code-based authentication protocol for Internet of Things

https://youtu.be/kNRuFmrtkJ8

장경배

한성대학교
HANSUNG UNIVERSITY

CryptoCraft LAB
https://crypto.modoo.at

Paper

The Journal of Supercomputing
https://doi.org/10.1007/s11227-019-03003-4

A privacy-preserving code-based authentication protocol for Internet of Things

Noureddine Chikouche[1] · Pierre-Louis Cayrel[2] · El Hadji Modou Mboup[3] · Brice Odilon Boidje[3]

Abstract
The Internet of Things (IoT) is an upcoming technology that permits to interconnect different devices and machines using heterogeneous networks. One of the most critical issues in IoT is to secure communication between IoT components. The communication between the different IoT components is insecure, which requires the design of a secure authentication protocol and uses hardness cryptographic primitives. In 2017, Wang et al. proposed an improved authentication protocol based on elliptic curve cryptography for IoT. In this paper, we demonstrate that Wang et al.'s protocol is not secure. Additionally, we propose a privacy-preserving authentication protocol using code-based cryptosystem for IoT environments. The code-based cryptography is an important post-quantum cryptography that can resist quantum attacks. It is agreed in design several cryptographic schemes. To assess the proposed protocol, we carry out a security and performance analysis. Informal security analysis and formal security validation show that our protocol achieves different security and privacy requirements and can resist several common attacks, such as desynchronization attacks, quantum attacks, and replay attacks. Moreover, the performance evaluation indicates that our protocol is compatible with capabilities of IoT devices.

Keywords Authentication protocol · Internet of Things · Code-based cryptography · Security · Privacy

