai,P44,Vi           : text

const  e_m,g_m,h_m,a_m,b_m,d_d:protocol_id
init          State := 0

transition
1.State = 0 /\RCV(start) =|>
  State' := 2 /\Un1' := new()
              /\ Pass1' := new()
              /\ SND(U.{Un1'.Pass1'}_Ea.S)
              /\secret(Pass1',g_m,{U,S})
              /\secret(Un1',h_m,{U,S})
              /\witness(U,S,seq1,Un1'.Pass1')

2.State = 2 /\ RCV({Ck'}_K) =|>
  State' := 4 /\N1' := new()
              /\Ga' := new()
              /\ Un1' := new()
              /\P1' := exp(N1',Ga)
              /\Ck2' := {{Ck'}_K}_K
              /\ P2' := exp(N1,Ck2')
              /\SND({Un1'.P1'.P2'}_Ea)
              /\secret(Un1',e_m,{U,S})
              /\secret(P1',a_m,{U,S})
              /\secret(P2',b_m,{U,S})
              /\witness(U,S,seq2,Un1'.P1'.P2')
              /\request(U,S,req1,Ck')

3.State =4  /\ RCV({Ti'.P3'.P4'}_K)=|>
   State' :=6 /\ Pass1' := new() /\ N1':=new()
              /\ Ai' := Hash(xor(Ti',Pass1'))
              /\ P44' := exp(P3',Ai') /\ Ck':=new()
              /\ Vi':=Hash(exp(N1',Ck').P44')
              /\SND({Vi'}_K)
              /\secret(Vi',d_d,{U,S})
              /\witness(U,S,seq3,Vi')
              /\request(U,S,req2,Ti'.P3'.P4')
end role
```

Fig. 2. Role—device. (*Note*: The names of the parameters in the protocol have been changed in the code according to the syntax of the AVISPA tool.)

8. Formal verification of the protocol using AVISPA

The Automated Validation of Internet Security Protocols and Applications (AVISPA) [39] project was funded by the European Union. This project intended to develop a real working environment for error detection in security protocols by targeting the design of the protocols. In order to achieve this, High Level Protocol Specification Language (HLPSL) was developed. HLPSL is an expressive language for modeling communication and security protocols. HLPSL has been defined in such a way as to closely resemble a language for defining guarded transitions within a state-transition system and is equipped with constructs that allow modular specification of protocols. It supports symmetric and asymmetric keys, key-tables, hash functions, etc. Protocol specification in HLPSL is divided into roles that describe the actions of one single agent in a run of a protocol or sub-protocol. Also, HLPSL code must be open to automated formal analysis. This is achieved by a translation of HLPSL into the Intermediate Format. The HLPSL2IF translator automatically translates a HLPSL protocol specification provided by the user into an IF specification, which is then given as input to the different back-ends of the AVISPA tool [40]. Hence, the main goal in the design of the IF was to provide a low-level description of the protocol that is suitable for automatic analysis and yet this format should be independent from the analysis methods employed by the various back-ends. The back-ends are implemented using formal methods and theoretical axioms and are used to provide protocol falsification, bounded and unbounded verification. Two backs ends used for analysis of the proposed protocol are On-the-Fly Model-Checker (OFMC) and CL-based Attack Searcher (CL-Atse).

- **OFMC**: This back-end builds the infinite tree defined by the protocol analysis problem and executes different symbolic techniques to search the state space in a demand-driven way, i.e., on-the-fly. OFMC helps to detect attacks and verify

```
role server(U,S        : agent,
            K,Ea,Sk    : symmetric_key,
            Hash       : hash_func,
            SND,RCV: channel(dy))
played_by S
def=

local
State                                    : nat,
Un1,Pass1,Ra,Ai,Ai1,Xa,Ga,Ck,Vii         : text,
Exp_tm,P1,P2,P22,P3,P4,N2,Da,Ti,Vi       : text

