2017 IEEE Trustcom/BigDataSE/ICESS

A Lightweight Anonymous Mutual Authentication with Key Agreement Protocol on ECC

Wei Zhang*†§, Dongdai Lin*†‡, Hailong Zhang*†§, Cong Chen*†§, Xiaojun Zhou*†§
*State Key Laboratory of Information Security,
Institute of Information Engineering, Chinese Academy of Sciences
†School of Cyber Security, University of Chinese Academy of Sciences
‡Corresponding Author: ddlin@iie.ac.cn
§{zhangwei, zhanghailong, chencong, zhouxiaojun}@iie.ac.cn

***Abstract*—Recently, Goutham et al. proposed an ID-based user authentication with key agreement on elliptic curve cryptography(ECC), which is suitable to be applied in client-server-environment. The scheme mainly negotiates a temporary session key between two parties. However, we find that the scheme contains some security flaws, such as incomplete anonymity, no provision for updating private key and so on. In light of this, we propose a new version of anonymous authentication with key agreement protocol used for client-server environment, especially, the calculation of both sides are lower than the previous scheme. The proposed scheme provides more security features like complete anonymity, dynamic private key updating while keeping the merits of Goutham et al.'s scheme. We also optimize the performance of the scheme to get a lightweight protocol which is more suitable for resource-constrained device applied in Internet of Things(IoT) or wireless sensor network(WSN) applications.**

***Keywords*—anonymous; mutual authentication; key agreement; ECC**

I. Introduction

Internet of Things(IoT) applications as the extension of internet, cover more and more aspects of our daily life. For instance, Smart Home can remote control your home monitor and manage your household applications. Body area netwoks(BAN) can be used to monitor human health in the manner of attaching human body by various medical sensors. The major component of IoT applications are sensors, embedded microprocessor, IoT gateway for transporting heterogeneous network data and a server for storing, analyzing as well as making decision. The secure problems of IoT applications such as weak password, plaintext transmission of sensitive data etc., have become increasingly serious. Take for a recent example, Friday's DDoS(Distributed Denial-of Service) attacks in the United States and Europe in October last year, that created major website outages across the Internet may be deeply impressed for people. The series of attacks recruited millions of IP address, many associated with IoT devices, flooded Domain Name System(DNS) server provider Dyn, shutting down the websites of Twitter, Spotify, Netflix, Github, Amazon and other Dyn clients. People have become conscious of the security vulnerabilities of IoT device. Due to their low cost, most IoT devices are generally resource-constrained in terms of processing capacity and storage, which consume much time to run complicated cryptographic primitives and protocols. A state-of-the-art IoT device features an 8 or 16-bit low power microprocessor whose working frequency is at less than 10MHz and a few KiloByte(KB) of RAM and flash memory is equipped. So using efficient cryptographic primitive and designing light-weight protocol is more important for IoT device.

Authentication protocol for IoT device aims at protecting identity of the devices. The server could identify legal or registered IoT device, and refuse illegal, unregistered or fake devices. Then the server and the IoT device establishes session-key for secure communication which includes transmitting sensitive data and command, and updating device information, certificate, software or firmware, etc. So, authentication plays a crucial role in satisfying the security requirements for IoT devices. In recent years, many authentication protocols have been proposed for embedded devices, which use hash chain, symmetric cryptography, or public-key cryptography. However, hash chain and symmetric cryptography based schemes need store more data about user's identity and the corresponding symmetric key when the number of devices increases, more serious, it is complicated to update user's key and establish a session key in this kind of protocols. The schemes with public-key cryptography could have a good deal with these aspects. But some of them need too much time to run because the public-key cryptographic algorithms used in the schemes can result in significant overheads in terms of time and energy consumption.

As an efficient public-key cryptographic primitive that could be applied to IoT device, Elliptic Curve Cryptography(ECC) has smaller key sizes and storage and few computations compared with other public-key primitives like Rivest Shamir Adleman(RSA), pairing and Learning With Errors(LWE) under the same secure level. 160-bit ECC key provides the same level of security as a 1024-bit RSA key. The computation cost of pairing is approximately 20 times higher than that of the scalar multiplication at the same security level [1]. The advantages provided by ECC can be important in environments where processing power, storage, bandwidth or power consumption is constrained.

