Special Collection Article

International Journal of
Distributed
Sensor Networks

International Journal of Distributed Sensor Networks
2018, Vol. 14(1)
© The Author(s) 2018
DOI: 10.1177/1550147718754793
journals.sagepub.com/home/dsn
SAGE

An authentication and key agreement mechanism for OPC Unified Architecture in industrial Internet of Things

Min Wei[1], Shuaidong Zhang[1], Ping Wang[1] and Keecheon Kim[2]

Abstract

As an industrial communication data interaction specification, OPC Unified Architecture effectively solves the industrial Internet of Things system integration problem. This article designs an authentication and key agreement scheme based on implicit certificate using the security model provided by OPC Unified Architecture. It establishes the secure channel and provides a guarantee for secure session for the OPC Unified Architecture server and client. Then, the test verification platform is implemented to verify the feasibility of the scheme. The result shows that the mechanism is feasible, and the system security and availability are effectively improved.

Keywords

OPC Unified Architecture, industrial Internet of Things, security, authentication, key agreement

Date received: 12 June 2017; accepted: 19 December 2017

Handling Editor: Jing Liu

Introduction

Industrial Internet of Things (IIoT) is a network of field devices, actuator, and intelligent computers that collect and share huge amounts of data in manufacturing. Connectivity is one of the foundational technologies enabling data sharing among participating components of an IIoT system. As a connectivity framework standard, OPC Unified Architecture (OPC UA) facing device interchangeability issues has been used in the manufacturing industry. OPC UA is designed to support multiple transports and can provide a consistent and complete address space and service model, which contains a series of services. The OPC Foundation[1] maintains the OPC UA family of specifications.

The openness of IIoT incurs more threats and risks, and security issues have become the main obstacle to the development of IIoT. In order to ensure the confidentiality, integrity, and availability between client and server, OPC UA specification provides a security framework for the secure and reliable transmission from manufacturing level to production planning or enterprise management level.[2] However, IIoT system components run on a variety of platforms, from small resource-constrained to enterprise-class machines. It is challenging to develop light security mechanism to support the devices being used, in which the computing capability and storage capacity are limited. For the security, it is urgent to design and implement an effective authentication and key agreement mechanism for resource-constrained devices based on OPC UA.

[1]National Industrial Internet of Things International S&T Cooperation Base, Chongqing University of Posts and Telecommunications, Chongqing, P.R. China
[2]Department of Computer Science and Engineering, Konkuk University, Seoul, Republic of Korea

Corresponding author:
Min Wei, National Industrial Internet of Things International S&T Cooperation Base, Chongqing University of Posts and Telecommunications, 2 Chongwen Road, Chongqing 400065, P.R. China.
Email: thinker9@163.com

Several previous studies have been conducted on this issue. S Cavalieri and G Cutuli[3] aimed to deal with the performance evaluation of OPC UA, considering both security and subscription mechanisms. A Fernbach and W Kastner[4] aimed to give a discussion about how the structure of the environmental system where OPC UA applications are installed, the resources available, and a request to dependability shall affect the strategy of managing certificates, which provided a theoretical basis for the optimization of OPC UA security mechanism. O Post et al.[5,6] discussed the security at device level of automation system and implemented a new authentication-only security policy profile or using OPC UA for data transfer and IPsec for authentication is proposed. KH Wu et al.[7] proposed a new security communication model to provide integrity and authentication in OPC UA; through this model that uses the Whirlpool hash function to check integrity and generates digital signature along with Rivest–Shamir–Adleman (RSA) in message transmission, terminals in the upper layer can communicate with field devices via a channel with high security and efficiency. Given the security of RSA, the key length needs to be increased to enhance the security strength of RSA.[8] It may limit the application of RSA in the system with limited computing capability. Several studies have been conducted on implicit certificates. S Sciancalepore and colleagues[9,10] proposed a key management protocol which suitably integrates implicit certificates with Elliptic Curve Diffie–Hellman (ECDH) exchange in the IEEE 802.15.4 protocol. Park[11] proposed an ECQV certificate issuance protocol that addresses the problems of the previous PAuthKey protocol. DA Ha et al.[12] designed and implemented a security scheme based on ECQV Implicit Certificates and compared the execution time of public key extraction operation with the RSA certificates.

At present, with the rapid development of IIoT, a large number of resource-constrained devices with limited communication, energy, and bandwidth are used. According to the security model of OPC UA, it is the key of solving problem to design and implement a low overhead authentication and key agreement mechanism. In this article, according to the security model defined by the OPC UA specification, a new authentication and key agreement mechanism based on implicit certificate is designed for authentication requirements between OPC UA client and server.

OPC UA security mechanism

OPC UA divides system software into clients and servers. The servers usually reside on a device; they provide a way to access the device through a standard "device model." The servers expose an object-oriented, remotely callable Application Programming Interface (API) that implements the device model. Certificate authority (CA) is responsible for issuing digital certificates for OPC UA clients and servers.

OPC UA security model is proposed in the OPC UA specification part 2.[13] In order to implement the OPC UA security model, five security service functions are provided in the OPC UA security specification, which are related to the security channel and the session.

The establishment of secure channel is the basis of OPC UA security mechanism. The purpose of secure channel establishment is to derive a symmetric key by exchanging private information between OPC UA client and server. Only the connection between the client and the server is established; the message transmission, encryption, and decryption works will be executed. The specific process of establishing a secure channel is shown in Figure 1.

RSA algorithm is used for signing or encryption in OPC UA. However, a large number of numerical operations using RSA algorithm are needed in the process of calculation of encryption, decryption, signature, and verifying signature. In addition, OPC UA relies on the Public Key Infrastructure (PKI) to manage keys used for symmetric and asymmetric encryption, which requires complex certificate management, high computing, and storage overhead. It is difficult to be implemented and used in the resource-constrained and highly real-time environment.

