An Improved Authenticated Key Agreement Protocol for IoT and Cloud Server

Yongliu Ma[1,2], Yuqian Ma[1,2], and Qingfeng Cheng[1,2(✉)]

[1] State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou, China
qingfengc2008@sina.com
[2] Strategic Support Force Information Engineering University, Zhengzhou 450001, China

Abstract. The internet of things (IoT) is a complex network system with applications in all walks of life. However, there are various risks in the process of information transmission between IoT devices and servers. Recently, research on the security of authenticated key agreement (AKA) protocols in the IoT environment has gradually increased. Iqbal et al. proposed an AKA protocol between IoT and cloud servers and proved that it was secure under the eCK model. This paper shows that the Iqbal et al.'s protocol has two security flaws, which are resisting ephemeral key leakage attack and key compromise impersonation attack, and proposes a new AKA protocol in the IoT environment. Through the security proof and formal analysis, it is proved that the new protocol is secure under the eCK model. Comparing the protocol proposed in this paper with other similar protocols, it is found that the protocol in this paper achieves a balance between security performance and communication consumption.

Keywords: IoT · AKA protocol · eCK model · Scyther formal tool

1 Introduction

1.1 Related Work

The internet of things (IoT) is a network that can realize the interconnection of any object anytime, anywhere. With the popularization of IoT smart devices, IoT plays an increasingly important role in life [1]. The IoT has a wide range of applications in infrastructure fields such as smart cities, smart malls, smart transportation, smart medical care, and smart logistics [2–4]. However, there may be attacks by malicious adversaries in the process of transmitting information between the devices and the servers [5]. Therefore, one of the foundations for building security services is authentication and session key agreement between IoT devices and servers. Many protocols for securing communication between IoT devices and cloud servers have been proposed.

Q. Jiang et al. (Eds.): SPNCE 2022, LNICST 496, pp. 15–31, 2023.
https://doi.org/10.1007/978-3-031-30623-5_2

Khelf et al. [6] proposed an IoT-oriented AKA micro-protocol to solve the security problem in wireless sensors, which could resist various types of attacks while reducing the computational cost. Qi et al. [7] proposed a two-factor AKA protocol based on elliptic curve cryptography for wireless sensor networks in the context IoT, and through heuristic security analysis showed that the protocol could resist various known attacks. Peng et al. [8] proposed an efficient protocol for IoT devices, which avoided the pairing operation of the client through an unbalanced computation method and proved the security of the protocol. Recently, Rostampour et al. [9] proposed an AKA protocol between IoT edge devices and cloud servers based on elliptic curve cryptography. Iqbal et al. [10] proved that the protocol proposed by Rostampour et al. was insecure under the eCK model, then they proposed a new AKA protocol called ITGR protocol below, at the same time, they used Scyther tool and BAN logic to prove that the protocol was secure under the eCK model. However, in this paper, we prove that the ITGR protocol cannot resist the ephemeral key leakage attack and key compromise impersonation attack, then we propose a new AKA protocol between IoT devices and cloud servers based on the ITGR protocol, called eITGR protocol. Finally, we prove that our protocol is secure under the eCK model.

1.2 Contribution

The contribution of this paper consists of the following four parts:

(i) This paper analyzes the ITGR protocol between IoT devices and cloud servers based on elliptic curve cryptography, and points out that it cannot resist ephemeral key leakage attack and key compromise impersonation attack.
(ii) A new AKA protocol between IoT devices and cloud servers is proposed, which makes up for the security defects of the ITGR protocol.
(iii) Use the security proof to prove that the eITGR protocol is secure under the eCK model, and confirm it through the Scyther tool.
(iv) By comparing the security properties and communication consumption of eITGR and similar protocols, the advantages of the eITGR protocol in terms of security and communication efficiency are shown.

1.3 Organization

The content of this paper is arranged as follows. Section 1 introduces the development status of the IoT, the research status of AKA protocol at home and abroad, and briefly introduces the research content and structure of this paper. Section 2 presents the basics of mathematics and cryptography applied during protocol design and analysis. A review of the ITGR protocol and analysis of security flaws are in Sect. 3. Section 4 proposes a new AKA protocol in the IoT environment, and its formal security proof, security analysis using Scyther tool and security properties analysis are shown in Sect. 5. The security properties, computation and communication cost of the protocol proposed in this paper and other similar protocols are compared in Sect. 6. Conclusion is given in Sect. 7.

2 Preliminaries and Security Model

The notations used in the paper are shown in Table 1.

Table 1. Notations used in the paper.

Notations	Description
U	User
S	Server
ID_u	Identity of U
t_u	Ephemeral private key of U
t_s	Ephemeral private key of S
x_s	Long-term private key of S
k	The security parameter
G	The cyclic additive group
p, q	The large prime
P	A generator of G
T	The timestamp
$H_i (i = 1, 2, 3)$	The hash functions

2.1 Computationally Difficult Problems

(i) *Elliptic Curve Discrete Logarithm Problem (ECDL)*: Let E be an elliptic curve on a finite field F. G is a cyclic subgroup of E with order q, and P is a generator of G. If the $P, Q \in G$ are known, it is hard to find $a \in Z_q^*$ satisfying $aP = Q$.

(ii) *Elliptic Curve Computational Diffie-Hellman Problem (ECCDH)*: Let the generator of prime order cyclic group G be P, and for $a_1, a_2 \in Z_q^*$, when P, a_1P, a_2P are known, it is hard to calculate $a_1a_2P \in G$.