const f_n,y_k,z_k,m_k,p_p,s_p : protocol_id
init State := 1

transition
1.State = 1 /\RCV(U.{Un1'.Pass1'}_Ea.S) =|>
  State' := 3 /\ Ra' := new()
              /\Xa' := new()
              /\Ga' := new()
              /\Exp_tm' := new()
              /\ Ti' := xor(Ra',Hash(Xa'))
              /\ Ai' := Hash(xor( xor(Ra',Hash(Xa')),Pass1'))
              /\ Ai1' := exp(Hash(xor(xor(Ra',Hash(Xa')),Pass1')),Ga')
              /\ Ck' := Hash(Hash(Ra'.Xa').Hash(Exp_tm'.Un1'))
              /\ SND({Ck'}_K)
              /\secret(Ck',f_n,{U,S})
              /\witness(S,U,seq4,Ck')
              /\request(S,U,req3,Un1'.Pass1')

2.State = 3 /\ RCV({Un1'.P1'.P2'}_Ea) =|>
  State' := 5 /\ Xa' := new()
              /\ Ti' := xor(Ra',Hash(Xa'))
              /\ Ra' := xor(Ti',Hash(Xa'))
              /\ Exp_tm' := new()
              /\ Ck' := Hash(Ra'.Xa'.Exp_tm'.Un1')
              /\ P22' :=exp(P1',Ck')
              /\ Ga' := new() /\ N2' := new()
              /\ P3' :=exp(N2',Ga') /\ Pass1' := new()
              /\ Ai' := Hash(xor( xor(Ra',Hash(Xa')),Pass1'))
              /\ P4' := exp(N2',Ai')
              /\SND({Ti'.P3'.P4'}_K)
              /\secret(Ti',y_k,{U,S})
              /\secret(P3',p_p,{U,S})
              /\secret(P4',s_p,{U,S})
              /\witness(S,U,seq5,Ti'.P3'.P4')
              /\request(S,U,req4,Un1'.P1'.P2')

3.State =5/\ RCV({Vi'}_K) =|>
   State' := 7 /\ P1':=new() /\ P4':=new()
               /\ Ck':=new()
               /\ Vii':=Hash(exp(P1',Ck').P4)

end role
```

Fig. 3. Role—server. (*Note*: The names of the parameters in the protocol have been changed in the code according to the syntax of the AVISPA tool.)

the correctness of the protocol for a bounded number of sessions but without limiting the number of messages that an intruder can generate [40–42].

- **CL-AtSe**: This back-end is used to detect the attacks on the protocol by using a set of constraints that are obtained by translating the security protocol specification written in Intermediate Format (IF). Detection of attacks and translation of protocol specifications that are designed based on the adversary's knowledge, are fully automated and are internally performed by the CL-AtSe model checker [40–42].

8.1. Protocol specification in AVISPA

The analysis of the protocol is performed by defining the protocol in HLPSL and testing it using AVISPA back-ends. This type of analysis is useful in locating design flaws and problems that would be very difficult and expensive to solve once the protocol has been deployed in real systems. Although this approach plays a very important role in the evaluation of security protocols by analyzing their design and security from a theoretical and formal point of view, the actual implementation results may vary. In order to analyze the protocol using AVISPA tool, the following steps are executed:

Step 1. The protocol is represented in the HLPSL specification.
Step 2. Using the translator HLPSL2IF, the HLPSL code is to be translated into IF.
Step 3. The translated IF specification is input to the back-end of the AVISPA tool.

In our protocol model described in HLPSL, there are two basic roles 'device' and 'server', which represent the agents *U* and *S* respectively. Here, we represent the HLPSL coding of 'device' role in Fig. 2 and 'server' role in Fig. 3.

```
role session(U,S          : agent,
             K,K4,K2,P : symmetric_key,
             Hash         :hash_func)

def=
local SA,SB,RA,RB :channel(dy)

composition
device(U,S,K,P,Hash,SA,RA) /\ server(U,S,K,K4,K2,Hash,SB,RB)


end role
```

Fig. 4. Role—session.