Security authentication and key agreement mechanism

Authentication and key agreement mechanism based on implicit certificate

In this article, an authentication and key agreement scheme-based implicit certificate is proposed, which

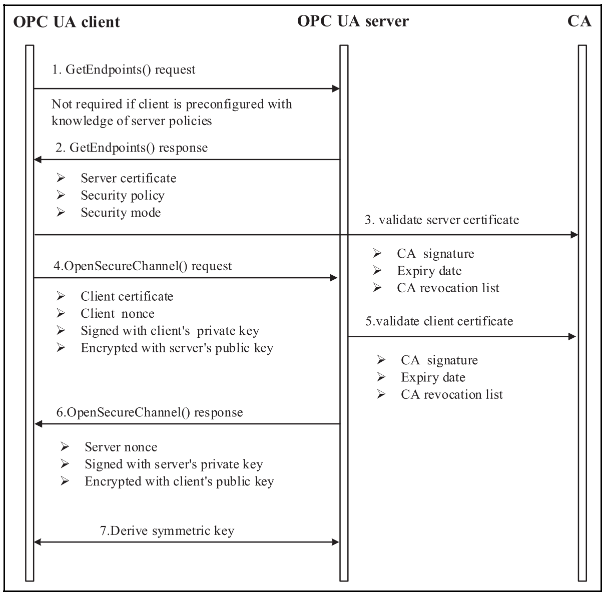

OPC UA client
OPC UA server
CA
1. GetEndpoints() request
Not required if client is preconfigured with knowledge of server policies
2. GetEndpoints() response
➢ Server certificate
➢ Security policy
➢ Security mode
3. validate server certificate
➢ CA signature
➢ Expiry date
➢ CA revocation list
4.OpenSecureChannel() request
➢ Client certificate
➢ Client nonce
➢ Signed with client's private key
➢ Encrypted with server's public key
5.validate client certificate
➢ CA signature
➢ Expiry date
➢ CA revocation list
6.OpenSecureChannel() response
➢ Server nonce
➢ Signed with server's private key
➢ Encrypted with client's public key
7.Derive symmetric key

Figure 1. The process of establishing a secure channel.

Table 1. Symbols.

Symbol	Description
ID	The identity information of the certificate requestor.
K_{CA}	The public key of certificate authority.
k_{CA}	The private key of certificate authority.
K	The random number of the certificate requestor.
N_{CA}	The random number of certificate authority.
G	The base point of an elliptic curve.
MIC	Integrity check code based on hash function.
Encode	Certificate encoding mode.
Text	The basic information of certificate.
Cert	The implicit certificate.
y	The private key of certificate requestor.
Y	The public key of certificate requestor.
Hash()	The hash function.
D	The factor of implicit certificate.
$Cert_A$	The implicit certificate of OPC UA client A.
$Cert_B$	The implicit certificate of OPC UA server B.
R_1	The random number of OPC UA client A.
R_2	The random number of OPC UA server B.
R_3	The random number of OPC UA client A.
R_4	The random number of OPC UA server B.
(*Q*, *q*)	The public–private key pair of OPC UA client A.
(*M*, *m*)	The public–private key pair of OPC UA server B.
L	Temporary public key of OPC UA client A.
L_1	Temporary public key of OPC UA server B.
||	String connector.
E_h	A hash operation.
E_d	A point addition operation.
E_s	A scalar multiplication operation.
R_s	A signature operation based on RSA.
R_y	A validation signature operation based on RSA.
R_j	An encryption operation based on RSA.
R_d	A decryption operation based on RSA.

UA: Unified Architecture; RSA: Rivest–Shamir–Adleman.

consists of two parts: the generation of implicit certificate and the establishment of secure channel. The symbols used in this scheme are described in Table 1.

The generation of implicit certificate. The generation process of implicit certificate is similar to digital certificates. First of all, OPC UA client or server generates the material used to construct the public key and transfers the public key material and the identity information ID to the CA. Then, CA begins to verify the identity information ID; if the verification is successful, the next operation is performed, otherwise the interaction is terminated. According to the public key material, CA constructs an implicit certificate which contains the parameters to reconstruct the public key. The specific process is shown in Figure 2, which includes three steps:

Step 1. First, requestor U produces a large prime number K as its own private key and calculates its own public key $R = KG$. Second, according to hash function, an integrity check code is calculated, $MIC = Hash(ID||R)$. Finally, requestor U generates a request for an implicit certificate, which is sent to CA.

Step 2. After CA receives the request message, the integrity check code is verified; if the verification is not successful, the communication is terminated; conversely, the following operations will be performed:

- CA generates a random number N_{CA} and calculates the implicit certificate factor $D = R + N_{CA}G$.

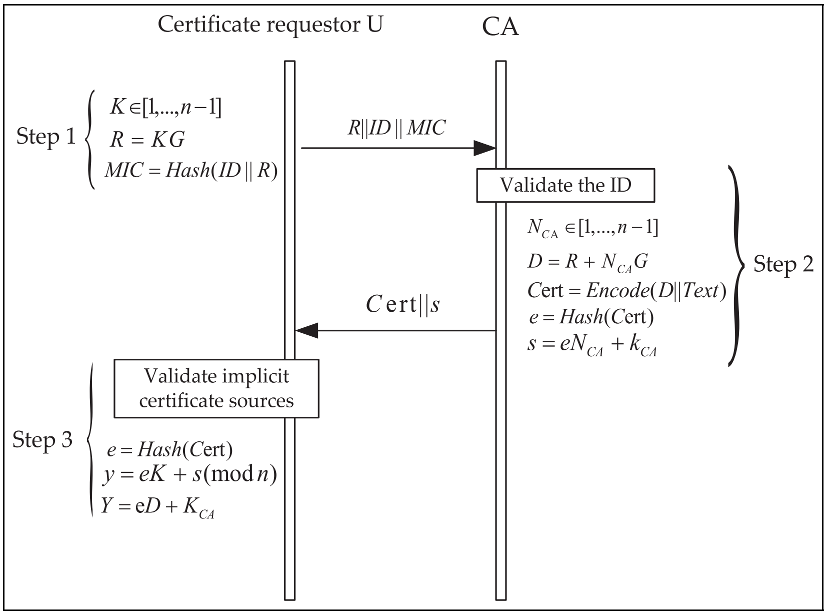


Figure 2. The issuance process of implicit certificate.

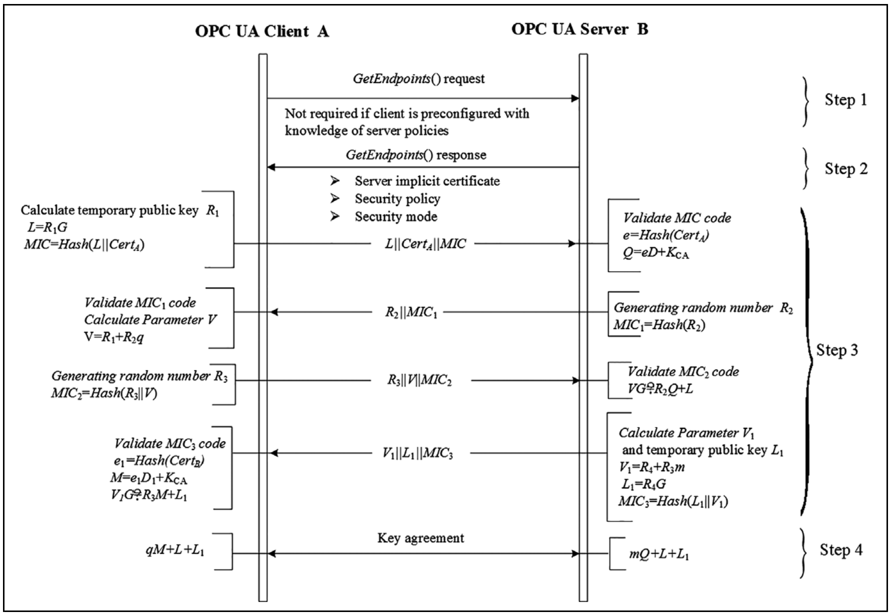


Figure 3. The process of secure channel establishment.