2.2 Security Model

In 2007, LaMacchia et al. [11] proposed a new security model, which gave the adversary stronger attack capabilities based on the CK model [12], referred to as the eCK model. Let $\{P_1, P_2, \cdots, P_n\}$ denote the set of all participants, $sid = (P_i, P_j, m_1, m_2, \cdots, m_l)$ is the symbol of the session, where participant P_i is the initiator of the session, and participant P_j is the responder of the session, $m_1, m_2, \cdots, m_l$ representing the message sent between the participants of the session. Let $\prod_{i,j}^{sid}$ be the sid session between participant P_i and P_j. The adversary's attack capabilities such as eavesdropping, tampering, and replay in the eCK model are reflected through the following query methods:

(i) StaticKeyReveal(P_i) query: The adversary can obtain the long-term private key of the participant P_i through this query.

(ii) EphemeralKeyReveal(P_i, sid) query: The adversary can obtain the ephemeral key of participant P_i in session sid through this query.
(iii) SessionKeyReveal(sid) query: The adversary obtains the session key generated in session sid through the query.
(iv) Send(sid, m) query: Through this query, the adversary can send message m to the session identified as sid, and get the corresponding reply according to the provisions of the protocol.
(v) Test(sid) query: The adversary can only interrogate the fresh session sid, which simulates a random coin tossing algorithm and makes corresponding answers according to the coin toss results. If the coin toss results $b = 1$, the oracle returns the real session key; if the coin toss results $b = 0$, the oracle returns a random value with the same distribution as the session key.

Definition 1. *If both sessions $\prod_{i,j}^{sid}$ and $\prod_{i,j}^{sid^*}$ are run successfully, the generated session keys are equal, and the session identifiers sid and sid^* are the same, then the two sessions are called matching sessions.*

Definition 2. *Let the session run by users P_i and P_j be $\prod_{i,j}^{sid}$, and the session is called fresh if none of the following conditions hold:*

(i) *The adversary has interrogated session $\prod_{i,j}^{sid}$ or matching session $\prod_{j,i}^{sid}$ (if exists) for SessionKeyReveal(sid) query;*
(ii) *If the matching session $\prod_{j,i}^{sid}$ of session $\prod_{i,j}^{sid}$ exists, the adversary has performed EphemeralKeyReveal(P_i, sid) and StaticKeyReveal(P_i) queries at the same time, or performed both EphemeralKeyReveal(P_j, sid) and StaticKeyReveal(P_j) queries;*
(iii) *If there is no matching session for session $\prod_{i,j}^{sid}$, the adversary has performed EphemeralKeyReveal(P_i, sid) and StaticKeyReveal(P_i) queries at the same time, or performed a StaticKeyReveal(P_j) query.*

Definition 3. *Let k be a security parameter, if the probability of the adversary $\mathcal{A}$ winning the game is Pr, then the adversary's winning advantage can be defined as:*

$$Adv(k) = |Pr - \frac{1}{2}|. \tag{1}$$

If two honest participants complete the matching session and calculate the same key, and there is no adversary to win the security game with a non-negligible advantage, then the AKA protocol is secure under the eCK model.

3 Review and Security Analysis of ITGR Protocol

In this section, we review the ITGR protocol [10] and analyze the security properties of the protocol.

3.1 Review of ITGR Protocol

There are two types of participants in the ITGR protocol [10], namely IoT device U and server S. At the same time, the protocol consists of the following two phases: registration phase, and login and authentication phase.

Registration Phase. The registration steps are given as follows:

(i) Device U selects random number $x_u \in Z_q^*$ and its identity ID_u. Then U computes $R_u = x_u \cdot ID_u \cdot P$ and sends it to the server.
(ii) Server S receives the R_u and splits its private key x_s into two unequal parts x_s^1, x_s^2 such as $x_s^1 \neq x_s^2$, then server calculates $Y_u^1 = R_u x_s^1$, $Y_u^2 = R_u x_s^2$. The server sends $\{Y_u^1, Y_u^2\}$ to U and stores (R_u, Y_u^1, Y_u^2) in its database.
(iii) The device receives $\{Y_u^1, Y_u^2\}$ from server and stores (R_u, Y_u^1, Y_u^2) as its long-term private key securely.

Login and Authentication Phase. The steps of the login and authentication process between device U and server S are given as follows:

(i) Device U chooses its ephemeral key $t_u \in Z_q^*$ and computes $X_u = (Y_u^1 + Y_u^2) \cdot t_u \cdot H(R_u)$, $T_u = (Y_u^1 + Y_u^2) \cdot t_u$, then U sends $\{X_u, T_u\}$ to the server.
(ii) Server S receives $\{X_u, T_u\}$ and checks whether $T_u \stackrel{?}{=} X_u[H(R_u)]^{-1}$ to authenticate U. If true, the device U passes the identity authentication of server; else, login request from U is rejected.
(iii) The server S selects $t_s \in Z_q^*$ and computes $X_s = X_u t_s$, $T_s = (Y_u^1 + Y_u^2) \cdot t_s$, then S calculates its session key with U as $SK_{su} = T_u t_s$ and sends $\{X_s, T_s\}$ to the device U.
(iv) The device U receives $\{X_s, T_s\}$ from server S and checks whether $T_s \stackrel{?}{=} X_s[H(R_u)]^{-1} t_u^{-1}$ to authenticate S. If true, the server S passes the identity authentication of device; else, login response from S is rejected.
(v) The device U calculates its session key with S as $SK_{us} = T_s t_u$, then encrypts $E_{SK_{us}}[H(SK_{us})]$ and sends it to the server S.
(vi) The server S receives $E_{SK_{us}}[H(SK_{us})]$ and checks whether $H(SK_{su}) \stackrel{?}{=} D_{SK_{su}}[E_{SK_{us}}[H(SK_{us})]]$ to authenticate U and approve login request. If true, the session key exchange between U and S is over; else, the session key is invalid.