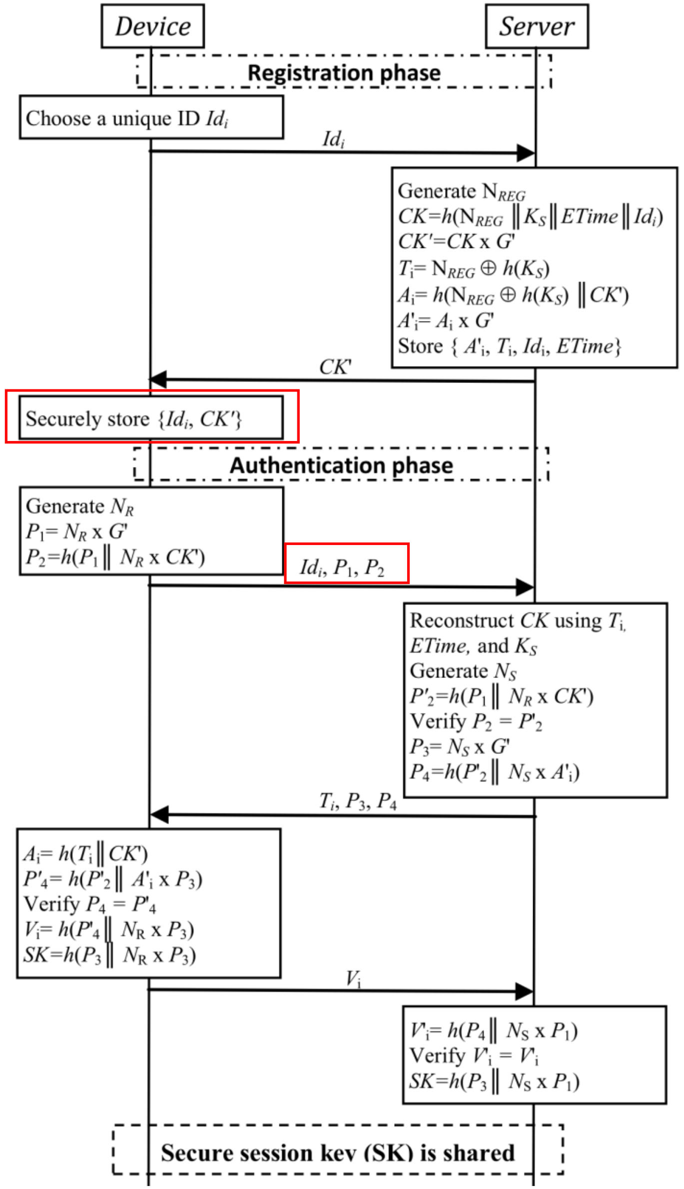
Contribution

1. Wang 의 프로토콜이 취약함을 증명

2. 안전성이 증명된 McEliece의 변형버전을 자신들의 프로토콜에 사용

3. IoT를 위한 상호인증 프로토콜에 코딩이론을 최초적용

Wang et al. Protocol



4.2 Cryptanalysis of Wang et al. scheme

4.2.1 Violation of device anonymity

Device 의 익명성이 보장되지 않는다.

4.2.2 Violation of untraceability

통신 채널에서 메세지를 캡쳐하여 Device 를 추적할 수 있다.

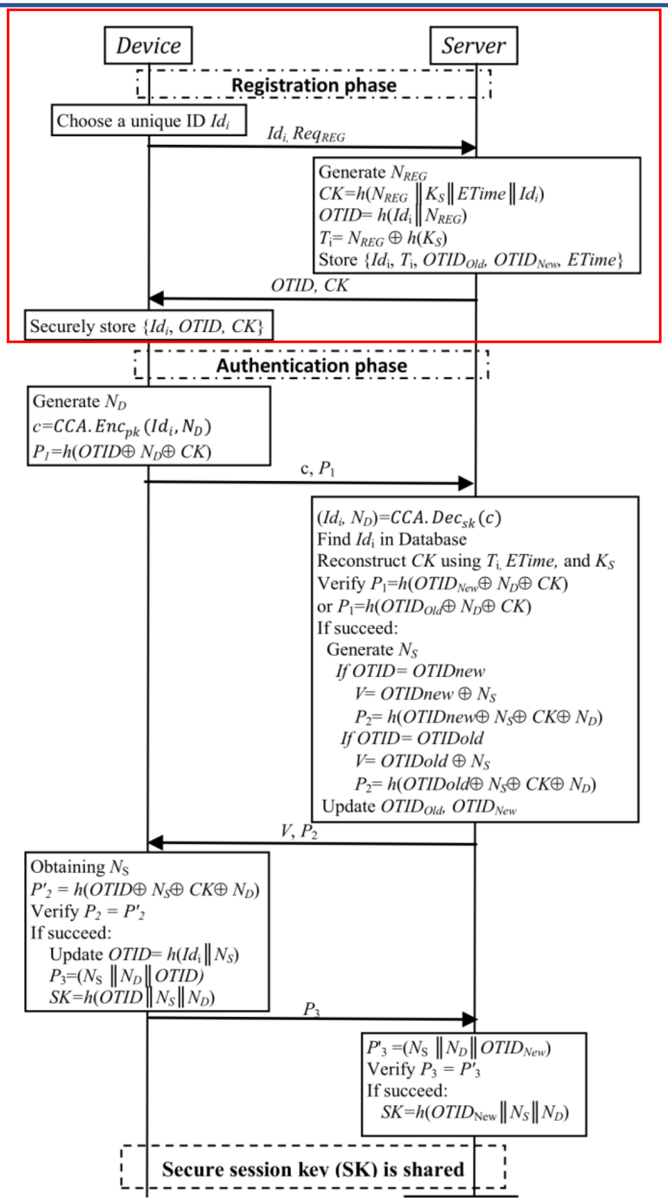
4.2.3 Violation of forward secrecy

Forward Secrecy 가 보장되지 않는다.

- Device에 저장되있는 {Id, CK'} 가 동적이 아니라 정적이며, 아래의 인증단계에서 그대로 사용된다.