- According to the implicit certificate factor, an implicit certificate is constructed, $Cert = Encode(D,Text)$; Text is the basic information of the certificate, such as requestor identity information and serial number of certificate.
- Calculate the abstract value of implicit certificate $e = Hash(Cert)$ and signature $s = eN_{CA} + k_{CA}(mod\, n)$.

Finally, CA generates a response message for an implicit certificate, which is sent to requestor U.

Step 3. After requestor U receives the response message, the source of the certificate needs to be verified by calculating $Hash(Cert)R + sG = Hash(Cert)D + K_{CA}$; if the verification is not successful, the communication is terminated, otherwise requestor U can calculate own public key pair by the implicit certificate and signature:

Private key: $y = eK + s(mod\, n)$
Public key: $Y = eD + K_{CA}$

Establishment of secure channel. An authentication and key agreement scheme based on implicit certificate is proposed in this section. It ensures OPC UA secure channel can be successfully established. It is assumed that the OPC UA client is A and the server is B in IIoT. The specific scheme is shown in Figure 3, which includes four steps:

Step 1. GetEndpoints request service

OPC UA client A reads the endpoint information of OPC UA server B by GetEndpoints request service. Not required if OPC UA client A is preconfigured with knowledge of server policies.

Step 2. GetEndpoints response service

After OPC UA server B receives the GetEndpoints request service, the security configuration information is sent to OPC UA client A by GetEndpoints response service. The security configuration information includes server certificate, security mode, and security policy.

Step 3. Two-way authentication

In the two-way authentication step, the process includes calculating temporary public key and validating integrity check codes for both sides:

- Calculate temporary public key

After OPC UA client A receives the GetEndpoints response service, OPC UA client A generates a random number R1. Moreover, the temporary public key of OPC UA client A is calculated as shown in formula (1)

$$L = R_1G \tag{1}$$

According to hash function, an integrity check code is calculated, $MIC = Hash(L||Cert_A)$. OPC UA client A generates a message $L||Cert_A||MIC$, which is sent to OPC UA server B.

- Validate *MIC* code

After OPC UA server B receives the message $L||Cert_A||MIC$, *RMIC* is calculated in a prescribed manner as shown in formula (2)

$$RMIC = Hash(L||Cert_A) \tag{2}$$

If *RMIC* and *MIC* are not equal, the message $L||Cert_A||MIC$ should be discarded; on the contrary, OPC UA client's public key is extracted according to the implicit certificate $Cert_A$, as shown in formula (3)

$$Q = eD + K_{CA} \tag{3}$$

where $e = Hash(Cert_A)$.

A random number R_2 is generated; according to hash function, then, an integrity check code is calculated, $MIC_1 = Hash(R_2)$. OPC UA server B generates a message $R_2||MIC_1$, which is sent to OPC UA client A.

- Validate MIC_1 code

After OPC UA client A receives the message $R_2||MIC_1$, $RMIC_1$ is calculated in a prescribed manner as shown in formula (4)

$$RMIC_1 = Hash(R_2) \tag{4}$$

If $RMIC_1$ and MIC_1 are not equal, the message $R_2||MIC_1$ should be discarded; on the contrary, parameter V is calculated as shown in formula (5)

$$V = R_1 + R_2q \tag{5}$$

where q is the client's private key.

A random number R_3 is generated; according to hash function, then, an integrity check code is calculated, $MIC_2 = Hash(R_3||V)$. OPC UA client A generates a message $R_3||V||MIC_2$, which is sent to OPC UA server B.

- Validate MIC_2 code

After OPC UA server B receives the message $R_3||V||MIC_2$, $RMIC_2$ is calculated in a prescribed manner as shown in formula (6)

$$RMIC_2 = Hash(R_3||V) \tag{6}$$

If $RMIC_2$ and MIC_2 are not equal, the message $R_3||V||MIC_2$ should be discarded; on the contrary, VG and $R_2Q + L$ are calculated and then judged to be equal; if not equal, the identity authentication of OPC UA client A fails, and the communication is directly terminated. On the contrary, OPC UA server B generates a new random number R_4, and the temporary public key L_1 and parameter V_1 are calculated as shown in formulas 7 and (8)

$$L_1 = R_4G \tag{7}$$

$$V_1 = R_4 + R_3m \tag{8}$$

where m is the own private key; according to hash function, an integrity check code is calculated, $MIC_3 = Hash(V_1||L_1)$. OPC UA server B generates a message $V_1||L_1||MIC_3$, which is sent to OPC UA client A.

- Validate MIC_3 code

After OPC UA server B receives the message $V_1||L_1||MIC_3$, $RMIC_3$ is calculated in a prescribed manner as shown in formula (9)

$$RMIC_3 = Hash(V_1||L_1) \tag{9}$$

If $RMIC_3$ and MIC_3 are not equal, the message $V_1||L_1||MIC_3$ should be discarded.

OPC UA server's public key is extracted according to the implicit certificate $Cert_B$, as shown in formula (10)

$$M = e_1D_1 + K_{CA} \tag{10}$$

where $e_1 = Hash(Cert_B)$.

Whereafter, V_1G and $R_3M + L_1$ are calculated and then judged to be equal; if not equal, the identity authentication of OPC UA server B fails, and the communication is directly terminated. On the contrary, two-way authentication is completed.

Step 4. Key agreement

If the two-way authentication is completed, a symmetric key that can be used for message encryption or signature in the session service can be derived based on ECDH key exchange protocol, as shown in formula (11)

$$qM + L + L_1 = mQ + L + L_1 \tag{11}$$

Scheme testing and result analysis

Testing platform implementation

In order to verify the authentication and key agreement scheme which has been designed and developed in this article, the testing platform is built, as shown in Figure 4.