3.2 Security Analysis of ITGR Protocol

In this subsection, we prove that the ITGR protocol is susceptible to ephemeral key leakage attack and key compromise impersonation attack, and give specific attack methods.

Ephemeral Key Leakage Attack. If the ephemeral private key t_s of the server or the ephemeral private key t_u of the device is obtained by the adversary, then the adversary can calculate the session key by using the monitored transmission information. The specific steps are described as below:

If the adversary obtains the ephemeral private key t_s of the server S, and obtains T_u by monitoring the communication between the device and the server in public channel, then the adversary calculates $SK_{su} = T_u t_s$, and verify the session key by judging $H(SK_{su}) \stackrel{?}{=} D_{SK_{su}}[E_{SK_{us}}[H(SK_{us})]]$.

If the adversary obtains the ephemeral private key t_u of the device U, and obtains T_s by monitoring the communication between the device and the server in public channel, then the adversary calculate $SK_{us} = T_s t_u$, and verify the session key by judgment $H(SK_{us}) \stackrel{?}{=} D_{SK_{us}}[E_{SK_{us}}[H(SK_{us})]]$.

Key Compromise Impersonation Attack. The registration information of each device is stored in the server, and the server is at risk of being attacked by an adversary, so (R_u, Y_u^1, Y_u^2) may be obtained by an adversary. If the adversary obtains the secret (R_u, Y_u^1, Y_u^2) of the device U, then the adversary can deceive the device by pretending to be a legitimate server in front of the device, and generate the session key with device through the AKA process. The specific attack steps are described as below:

(i) The adversary $\mathcal{A}$ receives $\{X_u, T_u\}$ from device U and checks whether $T_u \stackrel{?}{=} X_u[H(R_u)]^{-1}$ to authenticate U. If true, the device U passes the identity authentication of adversary $\mathcal{A}$; else, login request from U is rejected.

(ii) The adversary $\mathcal{A}$ selects $t'_s \in Z_q^*$ and computes $X'_s = X_u t'_s$, $T'_s = (Y_u^1 + Y_u^2) \cdot t'_s$, then $\mathcal{A}$ calculates its session key with U as $SK'_{su} = T_u t'_s$ and sends $\{X'_s, T'_s\}$ to the device U.

(iii) The device U receives $\{X'_s, T'_s\}$ from adversary $\mathcal{A}$ and checks whether $T'_s \stackrel{?}{=} X'_s \cdot [H(R_u)]^{-1} t_u^{-1}$. If true, the adversary $\mathcal{A}$ passes the identity authentication of device U; else, login response from $\mathcal{A}$ is rejected.

(iv) The device U calculates its session key as $SK'_{us} = T'_s t_u$, then encrypts $E_{SK'_{us}}[H(SK'_{us})]$ and sends it to the adversary $\mathcal{A}$.

(v) The adversary $\mathcal{A}$ receives $E_{SK'_{us}}[H(SK'_{us})]$ from U and checks whether $H(SK'_{su}) \stackrel{?}{=} D_{SK'_{su}}[E_{SK'_{us}}[H(SK'_{us})]]$ to authenticate U and approve login request. If true, the session key negotiation between U and $\mathcal{A}$ is over; else, the session key is invalid.

Since the session key calculated by adversary $\mathcal{A}$ and device U is equal, the session key negotiation is completed.

4 Proposed Scheme

The ITGR protocol cannot resist key compromise impersonation attack and ephemeral key leakage attack, so this paper proposes a new AKA protocol called

eITGR protocol which makes up for the defects of ITGR protocol. The proposed protocol consists of three phases: system establishment phase, registration phase, and login and authentication phase.

4.1 System Establishment Phase

The server S establishes an elliptic curve $E(a,b) : y^2 = x^3 + ax + b$ on the finite field F, where G is a cyclic subgroup of E with order q, and P is a generator of G. Then the S selects three hash functions $H_1 : \{0,1\}^* \times G \rightarrow Z_q^*$, $H_2 : \{0,1\}^* \times G \times G \times \{0,1\}^* \rightarrow G$, $H_3 : \{0,1\}^* \times Z_q^* \rightarrow Z_q^*$.

4.2 Registration Phase

The registration phase is when the IoT device registers on the server in the secure communication environment, and generates a private key for communication between the two parties. The registration steps between device U and server S are as follows:

(i) Device U selects random number $x_u \in Z_q^*$ and its identity ID_u. Then U computes $X_u = x_u P, R_u = H_1(ID_u \| X_u) \cdot P$ and sends $\{ID_u, R_u\}$ to the server.
(ii) Server S receives the $\{ID_u, R_u\}$ from U and splits its private key x_s into two unequal parts x_s^1, x_s^2 such as $x_s^1 \neq x_s^2$, then server S calculates $y_u = x_s^1 + x_s^2 \cdot H_1(ID_u \| R_u)$, $Y_u = y_u P$. The server S sends Y_u to U and stores (ID_u, R_u, y_u) in its database.
(iii) The device receives Y_u from server and stores (x_u, R_u, Y_u) as its long-term private key securely.