Regarding to the authentication issues in the IoT applications, some researchers also proposed related secure schemes in [2]–[8]. Huang et al. [2] used ECC to construct an authen-

2324-9013/17 $31.00 © 2017 IEEE
DOI 10.1109/Trustcom/BigDataSE/ICESS.2017.234

tication key exchange protocols for secure communication in a self-organizing sensor network in 2003. In 2005, Tian et al. [3] analyzed Huang et al.'s scheme and pointed out that the scheme is not secure if an attacker has compromised one single security manager, and then proposed an improvement authenticated key exchange version. In 2011, Debiao et al. [9] proposed an ID-based mutual authentication with key agreement protocol on ECC with provable security, which server could compute the user's private key by its identity, and gave the comparison with some similar schemes. In the next year, Islam et al. [10] stated Debiao et al.'s scheme suffers some known attacks, such as many-logged in user's problem, impersonation attack, and claimed their protocol is not secure. In 2015, Goutham et al. [11] proposed an anonymous ID-based remote mutual authentication with key agreement protocol using smart cards based Debiao et al.'s scheme. To avoiding user identity be revealed to adversary, the improved version add some new security features like anonymity and ID updating, but also bring in more communication overload on both sides and more data that server need to store.

In this paper, we show that schemes proposed by Debiao et al. and Goutham et al. schemes have some security flaws, and we propose a new protocol that is completely anonymous and computationally efficient. Therefore the scheme is not only more secure than previous ones, but can also further reduce the computation of resource-constrained devices in practice.

The rest of this paper is organized as follows: In section 2, we review the preliminaries of related knowledge. In section 3, we briefly introduce previous schemes and show their weaknesses. Our scheme is proposed in Section 4 and the security analysis is shown in Section 5. In Section 6, we analyze the performance of our scheme and we compare it with other schemes. Finally, conclusions are given in Section 7.

II. Elliptic Curve Cryptography(ECC)

In the middle of 1980s, Victor Miller [12] and Neal Koblitz [13] firstly used elliptic curve for cryptography independently. Elliptic curve cryptography computation built on finite fields, which can either choose a prime field or a binary field. Point addition and Point doubling are the basic arithmetic of elliptic curves and the basic operations of scalar point multiplication $Q = kP$, where $k \in \mathbb{Z}$, point $Q, P \in E(\mathbb{F}_q)$, $\mathbb{F}_q$ is a prime finite field. The hardness of elliptic curve discrete logarithm problem is essential for the security of all elliptic curve cryptographic schemes as follows.

Definition 1: Let E be an elliptic curve defined over a finite field $\mathbb{F}_q$. P and Q be points in $E(\mathbb{F}_q)$, and suppose that P has prime order n, assuming that $Q = dP$, where d is an integer from the interval [1, n-1]. The problem of detemining d given the domain parameters and Q is the elliptic curve discrete logarithm problem(ECDLP).

Definition 2: The elliptic curve Diffie-Hellman problem (ECDHP) is: given an elliptic curve E defined over a finite field $\mathbb{F}_q$, a point $P \in E(\mathbb{F}_q)$ of order n, and points $A = aP$, $B = bP \in < P >$, find the point $C = abP$.

Definition 3: The elliptic curve decision Diffie-Hellman problem(ECDDHP) is: given an elliptic curve E defined over a finite field $\mathbb{F}_q$, a point $P \in E(\mathbb{F}_q)$ of order n, and points $A = aP$, $B = bP$, and $C = cP \in < P >$, determine whether $C = abP$ or, equivalently, whether $c \equiv ab \pmod n$.

III. Review of previous authentication protocols

In this section, we briefly review the Debiao et al.'s ID-based remote user authentication scheme, and Goutham et al.'s scheme which is a modified version based on previous one. These protocols consist of 3 basic steps. 1) system initialization phase. 2) client registration phase. 3) mutual authentication with key agreement phase [9], [11]. The Goutham et al.'s scheme also has other two phases named max, min values and USN phase and identity updating phase.

A. Debiao et al.'s scheme

1) System Initialization Phase: In this phase, server S generates system parameters as follows: S chooses an elliptic curve equation E over a finite field $\mathbb{F}_p$, and a base point $P \in E(\mathbb{F}_p)$ of order n, then S selects its master key x and computes corresponding public key $P_s = xP$. In addition to accomplish initialization, S also needs some auxiliary calculation modules like three secure one-way hash function H_1, H_2, H_3 and a message authentication code $MAC_k(m)$. Then, S keeps master key x as a secret, and publishes system parameter tuple $(\mathbb{F}_p, E, n, P, P_s, H_1, H_2, H_3, MAC_k(m))$.

2) Client registration phase: When a client C_i wants to register to the server S, C_i submits its identity ID_{C_i} to S. S computes $h_{C_i} = H_1(ID_{C_i})$ by ID_{C_i} and H_1, client's private key $D_{C_i} = \frac{1}{x+h_{C_i}} P \in G$, then S sends D_{C_i} to C_i through a secure channel. The corresponding public key is $P_{C_i} = (h_{C_i} + x)P = h_{C_i}P + P_s$.