Proposed Protocol

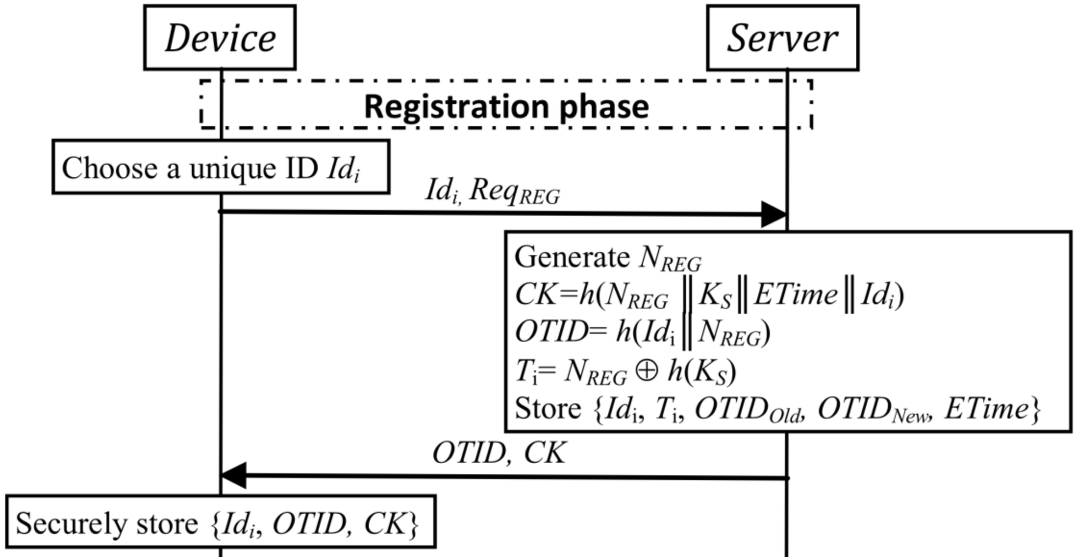
Registration



*

D_i	The IoT device
S	The server
ID_i	Identifier of IoT device
$OTID$	one-time identifier
N_D	Random number generated by D
N_S, N_{REG}	Random numbers generated by S
K_S	Server's secret key
SK	A session key outputted at the end of a scheme
$h(.)$	cryptographic hash function