For the purpose of performance evaluation, we run the implementation of the proposed scheme in the following platform: RT5350 device (360 MHz MIPS24KEc CPU core, 32 MB RAM), CA runs on a resource-rich computer with Intel® Core(5) Quad CPU Q9400 2.6 GHz and 4 GB RAM, and implicit certificate application client run on a resource-rich computer with Intel Core(3) Quad CPU Q9400 1.8 GHz and 2 GB RAM. PC plays the role of OPC UA client and

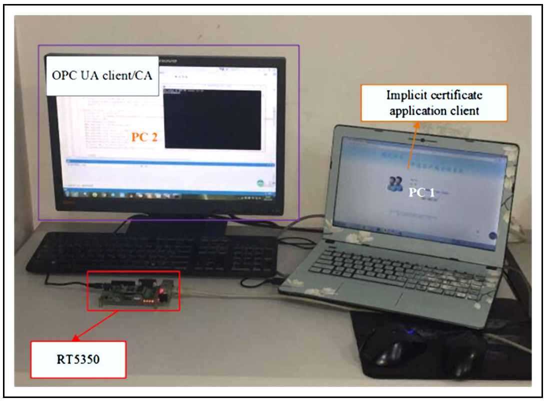


Figure 4. The testing platform.

RT5350 plays the role of OPC UA server. Wireshark is used to sniff the interaction message between OPC UA client and server. OPC UA stack includes OpenSSL[14] library that provide the API of many algorithms.

The proposed scheme adopts ECC (Elliptic Curve Diffie–Hellman) 160 algorithm. The hash algorithm is implemented by invoking the SHA-1 using the OpenSSL library [OpenSSL 2014] on PC. And we use our own software to implement HASH SHA-1 on RT5350 device.

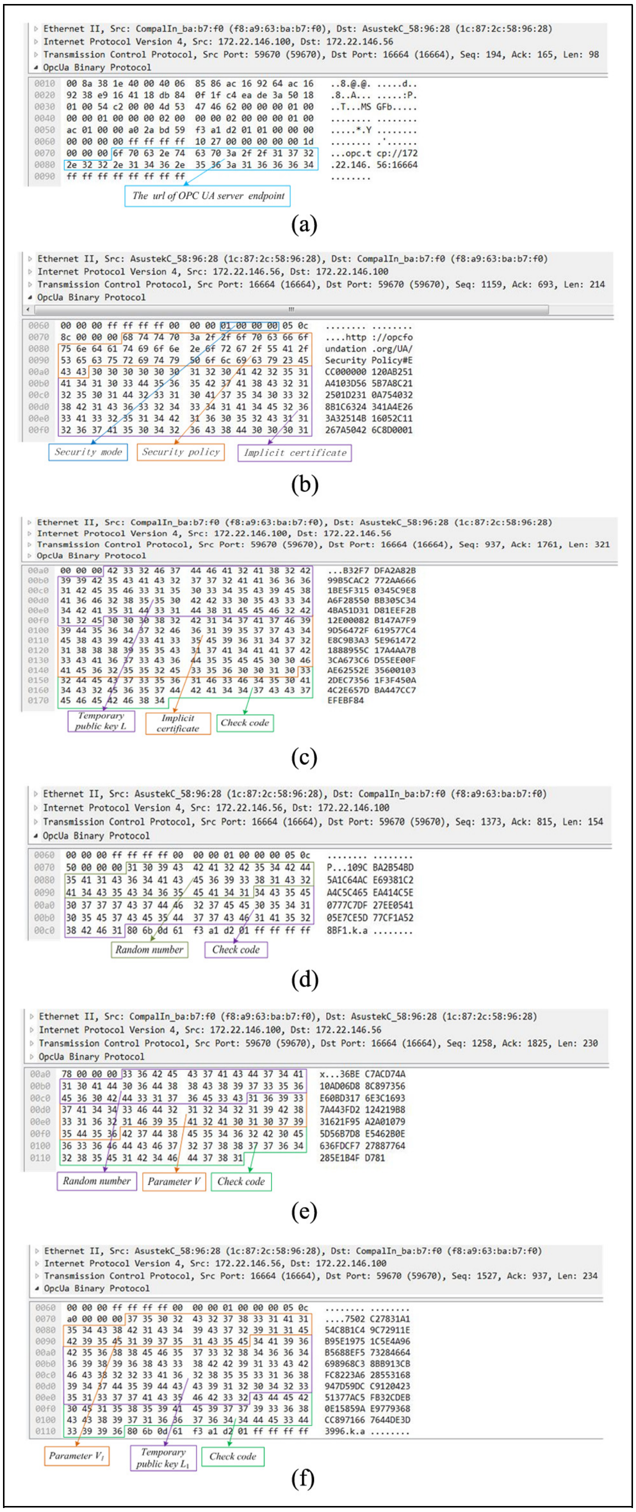
According to the testing platform, the authentication and key agreement scheme designed in this article is implemented.

This scheme mainly contains the following:

1. GetEndpoints request service and GetEndpoints response service.
2. Two-way authentication between OPC UA client and server.

The purpose of the test is to make sure that this authentication and key agreement scheme works in accordance with the design process. GetEndpoints request message information, GetEndpoints response message information, and four authentication messages are captured by Wireshark capture tool. The result is shown in Figure 5.