3) Mutual authentication with key agreement phase: In this phase, the client C_i requests server S through a message.

i Client C_i chooses a random number $r_{C_i} \in Z_n^*$, and computes $M = r_{C_i} \cdot P$, $M' = r_{C_i} \cdot D_{C_i}$, $k = H_2(ID_{C_i}, T_{C_i}, M, M')$, where T_{C_i} is the current timestamp of client. Then, C_i sends the request message $M_1 = \{ID_{C_i}, T_{C_i}, M, MAC_k(ID_{C_i}, T_{C_i}, M)\}$ to the server.

ii After receiving M_1, S checks the validity of ID_{C_i} and the time freshness of T_{C_i} firstly. The freshness is decided by $T' - T_{C_i} < \Delta T$, where T' is current time of server and ΔT is the valid time interval. S continues the execution only if time verification passes. S computes $h_{C_i} = H_1(ID_{C_i})$, $M' = \frac{1}{x+h_{C_i}} M$ and $k = H_2(ID_{C_i}, T_{C_i}, M, M')$, and then uses k to check the integrity of $MAC_k(ID_{C_i}, T_{C_i}, M)$. S will quit the current session if the check produces a negative result. Otherwise, S chooses a random number $r_s \in Z_n^*$ and computes $W = r_s \cdot P$, $K = r_s \cdot M$ and the session key $sk = H_3(ID_{C_i}, T_{C_i}, T_s, M, W, K_s)$. Then S sends $M_2 = \{ID_{C_i}, T_s, W, MAC_k(ID_{C_i}, T_s, W)\}$ to C_i, where T_s is server's current timestamp.

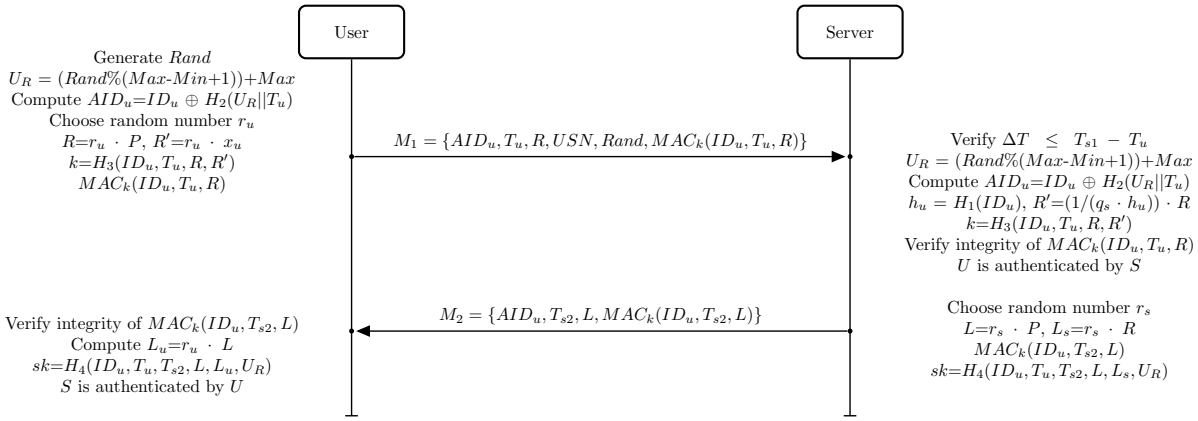

Fig. 1. Goutham et al. proposed mutual authentication with key agreement phase

iii Upon receiving M_2, C_i checks the integrity of $MAC_k(ID_{C_i}, T_s, W)$ using key k, and then C_i computes $K_{C_i} = r_{C_i} \cdot W$ and computes the session key $sk = H_3(ID_{C_i}, T_{C_i}, T_s, M, W, K_{C_i})$.

B. Goutham et al.'s scheme

The scheme of Goutham et al., shown in Fig.1, is an optimized version of the previous scheme, which adds some new secure features like anonymous, identity updating in the protocol. In general, the step of Goutham's is similar to the step of Debiao et al.'s, and we only point out the different processes.

1) Max, Min values and USN phase: The Max, Min value and USN phase is before user registration phase, and it helps user to hide its identity. These three values are assigned by server. The anonymous ID chosen by the user is between Min and Max. The USN value that helps server to find user's information in database is a unique sequence number of user. Max, Min and USN values are saved in server-side.