$\|$	Concatenation of two inputs
CK	Cookie information
$ETime$	Expiration time of the cookie
$KDF(.)$	Key Derivation Function

Registration Phase



Step 1. Server에 등록하고 싶은 Device는 고유의 Id_i 를 선택하여 등록을 요청하는 Req_{REG} 메세지를 서버에 전송

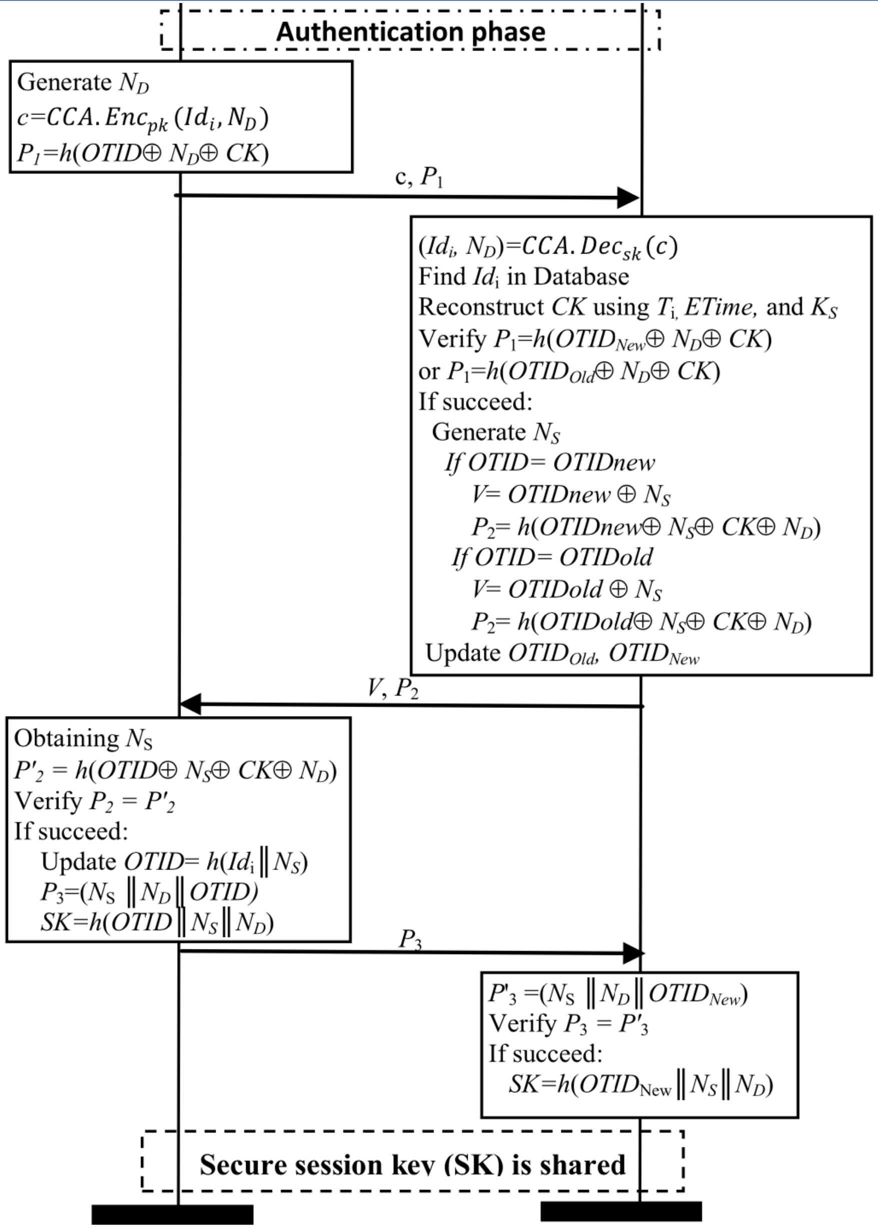
Step 2 . 등록요청 메세지를 받은 서버는 Nonce 값N_{REG} 생성하고, Cookie $CK = h(N_{REG} \| K_S \| ETime \| Id_i)$ 를 계산