GetEndpoints request service is implemented correctly according to Figure 5(a); GetEndpoints response service is also implemented correctly according to Figure 5(b); similarly, two-way authentication service is implemented correctly according to Figure 5(c)–(f).



```
▷ Ethernet II, Src: CompalIn_ba:b7:f0 (f8:a9:63:ba:b7:f0), Dst: AsustekC_58:96:28 (1c:87:2c:58:96:28)
▷ Internet Protocol Version 4, Src: 172.22.146.100, Dst: 172.22.146.56
▷ Transmission Control Protocol, Src Port: 59670 (59670), Dst Port: 16664 (16664), Seq: 194, Ack: 165, Len: 98
◢ OpcUa Binary Protocol

0010  00 8a 38 1e 40 00 40 06  85 86 ac 16 92 64 ac 16   ..8.@.@. .....d..
0020  92 38 e9 16 41 18 db 84  0f 1f c4 ea de 3a 50 18   .8..A... .....:P.
0030  01 00 54 c2 00 00 4d 53  47 46 62 00 00 00 01 00   ..T...MS GFb.....
0040  00 00 01 00 00 00 02 00  00 00 02 00 00 00 01 00   ........ ........
0050  ac 01 00 00 a0 2a bd 59  f3 a1 d2 01 01 00 00 00   .....*.Y ........
0060  00 00 00 00 ff ff ff ff  10 27 00 00 00 00 00 1d   ........ .'......
0070  00 00 00 6f 70 63 2e 74  63 70 3a 2f 2f 31 37 32   ...opc.t cp://172
0080  2e 32 32 2e 31 34 36 2e  35 36 3a 31 36 36 36 34   .22.146. 56:16664
0090  ff ff ff ff ff ff ff ff                            ........
```

```
▷ Ethernet II, Src: AsustekC_58:96:28 (1c:87:2c:58:96:28), Dst: CompalIn_ba:b7:f0 (f8:a9:63:ba:b7:f0)
▷ Internet Protocol Version 4, Src: 172.22.146.56, Dst: 172.22.146.100
▷ Transmission Control Protocol, Src Port: 16664 (16664), Dst Port: 59670 (59670), Seq: 1159, Ack: 693, Len: 214
◢ OpcUa Binary Protocol

0060  00 00 00 ff ff ff ff 00  00 00 01 00 00 00 05 0c   ........ ........
0070  8c 00 00 00 68 74 74 70  3a 2f 2f 6f 70 63 66 6f   ....http ://opcfo
0080  75 6e 64 61 74 69 6f 6e  2e 6f 72 67 2f 55 41 2f   undation .org/UA/
0090  53 65 63 75 72 69 74 79  50 6f 6c 69 63 79 23 45   Security Policy#E
00a0  43 43 30 30 30 30 30 30  31 32 30 41 42 32 35 31   CC000000 120AB251
00b0  41 34 31 30 33 44 35 36  35 42 37 41 38 43 32 31   A4103D56 5B7A8C21
00c0  32 35 30 31 44 32 33 31  30 41 37 35 34 30 33 32   2501D231 0A754032
00d0  38 42 31 43 36 33 32 34  33 34 31 41 34 45 32 36   8B1C6324 341A4E26
00e0  33 41 33 32 35 31 34 42  31 36 30 35 32 43 31 31   3A32514B 16052C11
00f0  32 36 37 41 35 30 34 32  36 43 38 44 30 30 30 31   267A5042 6C8D0001
```

```
▷ Ethernet II, Src: CompalIn_ba:b7:f0 (f8:a9:63:ba:b7:f0), Dst: AsustekC_58:96:28 (1c:87:2c:58:96:28)
▷ Internet Protocol Version 4, Src: 172.22.146.100, Dst: 172.22.146.56
▷ Transmission Control Protocol, Src Port: 59670 (59670), Dst Port: 16664 (16664), Seq: 937, Ack: 1761, Len: 321
▷ OpcUa Binary Protocol

00a0  00 00 00 42 33 32 46 37  44 46 41 32 41 38 32 42   ...B32F7 DFA2A82B
00b0  39 39 42 35 43 41 43 32  37 37 32 41 41 36 36 36   99B5CAC2 772AA666
00c0  31 42 45 35 46 33 31 35  30 33 34 35 43 39 45 38   1BE5F315 0345C9E8
00d0  41 36 46 32 38 35 35 30  42 42 33 30 35 43 33 34   A6F28550 BB305C34
00e0  34 42 41 35 31 44 33 31  44 38 31 45 45 46 32 42   4BA51D31 D81EEF2B
00f0  31 32 45 30 30 30 38 32  42 31 34 37 41 37 46 39   12E00082 B147A7F9
0100  39 44 35 36 34 37 32 46  36 31 39 35 37 37 43 34   9D56472F 619577C4
0110  45 38 43 39 42 33 41 33  35 45 39 36 31 34 37 32   E8C9B3A3 5E961472
0120  31 38 38 38 39 35 35 43  31 37 41 34 41 41 37 42   1888955C 17A4AA7B
0130  33 43 41 36 37 33 43 36  44 35 35 45 45 30 30 46   3CA673C6 D55EE00F
0140  41 45 36 32 35 35 32 45  33 35 36 30 30 31 30 33   AE62552E 35600103
0150  32 44 45 43 37 33 35 36  31 46 33 46 34 35 30 41   2DEC7356 1F3F450A
0160  34 43 32 45 36 35 37 44  42 41 34 34 37 43 43 37   4C2E657D BA447CC7
0170  45 46 45 42 46 38 34                               EFEBF84
```

```
▷ Ethernet II, Src: AsustekC_58:96:28 (1c:87:2c:58:96:28), Dst: CompalIn_ba:b7:f0 (f8:a9:63:ba:b7:f0)
▷ Internet Protocol Version 4, Src: 172.22.146.56, Dst: 172.22.146.100
▷ Transmission Control Protocol, Src Port: 16664 (16664), Dst Port: 59670 (59670), Seq: 1373, Ack: 815, Len: 154
◢ OpcUa Binary Protocol

0060  00 00 00 ff ff ff ff 00  00 00 01 00 00 00 05 0c   ........ ........
0070  50 00 00 00 31 30 39 43  42 41 32 42 35 34 42 44   P...109C BA2B54BD
0080  35 41 31 43 36 34 41 43  45 36 39 33 38 31 43 32   5A1C64AC E69381C2
0090  41 34 43 35 43 34 36 35  45 41 34 31 34 43 35 45   A4C5C465 EA414C5E
00a0  30 37 37 37 43 37 44 46  32 37 45 45 30 35 34 31   0777C7DF 27EE0541
00b0  30 35 45 37 43 45 35 44  37 37 43 46 31 41 35 32   05E7CE5D 77CF1A52
00c0  38 42 46 31 80 6b 0d 61  f3 a1 d2 01 ff ff ff ff   8BF1.k.a ........
```

```
▷ Ethernet II, Src: CompalIn_ba:b7:f0 (f8:a9:63:ba:b7:f0), Dst: AsustekC_58:96:28 (1c:87:2c:58:96:28)
▷ Internet Protocol Version 4, Src: 172.22.146.100, Dst: 172.22.146.56
▷ Transmission Control Protocol, Src Port: 59670 (59670), Dst Port: 16664 (16664), Seq: 1258, Ack: 1825, Len: 230
▷ OpcUa Binary Protocol

00a0  78 00 00 00 33 36 42 45  43 37 41 43 44 37 34 41   x...36BE C7ACD74A
00b0  31 30 41 44 30 36 44 38  38 43 38 39 37 33 35 36   10AD06D8 8C897356
00c0  45 36 30 42 44 33 31 37  36 45 33 43 31 36 39 33   E60BD317 6E3C1693
00d0  37 41 34 34 33 46 44 32  31 32 34 32 31 39 42 38   7A443FD2 124219B8
00e0  33 31 36 32 31 46 39 35  41 32 41 30 31 30 37 39   31621F95 A2A01079
00f0  35 44 35 36 42 37 44 38  45 35 34 36 32 42 30 45   5D56B7D8 E5462B0E
0100  36 33 36 46 44 43 46 37  32 37 38 38 37 37 36 34   636FDCF7 27887764
0110  32 38 35 45 31 42 34 46  44 37 38 31               285E1B4F D781
```

```
▷ Ethernet II, Src: AsustekC_58:96:28 (1c:87:2c:58:96:28), Dst: CompalIn_ba:b7:f0 (f8:a9:63:ba:b7:f0)
▷ Internet Protocol Version 4, Src: 172.22.146.56, Dst: 172.22.146.100
▷ Transmission Control Protocol, Src Port: 16664 (16664), Dst Port: 59670 (59670), Seq: 1527, Ack: 937, Len: 234
◢ OpcUa Binary Protocol

0060  00 00 00 ff ff ff ff 00  00 00 01 00 00 00 05 0c   ........ ........
0070  a0 00 00 00 37 35 30 32  43 32 37 38 33 31 41 31   ....7502 C27831A1
0080  35 34 43 38 42 31 43 34  39 43 37 32 39 31 31 45   54C8B1C4 9C72911E
0090  42 39 35 45 31 39 37 35  31 43 35 45 34 41 39 36   B95E1975 1C5E4A96
00a0  42 35 36 38 38 45 46 35  37 33 32 38 34 36 36 34   B5688EF5 73284664
00b0  36 39 38 39 36 38 43 33  38 42 42 39 31 33 43 42   698968C3 8BB913CB
00c0  46 43 38 32 32 33 41 36  32 38 35 35 33 31 36 38   FC8223A6 28553168
00d0  39 34 37 44 35 39 44 43  43 39 31 32 30 34 32 33   947D59DC C9120423
00e0  35 31 33 37 37 41 43 35  46 42 33 32 43 44 45 42   51377AC5 FB32CDEB
00f0  30 45 31 35 38 35 39 41  45 39 37 37 39 33 36 38   0E15859A E9779368
0100  43 43 38 39 37 31 36 36  37 36 34 34 44 45 33 44   CC897166 7644DE3D
0110  33 39 39 36 80 6b 0d 61  f3 a1 d2 01 ff ff ff ff   3996.k.a ........
```