2) The Anonymous Process: In order to achieve user identity anonymity, USN is assigned to represent the unique user identity, and server could search USN to find user's Max and Min values. At last, the original identity is computed by anonymous identity, Min and Max values. The anonymity process runs at the beginning of mutual authentication with key agreement phase as follows.

i Client C_i computes its anonymous ID by performing $U_R = Rand\%(Max - Min + 1) + Max$, $AID_U = ID_U \oplus H_2(U_R \parallel T_U)$.

ii Client C_i sends M_1 to S, which is similar to the previous scheme, except using AID instead of ID and inserting 2 parameters USN and $Rand$. $M_1 = \{AID_{C_i}, T_{C_i}, R = r_u \cdot P, USN, Rand, MAC_k(ID_{C_i}, T_{C_i}, R)\}$.

iii Server S needs to remove anonymous firstly when it receives M_1 by identifying USN value, then computes C_i's real ID using the corresponding Min and Max. $U_R = Rand\%(Max - Min + 1) + Max$. Finally, S continues mutual authentication process as Debiao's.

3) Identity Updating Phase: Client C_i can launch identity updating phase when it wants to change its original identity ID_{C_i}. The phase above can prevent parts of malicious attack from adversary.

i Client C_i re-selects an identity $ID^{\#}_{C_i}$ and computes $AID_{C_i} \oplus ID^{\#}_{C_i}$ and sends S a request message $M = \{AID_{C_i}, AID^{\#}_{C_i}, USN, Rand, T_{C_i}, MAC_k(ID_{C_i}, ID^{\#}_{C_i})\}$.

ii Server S receives the re-selected ID_{C_i} as $ID^{\#}_{C_i} = ID_{C_i} \oplus AID^{\#}_{C_i}$, and then Server S updates $ID^{\#}_{C_i}$ and computes $h^{\#}_{C_i} = H_1(ID^{\#}_{C_i})$ and client's secret key $x^{\#}_{C_i} = \frac{1}{q_s \cdot h_{C_i}} \cdot P$, and then delivers a message $\{x^{\#}_{C_i}, Max, Min, USN\}$ to the client smart card through a secure channel or an off-line interaction.

C. Cryptanalysis of Above Protocols

Debiao et al.'s scheme has several security weaknesses [10].

1) Many Logged-in Users' Problem: If the private key D_{C_i} is leaked, the adversary can get the identity ID_{C_i} by eavesdropping on the massage M_1, then login in the server S by selecting an random value $r_{C_i} \in Z^*_n$ and executing the following steps of original mutual authentication scheme. The server S cannot detect this attack because S does not have any information of C_i.

2) Privileged-Insider Attack: This attack is mainly caused by Identity-Based Cryptosystem(IBC) suffering from the well known key escrow problem. The attacker may access the server S if he has the knowledge of C_i's private key D_{C_i}.

3) No Provision for Changing/Updating Private Key: C_i's private key $D_{C_i} = \frac{1}{x+H_1(ID_{C_i})} \cdot P$ is computed by the identity ID_{C_i} and the server S's secret key x. So the client has to change its ID every time if it wants to get a new private key. Therefore, Debiao's protocol has not provided strong scalability for changing/updating the private key.

4) Inability to Protect Users Anonymity: As Debiao et al.'s scheme is based on IBC, the user identity has a close relation with user's private key, which means that user anonymity

is an important aspects for security. However, the user identity ID_{C_i} is transmitted openly in the message $M_1 = \{ID_{C_i}, T_{C_i}, M, MAC_k(ID_{C_i}, T_{C_i}, M)\}$. The revelation of user identity may foster adversary to break the mutual authentication process.

Goutham et al.'s scheme is an improved version of Debiao's and it repairs some weaknesses. But some defects also exist in the new protocol.

5) Incomplete Anonymity: Goutham et al.'s scheme protects client's anonymity through computing $U_R = (Rand\%(Max - Min + 1)) + Max$, then sends USN and $Rand$ value in message M_1 during mutual authentication with key agreement phase. Because USN is the unique value, server S can find client's Max and Min values by searching USN and then compute client's ID. It means that the scheme adds another identity named USN, although it hides the client's ID. The adversary recognises the client's USN as this value is not often changed. Adding USN and $Rand$ in M_1 also increases the communication overhead, which may affect performance of protocol in wireless environment.

6) No Updating The Private Key: Client's private key is $D_{C_i} = \frac{1}{x \cdot H_1(ID_{C_i})} \cdot P$, which only relates to server's secret key x and client's identity ID. Therefore the client can only update its private key by updating its identity. And even more serious is that if the server's secret key x is revealed, all the clients' private key could be computed by the adversary and suffered impersonate attack.

7) No Forward/Backward Security in Identity Updating Phase: Client C_i updates its original identity through $AID^{\#}_{C_i} = ID_{C_i} \oplus ID^{\#}_{C_i}$, where $AID^{\#}_{C_i}$ is sent to server using plaintext in the updating request message. The adversary could compute the new original identity if he knows the former original identity value. The former original identity could also be computed in the same way. Meanwhile, the identity value is selected by client, which may cause identity collision with other clients. The protocol also does not mention how to deal with this situation.