4.3 Login and Authentication Phase

The login and authentication process between device U and server S is as follows:

(i) The device U chooses its ephemeral key $t_u \in Z_q^*$ and timestamp T, computes $T_u = t_u P$, $H_u = H_2(ID_u \| R_u \| T_u \| T)$, then U sends $\{ID_u, H_u, T_u, T\}$ to the server S.
(ii) After receiving the message $\{ID_u, H_u, T_u, T\}$, the server S first verifies the freshness of the timestamp T, then uses ID_u to find the corresponding entry (ID_u, R_u, y_u) in its database, After this server S uses the stored data to calculate $H_s = H_2(ID_u \| R_u \| T_u \| T)$, $X_u = H_u - H_s$ and checks whether $R_u \stackrel{?}{=} H_1(ID_u \| X_u) \cdot P$ to authenticate U. If true, the device U passes the identity authentication of server; else, login request from U is rejected.
(iii) The server S selects $t_s \in Z_q^*$ and computes $T_s = t_s P$, then S calculates $K_s = (t_s + y_u + H_1(ID_u \| X_u))(T_u + X_u)$ and its session key with U as $SK_{su} = H_1(ID_u \| K_s)$, $Q_s = H_3(ID_u \| SK_{su})$, and sends $\{T_s, Q_s\}$ to the device U.

(iv) The device U receives $\{T_s, Q_s\}$ from S, calculates $K_u = (t_u + x_u)(T_s + Y_u + R_u)$ and its session key with S as $SK_{us} = H_1(ID_u \| K_u)$, then checks whether $Q_s \stackrel{?}{=} H_3(ID_u \| SK_{us})$ to authenticate S. If true, the server S passes the identity authentication of device and the session key exchange between U and S is over; else, the session key is invalid.

The login and authentication phase is given in Fig. 1.

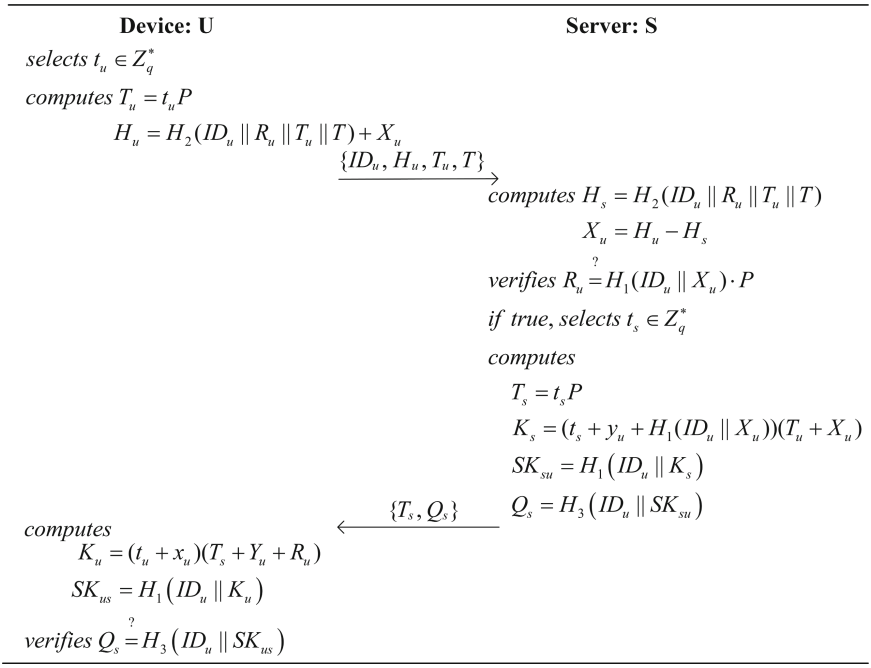

Fig. 1. Login and authentication phase of eITGR.

5 Security Analysis of eITGR Protocol

In this section, we present two proof methods for the eITGR protocol. First, we prove that the eITGR protocol is secure under the eCK model, then we use the Scyther tool to verify the security of the protocol.

5.1 Security Proof Under the eCK Model

We will prove that the protocol is secure under the eCK model [11] below.

Theorem 1. *The eITGR protocol satisfies eCK security if the ECCDH assumption holds on group G and H_1, H_2, H_3 are modeled by independent random oracles.*

Proof. Let the adversary $\mathcal{A}$ activates at most $n(k)$ honest parties and $s(k)$ sessions. The session key generated by the protocol is $SK = H_1(ID_u \| K)$ and H_1 is simulated as a random oracle, so the adversary has only three ways to distinguish the session key from the random value:

(i) Guessing attack: The adversary guesses the session key correctly.
(ii) Session key replication attack: The adversary establishes a session that does not match the test session but has the same session key. The adversary obtains the session key of the test session by querying the session key of the established session.
(iii) Forgery attack: The adversary $\mathcal{A}$ queries random oracle H_1 with the same secret value as that used to generate the test session key, and obtains the session key.