Figure 5. (a) The message information of GetEndpoints request, (b) the message information of GetEndpoints response, (c) first authentication message information, (d) second authentication message information, (e) third authentication message information, and (f) fourth authentication message information.

Result analysis

In order to evaluate the authentication and key agreement scheme, some performance indices are chosen, which are classified into security and effectiveness according to Fan and Lu[15] and Yang et al.[16] The security performance indices include forward security, known key security, non-key control, zero knowledge of interaction information, and the ability to resist attacks; the effectiveness performance indices include computing overhead, communication overhead, and storage overhead. Next, this scheme will be discussed in terms of security and effectiveness.

The security analysis. This scheme ensures that IIoT has the forward security, the known key security, the zero knowledge of interaction information, and the non-key control ability; meanwhile, this scheme can resist the attack of the middleman attack, replay attack, and so on.

In this scheme, different random values are used in each key agreement process, such as temporary public key L and L_1. In the case of long term, private key of OPC UA client A and server B is obtained by the attacker, and the attacker cannot derive the key obtained before the disclosure of the private key, thus this scheme ensures the forward security.

Even though attackers know last session key, owing to the change in temporary public key L_1 and the confidentiality of private key, the attacker is unable to push the current session key, and therefore, this scheme has known key security.

Meanwhile, the attacker is unable to calculate the symmetric key by these interaction information. Thus, this scheme has zero knowledge of interaction information. The symmetric key generation depends on the temporary public key of OPC UA client A and server B, long-term public keys Q and M, and the private key information q and m, which shows that OPC UA client A and server B make an equal contribution in the symmetric key generation. Therefore, this scheme has the ability of non-key control.

The middleman attack is the biggest security vulnerability of the ECDH key exchange protocol, and the reason is lack of the identity authentication of the participants. In this scheme, if communication parties want to establish communication, the identity authentication must be performed. In the process of identity authentication and key agreement, some private parameters are used, such as random numbers R_1 and R_4 and private keys q and m. The attacker is unable to obtain these private parameters, that is, these effective consultation information cannot be forged. Replay attacks can also be resisted using random numbers.

The effectiveness analysis. The performance analysis of authentication and key agreement scheme based on implicit certificate mainly consists of three parts: computation overhead, communication overhead, and storage overhead. Furthermore, this article also increases the comparison with other schemes.

Table 2. Computational overhead comparison result.

Object	This scheme	Original OPC UA scheme
OPC UA server	$4E_h + 2E_d + 4E_s$	$R_s + R_d + R_j + 2R_y$

UA: Unified Architecture.

- Computational overhead analysis

First of all, the computational overhead of this scheme is compared with OPC UA scheme. The theoretical comparison result is shown in Table 2.

As can be seen from Table 2, the computational overhead is the sum of $4E_h$, $2E_d$, and $4E_s$. According to ECC algorithm, the computational overhead of this scheme can be equivalent to twice cryptographic operations of ECC and $4E_h$. For OPC UA scheme, its computational overhead is the sum of $1R_s$, R_d, R_j, and $2R_y$. According to Verma et al.,[8] the encryption and signature operations of ECC algorithm are better than RSA algorithm under the same security intensity. Furthermore, there are hash operations in signature and validation signature operation. Therefore, the computational overhead of this scheme is acceptable.

Second, the actual computational cost of OPC UA server is also tested using the testing platform (see Figure 4). The main function of the scheme is to establish a secure channel, that is to say, the execution time of the program can be used to quantify the computational complexity of secure channel establishment. Therefore, time point T_1 is set when OPC UA client A initiates the GetEndpoints request; time point T_2 is set before completing the session service. $T_2 - \mathrm{T}_1$ is calculated.

Similarly, the computational overhead for the IoT device to establish secure channel can be obtained as shown in Figure 6.

The computational overhead cost is 310 ms when the security policy is chosen as Basic 256. The time cost of this proposed scheme is 390 ms using this ECC 160. In comparison, this proposed scheme computational overhead is higher than the Basic 256 security policy. As shown in Table 3, in Ha et al.,[12] which is implemented in Beaglebone black revision C boards (1 GHz ARM Cortex-A8, 512 MB RAM), the time that is needed for the IoT device using ECDH-ECQV to finish secure key establishment phase is 49.964 ms, and the time for DHE-RSA is 827.463 ms. Therefore, the handshake overhead of the proposed scheme is better than DHE-RSA and worse than the ECDH-ECQV in Ha et al.[12]

However, the security strength of ECC 160 is better than the Basic 256 security policy, and the key with 1024 bits length is no longer security for RSA algorithm.[17,18] Meanwhile, the IIoT devices rarely need to

Table 3. Scheme description.

Scheme	DHE-RSA[12]	ECDH-ECQV[12]	The proposed scheme ECC 160
Platform	Beaglebone boards	Beaglebone boards	RT5350
CPU	1 GHz	1 GHz	360 MHz
Overhead	0.827 s	0.04995 s	0.39 s

Table 4. Storage overhead comparison result.