IV. Proposed Scheme

The proposed scheme is a lightweight protocol between client C and server S, and it depends on previous schemes, which could provide mutual authentication with key agreement anonymous. The scheme supports client's complete anonymity which means no information about client's identity is used in the interaction of two parties, except an anonymous identity. We further reduce client's computation to save its time and power consumption in order to be adapted to resource-constrained embedded devices used in IoT applications. The proposed scheme is analyzed four phases: system initialization phase, user registration phase, mutual authentication with key agreement phase and identity/private key updating phase. The scheme is presented as follows.

A. System Initializing Phase

The system initializing phase is similar to Debiao et al. and Goutham et al.'s scheme. Server S generates system initialized parameters as follows:

i S selects a base point P with order n over a chosen elliptic curve E over a prime finite field $\mathbb{F}_q$.
ii S selects its secret key x randomly and then computes public key $P_s = xP$.
iii S selects four secure one-way hash functions H_1, H_2, H_3, H_4 and a message authentication code MAC_k.
iv S stores secret key x privately and publishes public parameters $\{E(\mathbb{F}_q), P, P_s, H_1, H_2, H_3, H_4, MAC_k(m)\}$.

B. Client Registration Phase

When a client C_i needs to register to server S, C_i sends its identity ID_{C_i} and a random value t_{C_i} to S. After receiving ID_{C_i} and t_{C_i}, S computes $h_{C_i} = H_1(ID_{C_i})$ and uses its secret key x to compute the client's private key $D_{C_i} = \frac{t_{C_i}}{x \cdot h_{C_i}} P \in \mathbb{G}$, and client's anonymous identity $AID_{C_i} = ID_{C_i} \oplus H_2(ID_{C_i} || t_{C_i})$. The server S needs to check whether the AID_{C_i} has a collision with other registered anonymous identities, if so, S needs to tell C_i to re-register again. Then D_{C_i} and AID_{C_i} are delivered to C_i through off-line secure approach or other secure channel, such as Secure Sockets Layer(SSL). S saves user's t_{C_i}, ID_{C_i} and AID_{C_i} in server database.

C. Mutual Authentication with Key Agreement Phase

In this phase, client C_i requests the server S by sending a message if C_i wants to access the S. S uses its secret key to verify timestamp and authenticity of C_i's message, and S generates a session key with C_i. The detail of this authentication phase, shown in Fig.2, is illustrated as follows.

i C_i chooses a random number $r_{C_i} \in \mathbb{Z}^*_n$, and computes $R = r_{C_i} \cdot P$, $R' = D_{C_i} + R$. Then C_i computes the key of message authentication code $k = H_3(ID_{C_i}, T_{C_i}, R, R')$, where T_{C_i} is a current timestamp of client. Finally, C_i sends a requested message $M_1 = \{AID_{C_i}, T_{C_i}, R, MAC_k(ID_{C_i}, T_{C_i}, R)\}$ to S.
ii S checks ID_{C_i}'s validity and time freshness of M_1 firstly by performing $T' - T_{C_i} \leq \Delta T$, where T' is server's current time received M_1. S will abort the current session if either validity or freshness fail. Then S computes $h_{C_i} = H_1(ID_{C_i})$, $R' = \frac{t_{C_i}}{x \cdot h_{C_i}} P + R$ and $k = H_3(ID_{C_i}, T_{C_i}, R, R')$. Next, S checks the integrity of $MAC_k(ID_{C_i}, T_{C_i}, R)$ using k, and S will abort the current session if the result is wrong.
iii S chooses a random value r_s, and computes $L = r_s \cdot P$, $L_s = r_s \cdot R$. The next anonymous client identity $AID^{\#}_{C_i} = AID_{C_i} \oplus H_2(ID_{C_i} || L_s)$ is generated afterwards. S will check whether the new generated anonymous identity has a collision with others. If so, a new random value r_s is generated and L, L_s and $AID^{\#}_{C_i}$ are computed again by S.
iv S generates $MAC_k(ID_{C_i}, T_s, L)$ and sends the mesage $M_2 = \{AID_{C_i}, T_s, L, MAC_k(ID_{C_i}, T_s, L)\}$ to C_i, where T_s is the timestamp when S sends message to C_i. Finally, S will compute the session key $sk = H_4(ID_{C_i}, T_{C_i}, T_s, R, L, L_s)$.