Since H_1 is a random oracle that obeys a uniform distribution, the probability $O(1/2^k)$ of the adversary guessing the correct key can be ignored. The two unmatched sessions cannot have the same session participant and ephemeral key, So the success probability of the session key replication attack is equivalent to find a collision of H_1, and the probability that two random oracles collide is $O(s(k)^2/2^k)$. Therefore, guessing attacks and session key replication attacks can be ignored, and only forgery attacks are considered below. Next, we show that if the adversary $\mathcal{A}$ wins the game by a non-negligible advantage, then the algorithm $\mathcal{S}$ can be constructed to solve the ECCDH problem with non-negligible probability. The algorithm $\mathcal{S}$ selects $A, B \in G$ where $A = aP$, $B = bP$, $a, b \in Z_q^*$, and sends the parameters (G, q, P, H_1, H_2, H_3) to $\mathcal{A}$. Then, algorithm $\mathcal{S}$ guesses that the adversary selects the session marked with *sid* as the test session, S_a and U_b are the owners of the test session and its matching session respectively with at least $1/n(k)^2 s(k)$ probability. Consider the following two cases according to the definition of session freshness:

(1) **Case 1:** The matching session of the test session exists, and the owner of the matching session is an honest participant;
(2) **Case 2:**No honest party owns a matching session for the test session.

Case 1 can be divided into the following four sub-cases:

(1) **Case 1.1:** The adversary performs the StaticKeyReveal(C) query on the test session and its matching session. Algorithm $\mathcal{S}$ sets A and B as the ephemeral public keys for participants S_a and U_b, and algorithm $\mathcal{S}$ assigns the long-time public-private key pair normally. Since algorithm $\mathcal{S}$ knows the long-term private keys of all participants, when queried by an adversary, algorithm $\mathcal{S}$ can answer truthfully according to the query ability in the model. When the adversary queries information about participants S_a and U_b, the algorithm $\mathcal{S}$ replies to the adversary as follows:

(i) StaticKeyReveal(C) query: The algorithm $\mathcal{S}$ provides the adversary with the long-term private key of participant C generated in the above simulation.
(ii) EphemeralKeyReveal(C, sid) query: If C is participant S_a or U_b, then algorithm $\mathcal{S}$ abandons the simulation, otherwise algorithm $\mathcal{S}$ returns the ephemeral key for party C in session sid.
(iii) Test(sid) query: If sid is not the test session, the simulation fails, otherwise, algorithm $\mathcal{S}$ returns a random value with the same distribution as the session key.
(iv) SessionKeyReveal(sid) query: If sid is a test session or its matching session, the simulation fails, otherwise set the session as $sid = (ID_s, ID_u, H_u, T_u, T_s, Q_s, T)$, then calculate the session key as $SK = H_1(ID_u \| K)$, where the input of K is as follows

$$K = ECCDH(T_u + X_u, T_s + Y_u + R_u). \tag{2}$$

Algorithm $\mathcal{S}$ queries whether the random oracle H_1 has been queried. If it has been queried, it outputs the correct session key; otherwise, it outputs a random value with the same distribution as the session key. Therefore, the algorithm $\mathcal{S}$ successfully simulates the environment in which the adversary $\mathcal{A}$ runs the protocol. If the adversary wins the forgery attack, then it must obtain the session key by querying, and the algorithm $\mathcal{S}$ must be able to solve the problem of ECCDH(A, B). In addition to, the only consideration is that the adversary solves the ECDL problem in time t while setting its advantage to be $Adv_G^{ECDL}(k, t)$. Let p_1 be the probability that the Case 1.1 occurs and the adversary succeeds through the forgery attack, so the advantages of simulating algorithm $\mathcal{S}$ to solve the ECCDH problem are as follows

$$Adv_G^{ECCDH}(k, t) \geq \frac{p_1}{n(k)^2 s(k)} - Adv_G^{ECDL}(k, t). \tag{3}$$

(2) **Case 1.2:** The adversary performs the StaticKeyReveal(C) query on the test session and EphemeralKeyReveal(C, sid) query on its matching session. Algorithm $\mathcal{S}$ sets A as the ephemeral public key of participant S_a, replaces X_u with B for participant U_b, and assigns a public-private key pair normally for the remaining participants. Then the algorithm $\mathcal{S}$ simulates the operating environment of the protocol, which is the same as in Case 1.1 except for the following query:
(i) StaticKeyReveal(C) query: If C is participant U_b, then algorithm $\mathcal{S}$ abandons the simulation; otherwise, algorithm $\mathcal{S}$ provides the adversary with the long-term private key of party C generated in the above simulation.
(ii) EphemeralKeyReveal(C, sid) query: If C is participant S_a, then algorithm $\mathcal{S}$ abandons the simulation; otherwise, algorithm $\mathcal{S}$ provides the adversary with the ephemeral key of participant C generated in the above simulation.