Object	Security material	Length (byte)
This scheme	Private key	20
	Implicit certificate $Cert_B$	46
Original OPC UA scheme	Private key	128
	Digital certificate *Cert*	256
	Certificate trusted list	*a*
	Certificate revocation list	*b*

UA: Unified Architecture.

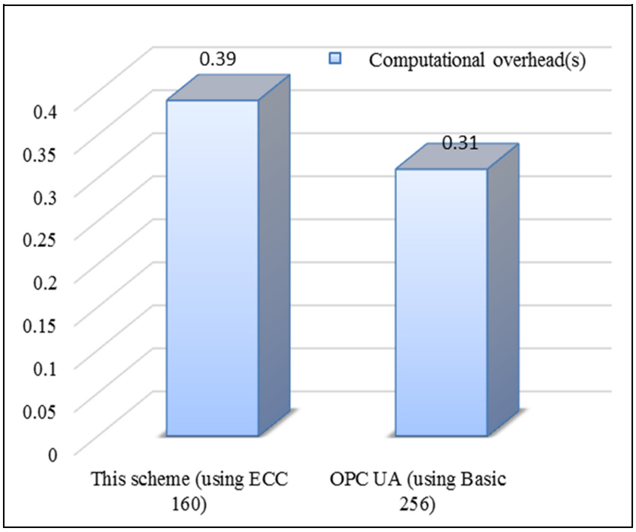


Figure 6. The comparison results of computational overhead.

perform this phase. Therefore, the proposed scheme is still a valid replacement for the scheme from OPC UA even though more computational overhead is costly. We will consider the ECDH-ECQV and other possible algorithms to be introduced in OPC UA and compare the work with them in same platform in the future.

- Storage overhead analysis

First of all, the storage overhead of this scheme is compared with original OPC UA scheme. The theoretical storage overhead of OPC UA server mainly considers some security material discussed in this above section. The theoretical comparison result is shown in Table 4.

As can be seen from Table 4, the storage overhead of OPC UA server is about 66 bytes in this scheme; the storage overhead of original OPC UA server is about ($384 + a + b$) bytes in OPC UA scheme, in which a and b depend on the network size. In addition, with the expansion of IIoT, the storage overhead of OPC UA scheme will continue to increase. Therefore, this scheme has better performance in terms of storage overhead.

Second, the actual storage overhead of OPC UA server is also tested using the testing platform (see Figure 4). The actual storage overhead mainly refers to the comparison of the code storage in two cases including the security mechanism and without the security mechanism. The specific testing process is as follows:

1. Compile the OPC UA protocol stack without the security mechanism to view code storage.
2. Compile the OPC UA protocol stack including the security mechanism to view code storage.
3. Compare the results from (1) and (2).

As shown in Figure 7, the code storage without the security mechanism is 468.5 kB, and the code storage with this scheme is 481.2 kB. The implementation for this proposed scheme based on implicit certificates requires 12.7 kB of ROM, whereas the protocol based on implicit certificates[9] requires 12.240 kB of ROM, and the protocol implementation based on X.509 explicit certificates[9] for benchmarking purposes uses 15.192 kB of ROM. The overhead in ROM for the proposed scheme is acceptable.

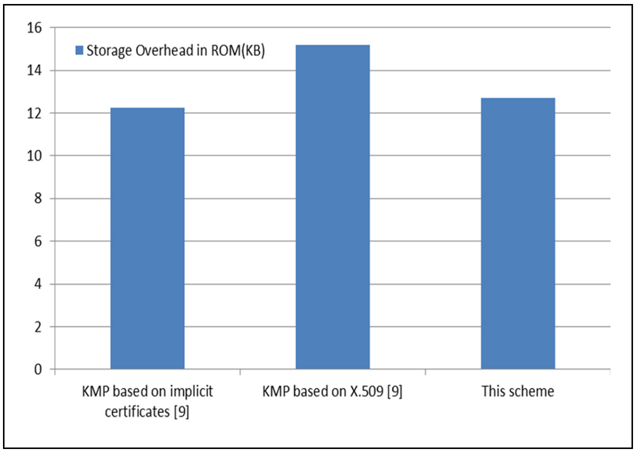


Figure 7. The storage overhead analysis result.

Table 5. Communication overhead comparison result.

Scheme	Interaction message	Data length (byte)
This scheme	$Cert_B$	46
	$R_2 \Vert MIC_1$	40
	$V_1 \Vert L_1 \Vert MIC_3$	80
Original OPC UA scheme	Digital certificate *Cert*	256
	$Epb(Nonce) \Vert Epv(Hash(Nonce))$	256

UA: Unified Architecture.

- Communication overhead analysis

The communication overhead of OPC UA server mainly comes from the message interaction in the authentication phase. Under the condition of security level 80, that is, the key length of ECC is 160 bits, the key length of RSA is 1024 bits. The communication overhead of the two schemes is compared as shown in Table 5.

As can be seen from Table 5, the communication overhead of OPC UA server is about 166 bytes in this scheme; the communication overhead of OPC UA server is about 512 bytes in original OPC UA security scheme. In summary, compared with the OPC UA scheme, this scheme has better performance. The actual communication overhead of OPC UA server is also tested using the testing platform (see Figure 4). The specific testing process is as follows:

1. When OPC UA protocol is not added to this scheme, the message information sent by OPC UA server is counted by Wireshark.
2. When OPC UA protocol is added to this scheme, the message information sent by OPC UA server is counted by Wireshark.
3. Compare the results from (1) and (2).

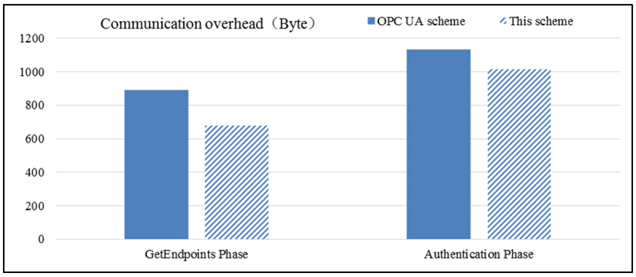


Figure 8. The communication overhead analysis result.

As can be seen from Figure 8, the communication overhead of this scheme is less than the communication overhead of OPC UA scheme in the GetEndpoints phase and authentication phase.

- Comparison analysis with other schemes

In order to verify adequately the feasibility and superiority of this scheme, the scheme is also compared with other schemes. In terms of performance, the comparison result is shown in Table 6.