```
role environment()  def=

const k1,e_m,g_m,h_m, f_n,y_k,z_k,m_k,p_p,s_p,a_m,b_m :protocol_id,
seq1,seq2,seq3,seq4,seq5                               :protocol_id,
req1,req2,req3,req4                                   :protocol_id,
kab,kai,kib,kba,k11,k12,k13,k14,kia,kbi                :symmetric_key,
k15,k16,k17,k18,k19,k20,k21,k22                        :symmetric_key,
u,s                                                    : agent,
h                                                      : hash_func

intruder_knowledge= {u,s,kai,kia,kbi,kib}

composition
  session(u,s,kab,k11,k12,k13,h)
/\session(u,i,kai,k14,k15,k16,h)
/\session(i,s,kib,k17,k18,k19,h)

end role

goal

secrecy_of  k1,e_m,g_m,h_m, f_n,y_k,z_k,m_k,p_p,s_p,a_m,b_m,d_d
authentication_on seq1
authentication_on seq2
authentication_on seq3
authentication_on seq4
authentication_on seq5

authentication_on req1
authentication_on req2
authentication_on req3
authentication_on req4

end goal

environment()
```

Fig. 5. Role—environment.

After mutual verification is accomplished, a session is setup between device and server. In order to define the session of the protocol, composed roles are defined after defining the basic roles. Fig. 4 depicts the role 'session'.

In session segment, both the basic roles (Device, Server) are instanced with concrete arguments. 'Session' role contains global constants and a composition of other roles, where the intruder may act as the role of a legitimate user. Then a top-level role 'Environment' is defined that is depicted in Fig. 5. The constant '*i*' is used to personify as an intruder. The intruder also participates in the execution of the protocol as a concrete session. The properties specified in 'Goal' section are also depicted in Fig. 5. The symbol (seq1) is the protocol ID used to authenticate the identity and password using predefined function of AVISPA library. WITNESS function authenticates the value transmitted over the channel safely to another role and check for various attacks. REQUEST function authenticates the value that the role receives safely by any other role. SECRET function is used to share the value in encrypted form among the roles corresponding to the protocol ID. The current version of HLPSL supports the standard authentication and secrecy goals and identifies various types of attacks over the communication channel.

8.2. Formal security analysis of the protocol

The proposed scheme is analyzed using OFMC and CL-AtSe back-ends and the results are shown in Figs. 6 and 7. Under this model, the intruder has full control over the network in such a manner that all messages sent by the agents corresponding to the roles are available to the intruder. The intruder may intercept, analyze and/or modify messages as long as he knows

```
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  /home/avispa/web-interface-computation/./tempdir/workfilevHfaiv.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 0.14s
  visitedNodes: 6 nodes
  depth: 3 plies
```

Fig. 6. Output of OFMC backend.