Let p_2 be the probability that the case 1.2 occurs and the adversary succeeds through the forgery attack, so the advantages of simulating algorithm $\mathcal{S}$ to solve the ECCDH problem are as follows

$$Adv_G^{ECCDH}(k,t) \geq \frac{p_2}{n(k)^2 s(k)} - Adv_G^{ECDL}(k,t). \tag{4}$$

(3) **Case 1.3:** The adversary performs the EphemeralKeyReveal(C, sid) query on the test session and StaticKeyReveal(C) query on its matching session. Algorithm $\mathcal{S}$ sets B as the ephemeral public key of participant U_b, replaces Y_u with A for participant S_a. Swap S_a and U_b, the rest is the same as case 1.2. Let p_3 be the probability that the case 1.3 occurs and the adversary succeeds through the forgery attack, so the advantages of simulating algorithm $\mathcal{S}$ to solve the ECCDH problem are as follows

$$Adv_G^{ECCDH}(k,t) \geq \frac{p_3}{n(k)^2 s(k)} - Adv_G^{ECDL}(k,t). \tag{5}$$

(4) **Case 1.4:** The adversary performs the EphemeralKeyReveal(C, sid) query on the test session and its matching session. Algorithm $\mathcal{S}$ replaces X_u with A and Y_u with B for participants S_a and U_b, computes $R_u = H_1(ID_u \| X_u) \cdot P$ and for the remaining $n(k) - 2$ participants, algorithm $\mathcal{S}$ assigns the public-private key pair normally. Then the algorithm $\mathcal{S}$ simulates the operating environment of the protocol, which is the same as in case 1.1 except for the following query:

(i) StaticKeyReveal(C) query: If C is participant S_a or U_b, then algorithm $\mathcal{S}$ abandons the simulation; otherwise, algorithm $\mathcal{S}$ returns the long-term private key for party C in session sid.

(ii) EphemeralKeyReveal(C, sid) query: The algorithm $\mathcal{S}$ provides the adversary with the ephemeral key of participant C generated in the above simulation.

Let p_4 be the probability that the case 1.4 occurs and the adversary succeeds through the forgery attack, so the advantages of simulating algorithm $\mathcal{S}$ to solve the ECCDH problem are as follows

$$Adv_G^{ECCDH}(k,t) \geq \frac{p_4}{n(k)^2 s(k)} - Adv_G^{ECDL}(k,t). \tag{6}$$

Case 2 can be divided into the following two sub-cases:

(1) **Case 2.1:** The adversary performs the EphemeralKeyReveal(C, sid) query on the test session. Since no honest participant participates in the matching session of the test session, it is equivalent that the adversary also obtains the ephemeral key of the intended communicating party, so it is similar with case 1.4.

(2) **Case 2.2:** The adversary performs the StaticKeyReveal(C) query on the test. Since no honest participant participates in the matching session of the test session, it is equivalent to the adversary also obtaining the ephemeral key of the intended communicating party, so it is similar with case 1.2.

In summary, if the adversary successfully distinguishes the session key from the random value with a non-negligible advantage, there is an algorithm $\mathcal{S}$ that solves the ECCDH problem with a non-negligible probability, which contradicts the ECCDH assumption. Therefore, the eITGR protocol given in this paper satisfies the security properties under the eCK model. □

5.2 Security Analysis Using Scyther Tool

Cremers and his team [13] developed the Scyther tool in 2006 for formal analysis of protocols. This paper uses Scyther tool to formally analyze the eITGR protocol, and the setting used is presented in Fig. 2 to achieve highly strong security, including perfect forward security and resistance to ephemeral key leakage attack. According to the analysis result in Fig. 3, we point out that the eITGR protocol is secure under the eCK model, which is consistent with the security proof result in Sect. 5.1.

6 Comparison with Other Protocols

In this section, we compare the security properties, computation cost, and communication overhead of the eITGR protocol with several similar protocols. The protocols involved in the comparison include Rostampour et al.'s protocol [9], ITGR [10], Hassan et al.'s protocol [14], Zhang et al.'s protocol [15] and Zhou et al.'s protocol [16].

6.1 Comparison of Security Properties

Table 2. Comparison of security properties.

Scheme	SP1	SP2	SP3	SP4	SP5
Rostampour et al. [9]	Y	Y	N	N	N
ITGR [10]	Y	Y	N	Y	N
Hassan et al. [14]	Y	Y	Y	N	N
Zhang et al. [15]	Y	N	N	Y	Y
Zhou et al. [16]	Y	Y	Y	N	N
eITGR	Y	Y	Y	Y	Y

The security properties of several recently proposed protocols are shown in Table 2, where SP1 refers to known key security, SP2 refers to forward security, SP3 represents resistance to key compromise impersonation attack, SP4 represents resistance to ephemeral key compromise impersonation attack, and SP5 represents resistance to ephemeral key leakage attack, respectively. Simultaneously, Y refers to the scheme achieves this security property and N refers to