As can be seen from Table 6, comparing with Hafizul Islam and Biswas,[19] twice hash and point addition operations are reduced in this scheme and one time scalar multiplication operation is added; as a whole, there is little difference in performance between this scheme and Hafizul Islam and Biswas.[19] Comparing with Yang,[20] one time hash operation is reduced in this scheme. Comparing with Liao and Hsiao,[21] one time point addition operation is added, but one time hash and three times scalar multiplication operations are reduced in this scheme. In summary, comparing with other schemes, this scheme has better performance.

In terms of security, Hafizul Islam and Biswas[19] and Liao and Hsiao[21] do not support two-way authentication. And there is little difference in security between this scheme and Yang.[20]

Conclusion and future work

In order to guarantee the security of IIoT especially for the resource-constrained environment, an authentication and key agreement mechanism based on implicit certificate is proposed based on OPC UA security model in this article. The implicit certificate is based on ECC cryptosystem. In the process of establishing secure channel, the lightweight ECC encryption algorithm is adopted to ensure the security of data transmission in the communication process. The mechanism is implemented and the test platform shows that this mechanism could be used in resource-constrained environments. In the future work, in order to further validate the feasibility of this scheme in resource-

Table 6. Scheme comparison result.

This scheme	Hafizul Islam and Biswas[19]	Yang[20]	Liao and Hsiao[21]
$4E_h + 2E_d + 5E_s$	$6E_h + 4E_d + 4E_s$	$5E_h + 2E_d + 5E_s$	$5E_h + 1E_d + 8E_s$

constrained environments, this scheme will be tested in practical applications. Currently, there are some studies on key agreement and authentication mechanism for constrained device in the IEEE 802.15.4 protocol stack.[10,11] Some device based on IEEE 802.15.4 supporting OPC UA has been implemented in our laboratory. The proposed scheme will be considered and additional implementation will be added in platform (such as STM32F103) in the future.

Acknowledgements

The authors would like to appreciate Zhuang Yuan for her contributions to this work.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported by Chongqing Education Committee Science and Technology Projects (KJ1600405), China; Chongqing Foundation and Frontier Project (cstc2017jcyjAX0235); and Chongqing IoT Industry Common Key Technology Innovation Project (cstc2015zdcy-ztzx70007).

References

1. OPC Foundation. OPC UA architecture specification: part 1—overview and concepts (Version 1.02), 10 July 2012, https://scadahacker.com/library/Documents/ICS_Protocols/OPCF%20-%20OPC-UA%20Part%201%20-%20Overview%20and%20Concepts%201.02%20Specification.pdf
2. Huang R, Liu F and Dongbo P. Research on OPC UA security. In: *Proceedings of the 2010 5th IEEE conference on industrial electronics and applications*, Taichung, Taiwan, 15–17 June 2010, pp.1439–1444. New York: IEEE.
3. Cavalieri S and Cutuli G. Performance evaluation of OPC UA. In: *Proceedings of the 2010 IEEE conference on emerging technologies and factory automation (ETFA)*, Bilbao, 13–16 September 2010, pp.1–8. New York: IEEE.
4. Fernbach A and Kastner W. Certificate management in OPC UA applications: an evaluation of different trust models. In: *Proceedings of the 2012 IEEE 17th conference on emerging technologies & factory automation (ETFA)*, Krakow, 17–21 September 2012, pp.1–6. New York: IEEE.
5. Post O, Seppälä J and Koivisto H. Certificate based security at device level of automation system. *IFAC Proc Volume* 2009; 42(21): 120–124.
6. Post O, Seppälä J and Koivisto H. The performance of OPC-UA security model at field device level. In: *Proceedings of the 6th international conference on informatics in control, automation and robotics, volume robotics and automation*, Milan, 2–5 July 2009, pp.337–341. Porto: INSTICC Press.
7. Wu KH, Li Y, Chen L, et al. Research of integrity and authentication in OPC UA communication using whirlpool hash function. *Appl Sci* 2015; 5(3): 446–458.
8. Verma D, Jain R and Shrivastava A. Performance analysis of cryptographic algorithms RSA and ECC in wireless sensor networks. *IUP J Telecommun* 2015; 7: 51–65.
9. Sciancalepore S, Piro G, Boggia G, et al. Public key authentication and key agreement in IoT devices with minimal airtime consumption. *IEEE Embed Syst Lett* 2017; 9(1): 1–4.
10. Sciancalepore S. Key management protocol with implicit certificates for IoT systems. In: *Proceedings of the 2015 workshop on IoT challenges in mobile and industrial systems*, Florence, 18 May 2015, pp.37–42. New York: ACM.
11. Park C-S. A secure and efficient ECQV implicit certificate issuance protocol for the Internet of Things applications. *IEEE Sens J* 2017; 17(7): 2215–2223.
12. Ha DA, Nguyen KT and Zao JK. Efficient authentication of resource-constrained IoT devices based on ECQV implicit certificates and datagram transport layer security protocol. In: *Proceedings of the 7th symposium on information and communication technology*, Ho Chi Minh City, Vietnam, 8–9 December 2016, pp.37–42. New York: ACM.
13. Schwarz MH and Borcsok J. A survey on OPC and OPC-UA: about the standard, developments and investigations. In: *Proceedings of the 2013 XXIV international symposium on information, communication and automation technologies (ICAT)*, Sarajevo, 30 October–1 November 2013, pp.1–6. New York: IEEE.
14. Viega J, Chandra P and Messier M. *Network security with OpenSSL*. Sebastopol, CA: O'Reilly & Associates, Inc., 2002.
15. Fan XW and Lu JZ. A lightweight WSN authentication and key agreement scheme. *Comput Eng* 2013; 39(3): 146–151.
16. Yang JJ, Lu JZ and Jiang JH. VANETs communication and efficient threshold anonymous authentication scheme. *Comput Eng* 2015; 41(2): 107–112.

17. NESSIE Consortium. Portfolio of recommended cryptographic primitives, 27 February 2003, http://www.cryptonessie.org/
18. Maitra S, Sarkar S and Gupta SS. Factoring RSA modulus using prime reconstruction from random known bits. In: Bernstein DJ and Lange T (eds) *Progress in cryptology-AFRICACRYPT 2010*. Berlin; Heidelberg: Springer, 2010, pp.82–99.
19. Hafizul Islam SK and Biswas GP. Improved remote login scheme based on ECC. In: *Proceedings of the 2011 IEEE international conference on recent trends in information technology (ICRTIT)*, Chennai, India, 3–5 June 2011, pp.1221–1226. New York: IEEE.
20. Yang XH. *Research on authenticated key agreement protocol based on ECC*. Jining, China: Qufu Normal University, 2015.
21. Liao YP and Hsiao CM. A novel multi-server remote user authentication scheme using self-certified public keys for mobile clients. *Future Gener Comp Sy* 2013; 29(3): 886–900.