```
SUMMARY
  SAFE

DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
  TYPED_MODEL

PROTOCOL
  /home/avispa/web-interface-computation/./tempdir/workfilevHfaiv.if

GOAL
  As Specified

BACKEND
  CL-AtSe

STATISTICS

  Analysed   : 2 states
  Reachable  : 0 states
  Translation: 0.05 seconds
  Computation: 0.00 seconds
```

Fig. 7. Output of CL-AtSe backend.

the required keys. He can act as any agent and send any message to some other agent participating over the communication channel. The back-ends execute all possible security threats on the protocol as in this model intruder is given full control of the communication channel. The simulation results show that the proposed protocol is secure. Moreover, the mathematical security analysis based on the attack model also proves the security of the protocol. Hence, the protocol can be deployed safely for practical implementations.

9. Conclusion

An ECC based mutual authentication protocol for secure communication between embedded devices and cloud servers has been presented in this paper. Previously proposed schemes based on ECC either have high computation cost or do not satisfy all the essential security requirements. A formal security analysis based on attack model proves that the protocol is robust against all the security threats. Automated verification of the protocol using AVISPA tool has been performed. Results of the protocol show that the protocol is safe and is efficient in terms of computation cost. Besides having low computation cost, the proposed security protocol can be practically implemented with any of the embedded devices that are HTTP enabled. Also, the implementation of the protocol will expand the coverage of capabilities offered by IoT making them more reliable.

References

[1] M. Sascha, W. Sebastian, Secure communication in microcomputer bus systems for embedded devices, J. Syst. Archit. 54 (2008) 1065–1076.
[2] H. Debiao, Z. Sherali, An analysis of RFID authentication schemes for Internet of Things in health care environment using elliptic curve cryptography, IEEE Internet Things J. 2 (1) (2015) 72–83.

[3] A. Rahat, S.C. Mehrotra, A review on elliptic curve cryptography for embedded systems, Int. J. Comput. Sci. Inf. Technol. 3 (3) (2011) 84–103.
[4] M. Salas, A secure framework for OTA smart device ecosystem using ECC encryption and biometrics, in: Communications in Computer and Information Sciences (CCIS), Vol. 381, Springer-Verlag, 2013, pp. 204–218.
[5] M. Kranz, P. Holleis, A. Schmidt, Embedded interaction: Interacting with the Internet of Things, IEEE Internet Comput. 14 (2) (2010) 46–53.
[6] S. Subashini, V. Kavitha, A survey on security issues in service delivery models of cloud computing, J. Netw. Comput. Appl. 34 (1) (2011) 1–11.
[7] T.S. Chou, Security threats on cloud computing vulnerabilities, Int. J. Comput. Sci. Inf. Technol. 5 (3) (2013) 79–88.
[8] Amazon.com, Amazon elastic compute cloud, URL: http://aws.amazon.com/ec2/ (accessed June 2015).
[9] Amazon.com, Amazon elastic block store, URL: http://aws.amazon.com/ebs/ (accessed June 2015).
[10] Microsoft Windows Azure Platform, URL: http://www.microsoft.com/azure/default.mspx (accessed June 2015).
[11] Q. He, S. Zhou, B. Kobler, D. Duffy, T. McGlynn, Case study for running hpc applications in public clouds, in: High Performance Distributed Computing, ACM, 2010, pp. 395–401.
[12] K.R. Jackson, L. Ramakrishnan, K.J. Runge, R.C. Thomas, Seeking supernovae in the clouds: a performance study, in: High Performance Distributed Computing, ACM, 2010, pp. 421–429.
[13] M. Lei, Y. Xiao, S.V. Vrbsky, C.C. Li, Virtual password using random linear functions for online services. ATM machines and pervasive computing, Comput. Comm. 31 (18) (2008) 4367–4375.
[14] Y. Wu, H. Yao, F. Bao, Minimizing SSO effort in verifying SSL anti-phishing indicators, in: International Information Security Conference, IFIP TC 11, vol. 278, 2008, pp. 47–61.
[15] Microsoft Passport, 2009. Available from: http://www.passport.net/ (accessed June 2015).
[16] S.K. Sood, A.K. Sarje, K. Singh, Dynamic identity-based single password anti-phishing protocol, Secur. Commun. Netw. 4 (4) (2009) 418–427.
[17] S.K. Sood, A.K. Sarje, K. Singh, Inverse cookie-based virtual password authentication protocol, Int. J. Netw. Secur. 12 (3) (2011) 292–302.
[18] K. Imamoto, K. Sakurai, Design and analysis of Diffie–Hellman based key exchange using one-time ID by SVO logic, Electron. Notes Theor. Comput. Sci. 135 (2005) 79–94.