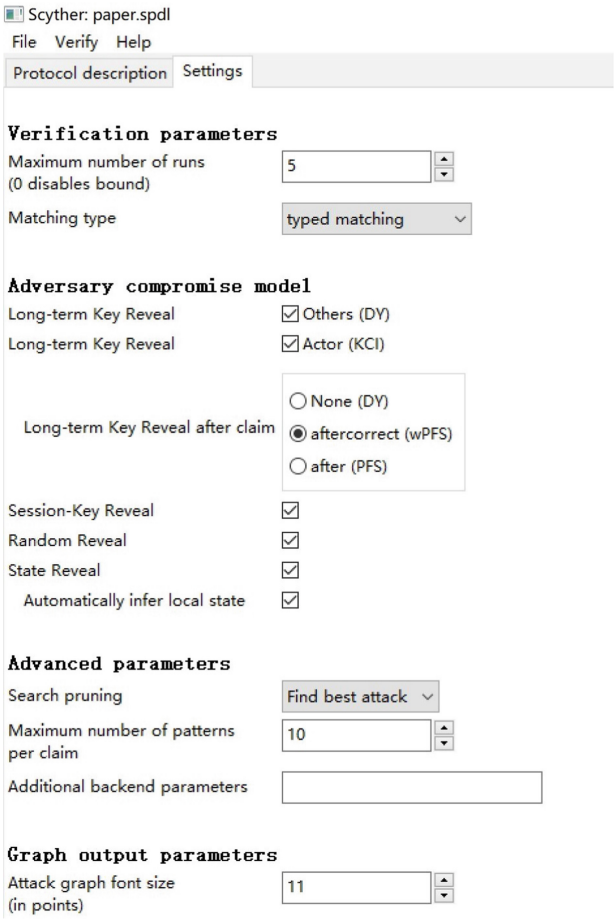

Fig. 2. The setting of Scyther.

the scheme does not achieve this security property. According to Table 2, the above-mentioned comparison protocols all have some security flaws. However, the protocol presented in this paper satisfies the above five security properties and is secure under the eCK model.

6.2 Computation and Communication Overhead Comparison

Computation efficiency and transmission consumption are important indicators to measure the communication performance of the protocol. In this section, we use T_b to denote the computation time of the bilinear map, T_m to represent the time of performing an scalar multiplication operation, T_a to represent the time of performing a point addition in elliptic curve group, T_i to represent the time of performing a multiplicative inverse operation and T_h to represent the time of

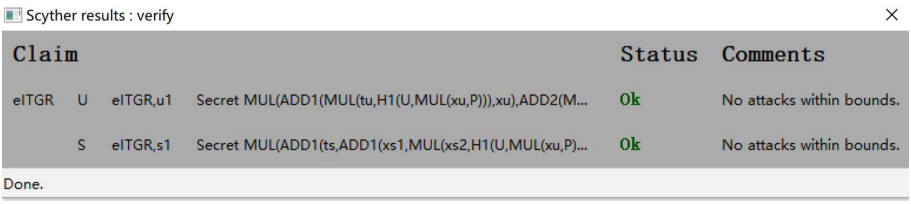

Fig. 3. Scyther verification results of the eITGR.

performing a hash function. Due to the relatively short operation time of XOR, integer addition and multiplication, they are ignored in this paper. The device computing cost and the total computing cost of each protocol are shown in the Table 3. It can be seen from Table 3 that our proposed protocol requires less computation on the device and greatly improves the computing efficiency on the device.

Table 3. Comparison of computation and communication cost.

Scheme	Device	Computation cost Total	Communication cost
Rostampour et al. [9]	$7T_m + T_a$	$13T_m + 2T_a$	$6\|G\|$
ITGR [10]	$4T_m + T_a + T_i + 2T_h$	$8T_m + 2T_a + 2T_i + 4T_h$	$4\|G\| + \|E_{SK}[H(SK)]\|$
Hassan et al. [14]	$4T_m + 4T_h$	$2T_b + 8T_m + 2T_a + 8T_h$	$2\|G\| + 2\|Z_q^*\| + \|ID\|$
Zhang et al. [15]	$4T_m + 3T_a + 3T_h$	$10T_m + 6T_a + 6T_h$	$6\|G\| + 3\|Z_q^*\|$
Zhou et al. [16]	$T_b + 3T_m + 3T_h$	$2T_b + 8T_m + T_a + 8T_h$	$2\|G\| + 4\|Z_q^*\| + 2\|T\|$
eITGR	$2T_m + 3T_a + 3T_h$	$5T_m + 5T_a + 7T_h$	$3\|G\| + \|Z_q^*\| + \|ID\| + \|T\|$

In order to avoid the difference of computing efficiency in different environments, this paper uses the Miracl function library in the C language environment [17]. According to the operation time in Table 4, the device running time and the total running time of the above protocols in the login and authentication phase are compared as shown in Fig. 4, where the abscissa is the protocol type, and the ordinate represents the time in milliseconds. It can be seen from Fig. 4 that the computing time on the device and total computing time of the eITGR protocol we propose is short, which greatly improves the practicability of the protocol in the IoT environment.

Table 4. Running time of various operations.

Operation	T_b	T_m	T_a	T_i	T_h
Computation time(ms)	7.8351	2.7580	0.0168	0.0147	0.0126

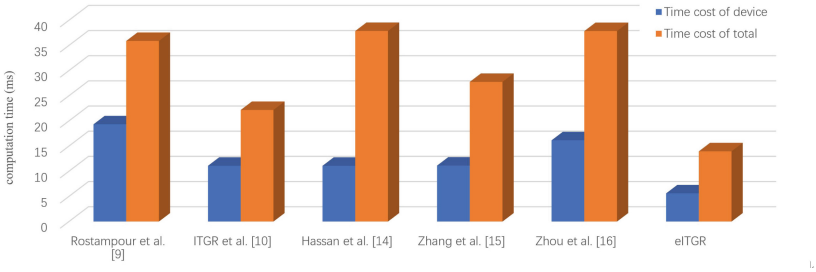

Fig. 4. Comparison of computation time.