One-time identifier $OTID = h(Id_i \| N_{REG})$, 그리고 $Ti = N_{REG} \oplus h(K_S)$ 계산

서버는 $OTID_{old}$, $OTID_{new}$ 를 초기화하고 $\{Id_i, T_i, OTID_{Old}, OTID_{New}, ETime\}$ 를 데이터베이스에 안전하게 저장

Device에 $\{OTID, CK\}$ 를 전송하고, Device는 Id와 함께 안전하게 저장한다. 또한, 서버와 Device 는 공개키 G 를 저장

Authentication Phase



Step1. 통신이 하고싶은 Device 는 Nonce 값 N_D 를 생성하고
공개키를 사용하여 c 를 계산 $c = CCA.Enc_{pk}(id_i, N_D)$

QC-MDPC McEliece

5.1 CCA McEliece based on QC-MDPC

In order to protect IoT, we propose an improved McEliece encryption scheme to be applied in our mutual authentication protocol. It consists in applying a random permutation to the ciphertext before extraction. This approach makes it difficult to locate the error position. We use $QC-MDPC.KeyGen$ (see Sect. 2.2) to generate pair key. The encryption and decryption algorithms of our approach are as follows:

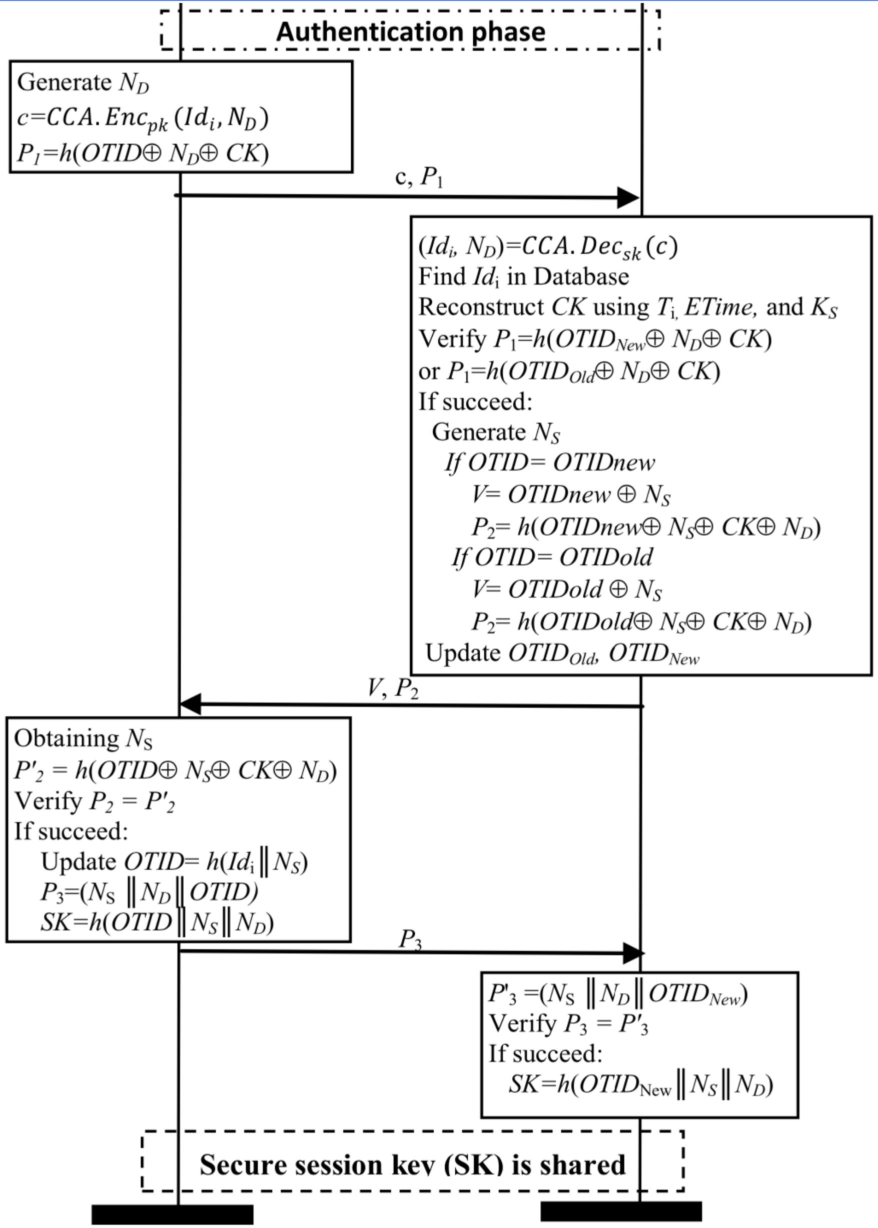
Encryption: $CCA.Enc_{pk}(m, s)$

- **Input:** $\boxed{\text{a message } m \text{ and a random secret word } s}$
- $c_1 := Enc^{qc}_{pk}(s)$,
- $K := KDF(s, \ell_{DEM} + \ell_{MAC})$,
- Parse $K := (K_1 || K_2)$,
- $c_2 := K_1 \oplus m$,
- Set $T := c_2$ and evaluate $v := Ev(K_2, T)$,
- **Output:** $\boxed{c} := (c_1 || c_2 || v)$.

Decryption: $CCA.Dec_{sk}(c)$

- **Input:** a cryptogram $\boxed{c}$
- Parse the ciphertext as $c := (c_1 || c_2 || v)$,
- $s := Dec^{qc}_{sk}(c_1)$,
- if decryption succeeds calculate $K := KDF(s, \ell_{DEM} + \ell_{MAC})$
- else sample a random permutation σ and calculate $K := KDF(\sigma(c_1), \ell_{DEM} + \ell_{MAC})$,
- Parse $K := (K_1 || K_2)$,
- Set $T := c_2$, then calculate $v' := Ev(K_2, T)$,
- if $(v \neq v')$ the check fails and return $\perp$
- Otherwise, calculate $m := K_1 \oplus c_2$,
- **Output:** $\boxed{(m, s)}$.