[19] S. Kalra, S. Sood, Elliptic curve cryptography: survey and its security applications, in: International Conference on Advances in Computing and Artificial Intelligence, ACM, 2011, pp. 113–117.
[20] S.T. Wu, J.H. Chiu, B.C. Chieu, ID-based remote authentication with smart cards on open distributed system from elliptic curve cryptography, in: IEEE International Conference on Electro Information Technology, 2005.
[21] X. Tian, D.S. Wong, R.W. Zhu, Analysis and improvement of authenticated key exchange protocol for sensor networks, IEEE Commun. Lett. 9 (11) (2005) 970–972.
[22] P.E. Abichar, A. Mhamed, B. Elhassan, A fast and secure elliptic curve based authenticated key agreement protocol for low power mobile communications, in: International Conference on Next Generation Mobile Applications, Services and Technologies, 2007, pp. 235–240.
[23] J.H. Yang, C.C. Chang, An ID based remote mutual authentication with key agreement scheme for mobile devices on elliptic curves cryptosystems, Comput. Secur. 28 (3–4) (2009) 138–143.
[24] S.K. Hafizul, G.P. Biswas, A more efficient and secure ID-based remote mutual authentication with key agreement scheme for mobile devices on elliptic curve cryptosystems, J. Syst. Softw. 84 (11) (2011) 1892–1898.
[25] H. Debiao, C. Jianhua, H. Jin, A ID-based client authentication with key agreement protocol for mobile client–server environment on ECC with provable security, Inform. Fusion 13 (3) (2012) 223–230.
[26] S. Ray, G.P. Biswas, Establishment of ECC based initial secrecy usable for IKE implementation, in: World Congress on Expert Systems, 2012, pp. 1–9.
[27] J. Granjal, E. Monteiro, J. Silva, End to end transport layer security for Internet integrated sensing applications with ECC public-key authentication, in: IFIP Networking Conference, 2013, pp. 530–535.
[28] R. Jiang, C. Lai, J. Luo, X. Wang, H. Wang, EAP based group authentication and key agreement protocol for machine type communication, Int. J. Distrib. Sens. Netw. (2013) http://dx.doi.org/10.1155/2013/304601.
[29] X. Yao, Z. Chen, Y. Tian, A lightweight attribute-based encryption scheme for the Internet of Things, Future Gener. Comput. Syst. (2014) http://dx.doi.org/10.1016/j.future.2014.10.010.
[30] S.R. Moosavi, E. Nigussie, S. Virtanen, J. Isoaha, An elliptic curve-based mutual authentication scheme for RFID implant system, in: International Conference on Ambient Systems, Network and Technologies, vol. 32, 2014, pp. 198–206.
[31] Y.P. Liao, C.M. Hsiao, A secure ECC-based RFID authentication scheme integrated with ID-verifier transfer protocol, Ad Hoc Networks 18 (2014) 133–146.
[32] M. Turkanovic, B. Brumen, M. Holbl, A novel user authentication and key agreement scheme for heterogeneous ad hoc wireless sensor networks based on the Internet of Things notion, Ad Hoc Networks 20 (2015) 96–112.
[33] K.T. Nguyen, M. Laurent, N. Oualha, Survey on secure communication protocols for the Internet of Things, Ad Hoc Networks 32 (2015) 17–31.
[34] S. Distefano, G. Merlino, A. Puliafito, A utility paradigm for IoT: The sensing cloud, Pervasive Mob. Comput. 20 (2015) 127–144.
[35] P. Persson, O. Angelsmark, Calvin—Merging cloud and IoT, in: International Conference on Ambient Systems, Network and Technologies, vol. 52, 2015, pp. 210–217.
[36] B. Hancock, Security views, Comput. Secur. 18 (7) (1999) 553–564.
[37] W.J. Caelli, E.P. Dawson, S.A. Rea, PKI, elliptic curve cryptography and digital signatures, Comput. Secur. 18 (1999) 47–66.
[38] M. Ayedemir, L. Bottomley, M. Coffin, C. Jeffries, P. Kiessler, K. Kumar, W. Ligon, J. Marin, A. Nilsson, J. McGovern, A. Rindos, K. Vu, S. Woolet, A. Zaglou, K. Zhu, Two tools for network traffic analysis, Comput. Netw. 36 (2–3) (2001) 169–179.
[39] L. Vigano, Automated security protocol analysis with the AVISPA Tool, Electron. Notes Theor. Comput. Sci. 155 (2006) 61–86.
[40] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Companga, J. Cuellar, P.H. Drielsma, P.C. Heam, O. Kouchnarenko, J. Mantovani, S. Modersheim, D. Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Vigano, L. Vigneron, The AVISPA Tool for the Automated Validation of Internet Security Protocols and Application, in: LNCS, vol. 3576, Springer-Verlang, 2005, pp. 281–285.
[41] A. Izquierdo, J.M. Sierra, J. Torres, An analysis of conformance issues in implementations of standardized security protocol, Comput. Stand. Interfaces 31 (2009) 249–251.
[42] L. Chao, M. Maode, M. Jianfeng Ma, Z. Yaoyu, A novel three-party authenticated key exchange protocol using one-time key, J. Netw. Comput. Appl. 36 (2013) 298–503.