The more data transmitted during the protocol operation, the more communication overhead. This paper compares the communication efficiency by comparing the amount of data that needs to be transmitted during the operation of different protocols. Table 3 shows the communication transmission data during the operation of each protocol. In order to compare the data transmission efficiency intuitively during the operation of the protocol, we set the parameter lengths as $|ID| = 10$ bytes, $|Z_q^*| = 20$ bytes, $|G| = 40$ bytes, timestamp $|T| = 8$ bytes and $|E_{SK}[H(SK)]| = 25$ bytes. The comparison of the amount of data transmitted by each protocol is shown in Fig. 5, where the abscissa represents the type of protocol, and the ordinate represents the amount of transmitted data in bytes.

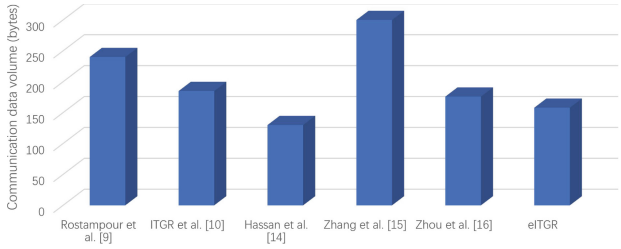

Fig. 5. Comparison of communication time.

In summary, in the case of more perfect security performance, the computation and communication overhead of the eITGR protocol proposed in this paper is relatively low, therefore, the eITGR protocol achieves a balance between security and communication efficiency.

7 Conclusion

With the wide application of IoT technology in life, the security of data transmission and identity authentication between devices and servers in the IoT has

attracted much attention. This paper analyzes that the ITGR protocol is vulnerable to ephemeral key leakage attack and key compromise impersonation attack, so an AKA protocol suitable for the IoT environment is proposed. Our protocol is proven to be secure under the eCK model based on the ECCDH assumption, and then the security of the protocol is verified using Scyther tool. Finally, we compare the proposed protocol with similar protocols in terms of security properties and communication cost in the same environment, the analysis results show that our protocol is secure and efficient. In future work, we will consider how to design a secure and efficient group authenticated key agreement protocol for IoT and cloud server.

Acknowledgments. This work was supported in part by the National Natural Science Foundation of China (nos. 61872449 and 62172433).

References

1. Hassija, V., Chamola, V., Saxena, V., et al.: A survey on IoT security: application areas, security threats, and solution architectures. IEEE Access **7**, 82721–82743 (2019)
2. Al-Turjman, F., Zahmatkesh, H., Shahroze, R.: An overview of security and privacy in smart cities' IoT communications. Trans. Emerg. Telecommun. Technol. **33**(3), 1–19 (2022)
3. Lova Raju, K., Vijayaraghavan, V.: A self-powered, real-time, NRF24L01 IoT-based cloud-enabled service for smart agriculture decision-making system. Wirel. Pers. Commun. **124**(1), 207–236 (2022)
4. Saini, K., Kalra, S., Sood, S.K.: An integrated framework for smart earthquake prediction: IoT, fog, and cloud computing. J. Grid Comput. **20**(2), 1–20 (2022)
5. Kumar, P., Kumar Bhatt, A.: Enhancing multi-tenancy security in the cloud computing using hybrid ECC-based data encryption approach. IET Commun. **14**(18), 3212–3222 (2020)
6. Khelf, R., Ghoualmi-Zine, N., Ahmim, M.: TAKE-IoT: tiny authenticated key exchange protocol for the internet of things. Int. J. Embed. Real-Time Commun. Syst. **11**(3), 1–21 (2020)
7. Qi, M., Chen, J.: Secure authenticated key exchange for WSNs in IoT applications. J. Supercomput. **77**(12), 13897–13910 (2021)
8. Peng, A.L., Tseng, Y.M., Huang, S.S.: An efficient leakage-resilient authenticated key exchange protocol suitable for IoT devices. IEEE Syst. J. **15**(4), 5343–5354 (2020)
9. Rostampour, S., Safkhani, M., Bendavid, Y., et al.: ECCbAP: a secure ECC - based authentication protocol for IoT edge devices. Pervasive Mob. Comput. **67**, 1–36 (2020)
10. Iqbal, U., Tandon, A., Gupta, S., et al.: A novel secure authentication protocol for IoT and cloud servers. Wirel. Commun. Mob. Comput. **2022**(5), 1–17 (2022)
11. LaMacchia, B., Lauter, K., Mityagin, A.: Stronger security of authenticated key exchange. In: Susilo, W., Liu, J.K., Mu, Y. (eds.) ProvSec 2007. LNCS, vol. 4784, pp. 1–16. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-75670-5_1

12. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 453–474. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44987-6_28
13. Cremers, C.J.F.: Scyther: semantics and verification of security protocols. Eindhoven University of Technology, Eindhoven Netherlands (2006)
14. Hassan, A., Eltayieb, N., Elhabob, R., et al.: An efficient certificateless user authentication and key exchange protocol for client-server environment. J. Ambient. Intell. Humaniz. Comput. **9**(6), 1713–1727 (2018)
15. Zhang, J., Huang, X., Wang, W., et al.: Unbalancing pairing-free identity-based authenticated key exchange protocols for disaster scenarios. IEEE Internet Things J. **6**(1), 878–890 (2018)
16. Zhou, Y., Liu, T., Tang, F., et al.: An unlinkable authentication scheme for distributed IoT application. IEEE Access **7**, 14757–14766 (2019)
17. Li, Y., Cheng, Q., Liu, X., et al.: A secure anonymous identity-based scheme in new authentication architecture for mobile edge computing. IEEE Syst. J. **15**(1), 935–946 (2020)