Authentication Phase



Step1. 통신이 하고싶은 Device 는 Nonce 값 N_D 를 생성하고
공개키를 사용하여 c 를 계산 $c = CCA.Enc_{pk}(id_i, N_D)$
서버에게 받은 $OTID$ 와 CK 그리고 생성한 Nonce 값을 해쉬하여 P_1계산

Step2. 암호문 c 와 P_1 을 받은 서버는 복호화를 통해 Id_i 와 N_D 을 획득

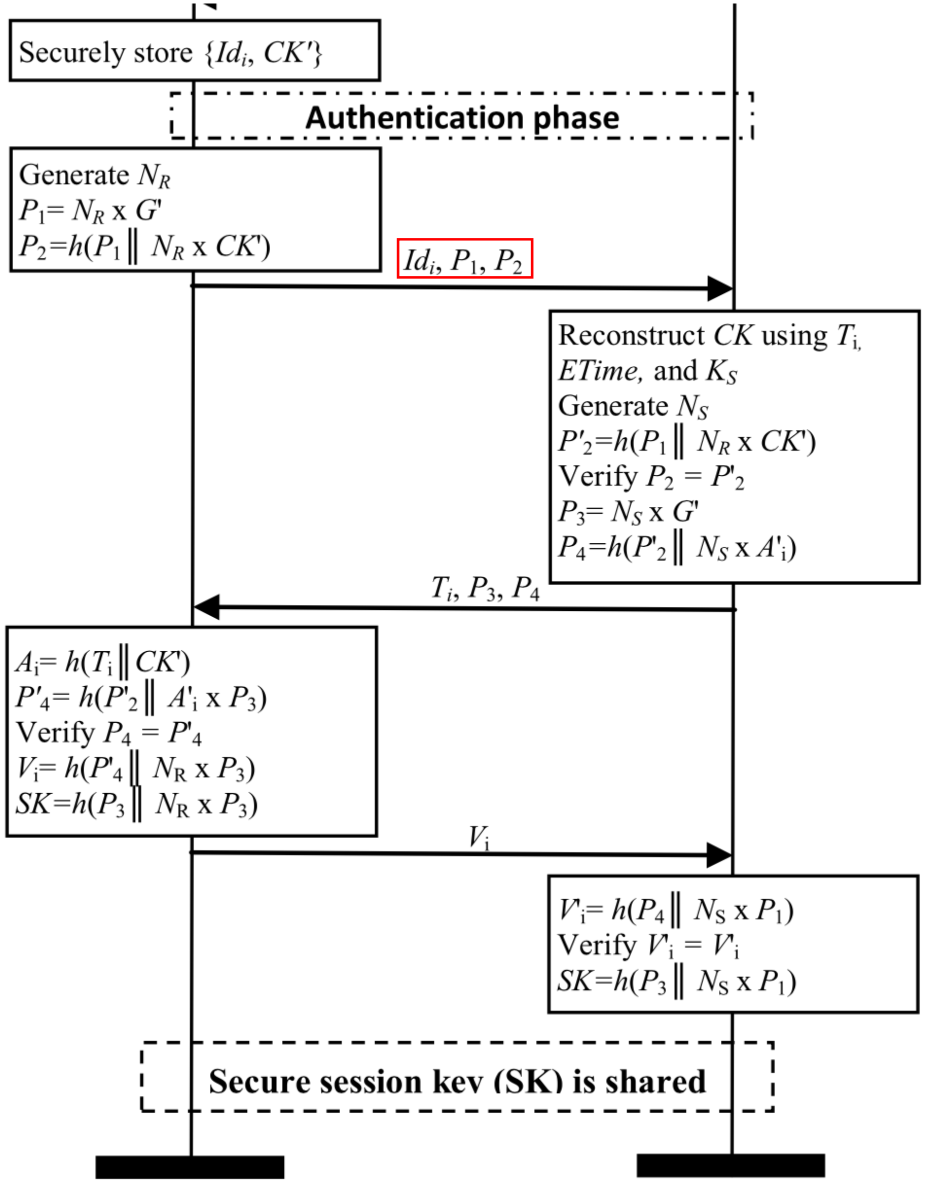
데이터베이스에서 Id_i 를 대조하여 $\{T_i, OTID_{Old}, OTID_{New}, ETime\}$ 로드
로드해온 것을 통해 CK 를 구성하여 P'_1 를 계산

$CK = h(N_{REG} \| K_S \| ETime \| Id_i)$
$P'_1 = h(OTID \oplus N_R \oplus CK \oplus Id_i)$