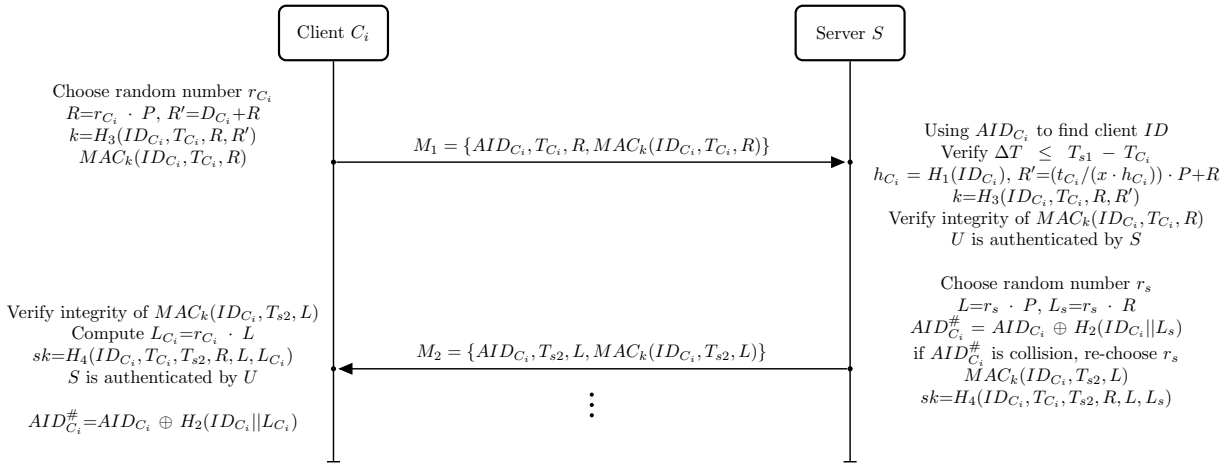

Fig. 2. Proposed mutual authentication with key agreement phase

v When C_i receives M_2, C_i checks the integrity of message authentication code MAC_k using k firstly. If the result is negative, C_i will abort the session. Otherwise, C_i computes $L_{C_i} = r_{C_i} \cdot L$, and then generates $sk = H_4(ID_{C_i}, T_{C_i}, T_s, R, L, L_{C_i})$ as its session key to access the server.

vi C_i calculates next anonymous identity $AID^{\#}_{C_i} = AID_{C_i} \oplus H_2(ID_{C_i}||L_{C_i})$ and saves $AID^{\#}_{C_i}$ for next mutual authentication after this session is closed.

D. Identity/private key Updating Phase

Because the client original identity never occurs as plaintext, the client does not need to change its identity frequently. The updating of client's private key is similar to the previous situation. However, to resist brute force search attack for private key, client should update its identity and private key periodically. The best way to update client identity and private key is that the new value is embedded in the negotiated session as the reason of the adversary cannot distinguish the command from the data format, especially in wireless environment, where adversary could capture easily.

V. Security Analysis

This section proves that the proposed protocol can achieve the mutual authentication between client and server. Various kinds of issues including client anonymity, session key agreement and so on are illustrated as follows and the conclusion is shown in Table I.

A. Client Anonymity

The proposed scheme uses anonymous identity instead of the client original identity, as the anonymous identity is generated as $AID^{\#}_{C_i} = AID_{C_i} \oplus H_2(ID_{C_i}||L_{s(c)})$, where the ID_{C_i}, L_s and L_c are unpredictable values for adversary. The adversary also cannot get any useful information about the original identity ID_{C_i} as it is computed by a one-way trapdoor hash function H_2.

B. Complete Anonymity

The complete anonymity or perfect anonymity means no identity-related information is revealed except the anonymous identity. The only one related data transports as plaintext during the session is the anonymous identity. The anonymous identity is calculated by original identity ID_{C_i}, random value r_s, common computed session value $L_{c(s)}$, and one-way hash function H_2, which guarantee the randomness of anonymity. Because server has all original and anonymous identity of registered client, it should check there is no collision between new generated value and others. So the dynamic mechanism ensures user anonymity completely.

C. Mutual Authentication

Both sides need to authenticate each other in a client-server model. In authentication with key agreement phase, the authentication of client C_i is validity of $MAC_k(ID_{C_i}, T_{C_i}, R, R')$, while server is $MAC_k = (ID_{C_i}, T_{C_i}, L)$, respectively. Adversary cannot compute a valid session if he does not know the real identity of client, and compute the common computed value R' and the correct $k = H_3(ID_{C_i}, T_{C_i}, R, R')$.

D. Resistance to Replay Attack

An adversary can capture the authentication message and then replay it to server, which is more easily in the wireless environment for IoT application. So it is necessary to check the message's availability. In the proposed scheme, the timestamp T_{C_i} and T_s are involved in the interactive session and further embedded in the $MAC_k(ID_{C_i}, T_{C_i}, L)$ and $MAC_k(ID_{C_i}, T_s, L)$. Server will check the time firstly if it receives the client request and server would abort the session if $T' - T_{C_i} \geq \Delta T$. Adversary also cannot change the MAC without the knowledge of k. As a result, adversary cannot launch a success login in the server through replay attack.

E. Resistance to Modification Attack

Client or server could detect the change of the message, if adversary modifies data in the message. This is because MAC code guarantees the message cannot be altered. The adversary also cannot generate a legitimate message without a known original identity and relevant private key.

F. Dynamic Private Key Updating

Client could request to update its private key in private key updating phase. The randomness of private key is provided by a random value t_{C_i} in $D_{C_i} = \frac{t_{C_i}}{x \cdot h_{C_i}} P$. The new characteristic avoids the drawback that the client identity decides an unchangeable private.

G. Forward/Backward security of identity updating

A process is called forward and backward security of identity updating, if getting an ID of client does not affect the security of the previous and the next anonymous ID. The point is that the generation of anonymous identity $AID^{\#}_{C_i} = AID_{C_i} \oplus H_2(ID_{C_i} || L_{c(s)})$ not only uses the original identity value, but also introduces a common computed value named L_s in server or L_c in client, which the adversary does not know.

H. Resistance to Impersonation Attack

To make the proposed protocol free from impersonation attack, we assume an adversary gets the knowledge of a client original and anonymous identity. It can generate the message AID_{C_i}, T_{C_i} and $R = r_{C_i} \cdot P$, except the message authentication code $MAC_k(ID_{C_i}, T_{C_i}, R)\}$ as it does not know the private key D_{C_i}. This situation that adversary wants to impersonate a server is similar to the the previous example as it does not have the secret key x and the client original identity.

I. Known Session-Key Attack

A scheme is called Known session key secure if an adversary, getting previous session key, cannot access the current session to server. In our scheme, the agreed session is based on the ECDLP and one-way hash function, and the session key is an ephemeral key, which means each session agreed key is independent of the other. So, even one session key is revealed, the other key will still safe.

J. Resistance to De-synchronous Attack

The generating of new AID requires synchronous computation for both side. If the client does not receive the message from the server, it may never connect to the server. So we have a rational assumption that the mutual authentication with key agreement is only the beginning of the communication, which means a secure interactive session follows the proposed scheme. The subsequent session, implying an acknowledge message, plays an important role for this protocol. If the server does not receive message after the mutual authentication, server will know client may loss the message and stop to update the new AID value.

TABLE I
COMPARISON OF SECURITY ANALYSIS

Security Property	Debiao [9]	Goutham [11]	Proposed Scheme
Mutual Authentication	√	√	√
Replay Attack	√	√	√
Modification Attack	√	√	√
Client Anonymity	–	√	√
Impersonation Attack	–	√	√
Known session-key Attack	–	√	√
Identity Updating	–	√	√
Complete Anonymity	–	–	√
Private Key Updating	–	–	√
Forward/Backward Security	–	–	√

TABLE II
COMPARISON OF SCHEME PERFORMANCE

Scheme	Server	Client(User)
Yang [7]	$4C_{pm}$+$2C_{pa}$+$4C_h$	$4C_{pm}$+$2C_{pa}$+$4C_h$
Yoon [8]	$4C_{pm}$+$2C_{pa}$+$4C_h$	$4C_{pm}$+$2C_{pa}$+$4C_h$
Debiao [9]	$3C_{pm}$+$3C_h$+$2C_{mac}$	$3C_{pm}$+$2C_h$+$2C_{mac}$
Goutham [11]	$3C_{pm}$+$4C_h$+$2C_{mac}$	$3C_{pm}$+$3C_h$+$2C_{mac}$
This Paper	$2C_{pm}$+C_{pa}+$4C_h$+$3C_{mac}$	$2C_{pm}$+C_{pa}+$3C_h$+$2C_{mac}$

VI. PERFORMANCE ANALYSIS

For the convenience of evaluating the computational cost of our scheme in client and server, we define some notations as follows.

- C_{pm} : Computation time of a scalar multiplication on the elliptic curve.
- C_{pa} : Computation time of a point addition on the elliptic curve.
- C_h : Computation time of a one-way hash function or a map-to-point hash function.
- C_{mac} : Computation time of a message authentication code.

In Table II, we summarize the result of mutual authentication with key agreement phase in some similar proposed protocols. the consumption is mainly determined by above units, C_{pm}, C_{pa}, C_h and C_{mac}. In general, C_M is the most expensive execution among these computations and a m-bit scalar multiplication, in double-and-add method to compute, costs about $\frac{m}{2}A + mD$, where A and D mean point addition and point doubling respectively. So, if we could reduce the point multiplication operation, a lot of calculation will be saved. The proposed method uses point addition instead of point multiplication because client private key D_{C_i} is a secret value for adversary. So we could reduce the consumption of client whose resource is more precious than server. As shown in the table, the proposed scheme uses less calculation compared to other schemes, especially for client.

Another factor affected the performance is the length of communication message. The more data the message has, the more power is consumed to send it. So, compared with Goutham et al.'s protocol, M_1 message of mutual authentication phase in our proposed scheme reduces USN and $Rand$ value, which saves more energy for client. It is useful for client-server environment, wireless communication in particular.

VII. Conclusion

In this paper, we propose a new authentication with key agreement protocol based on ECC, which is suitable for resource-constrained device or other low-power IoT end device. The proposed mutual authentication with key agreement protocol fixes some drawbacks in Debiao et al. and Goutham et al.'s schemes such as impersonation attack, private updating problem and incomplete anonymity. It not only inherits merits of previous scheme, but also provides higher level security and reduces the consumption of time and power, which is more precious in low-power devices. Compared with other related work, our scheme is more efficient and practical in the smart card, client-server environment, wireless sensor networks(WSN) and other applications of IoT.

Acknowledgment

This work is supported by National Natural Science Foundation of China(No. 61379139 and No. 61602468), the Strategic Priority Research Program of the Chinese Academy of Sciences, Grant No. XDA06010701, and the State Grid Science and Technology project No. JL71-15-038. Many thanks go to the anonymous reviewers for their detailed comments and suggestions.

References

[1] D. Hankerson, A. J. Menezes, and S. Vanstone, *Guide to elliptic curve cryptography*. Springer Science & Business Media, 2006.

[2] Q. Huang, J. Cukier, H. Kobayashi, B. Liu, and J. Zhang, "Fast authenticated key establishment protocols for self-organizing sensor networks," in *Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*. ACM, 2003, pp. 141–150.

[3] X. Tian, D. S. Wong, and R. W. Zhu, "Analysis and improvement of an authenticated key exchange protocol for sensor networks," *IEEE Communications letters*, vol. 9, no. 11, pp. 970–972, 2005.

[4] A. G. Reddy, E.-J. Yoon, A. K. Das, and K.-Y. Yoo, "Lightweight authentication with key-agreement protocol for mobile network environment using smart cards," *IET Information Security*, vol. 10, no. 5, pp. 272–282, 2016.

[5] M.-C. Chuang and M. C. Chen, "An anonymous multi-server authenticated key agreement scheme based on trust computing using smart cards and biometrics," *Expert Systems with Applications*, vol. 41, no. 4, pp. 1411–1418, 2014.

[6] Y. Yang, H. Cai, Z. Wei, H. Lu, and K.-K. R. Choo, "Towards lightweight anonymous entity authentication for iot applications," in *Australasian Conference on Information Security and Privacy*. Springer, 2016, pp. 265–280.

[7] J.-H. Yang and C.-C. Chang, "An id-based remote mutual authentication with key agreement scheme for mobile devices on elliptic curve cryptosystem," *Computers & security*, vol. 28, no. 3, pp. 138–143, 2009.

[8] E.-J. Yoon and K.-Y. Yoo, "Robust id-based remote mutual authentication with key agreement scheme for mobile devices on ecc," in *Computational Science and Engineering, 2009. CSE'09. International Conference on*, vol. 2. IEEE, 2009, pp. 633–640.

[9] H. Debiao, C. Jianhua, and H. Jin, "An id-based client authentication with key agreement protocol for mobile client–server environment on ecc with provable security," *Information Fusion*, vol. 13, no. 3, pp. 223–230, 2012.

[10] S. H. Islam and G. Biswas, "Comments on id-based client authentication with key agreement protocol on ecc for mobile client-server environment," in *International Conference on Advances in Computing and Communications*. Springer, 2011, pp. 628–635.

[11] R. A. Goutham, G.-J. Lee, and K.-Y. Yoo, "An anonymous id-based remote mutual authentication with key agreement protocol on ecc using smart cards," in *Proceedings of the 30th Annual ACM Symposium on Applied Computing*. ACM, 2015, pp. 169–174.

[12] V. S. Miller, "Use of elliptic curves in cryptography," in *Conference on the Theory and Application of Cryptographic Techniques*. Springer, 1985, pp. 417–426.

[13] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of computation*, vol. 48, no. 177, pp. 203–209, 1987.