$P_1 \doteq P'_1$ 라면 검증완료
새로운 Nonc N_S 생성
$V = OTID \oplus N_S$ 그리고 $P_2 = h(OTID \oplus N_S \oplus CK \oplus Id_i)$ 계산

Device 에게 $\{V, P_2\}$ 를 전송하기 전, $OTID = OTID_{New}$ 였다면
$OTID_{old} = OTID_{New}$
$OTID_{New} = h(Id_i \| N_S)$. 로 업데이트 그리고 전송

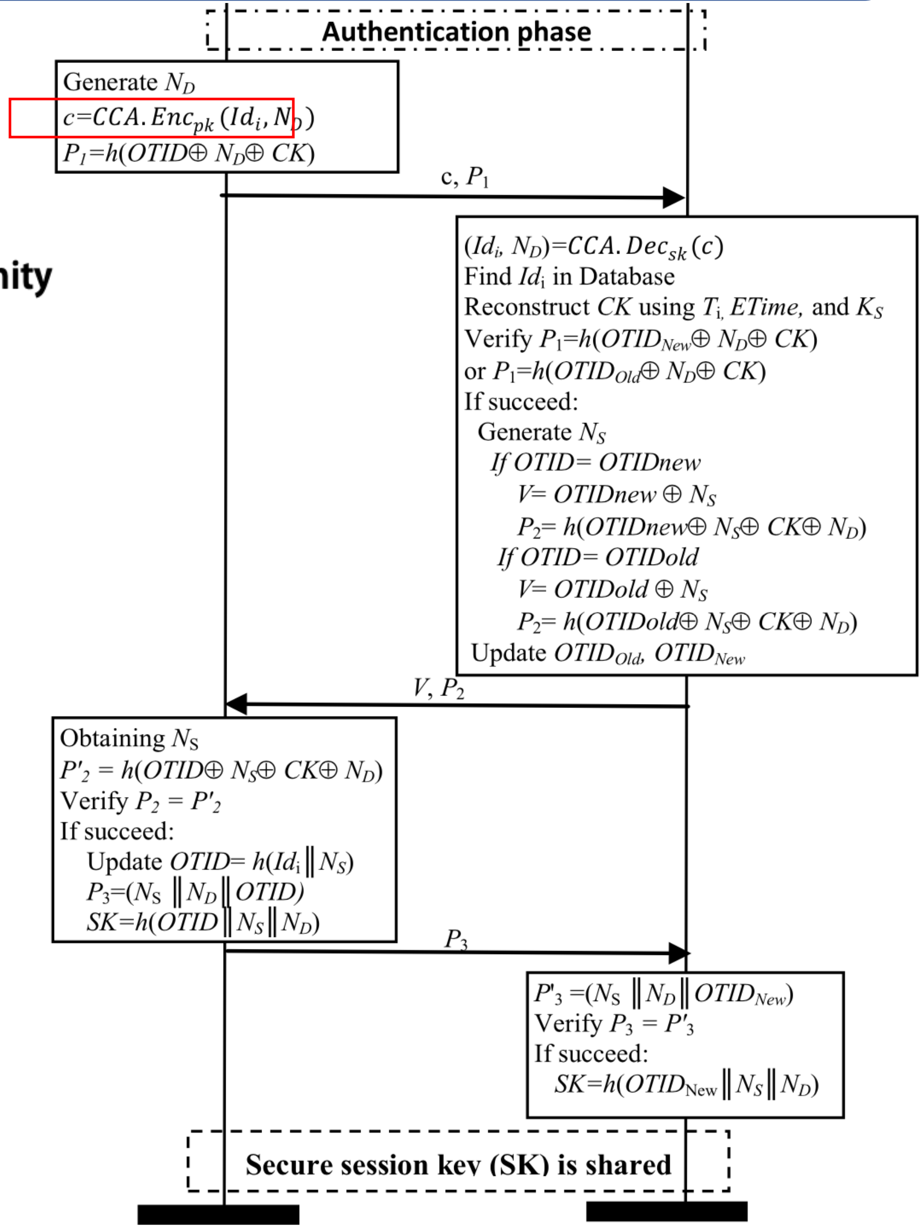
Wang vs Proposed



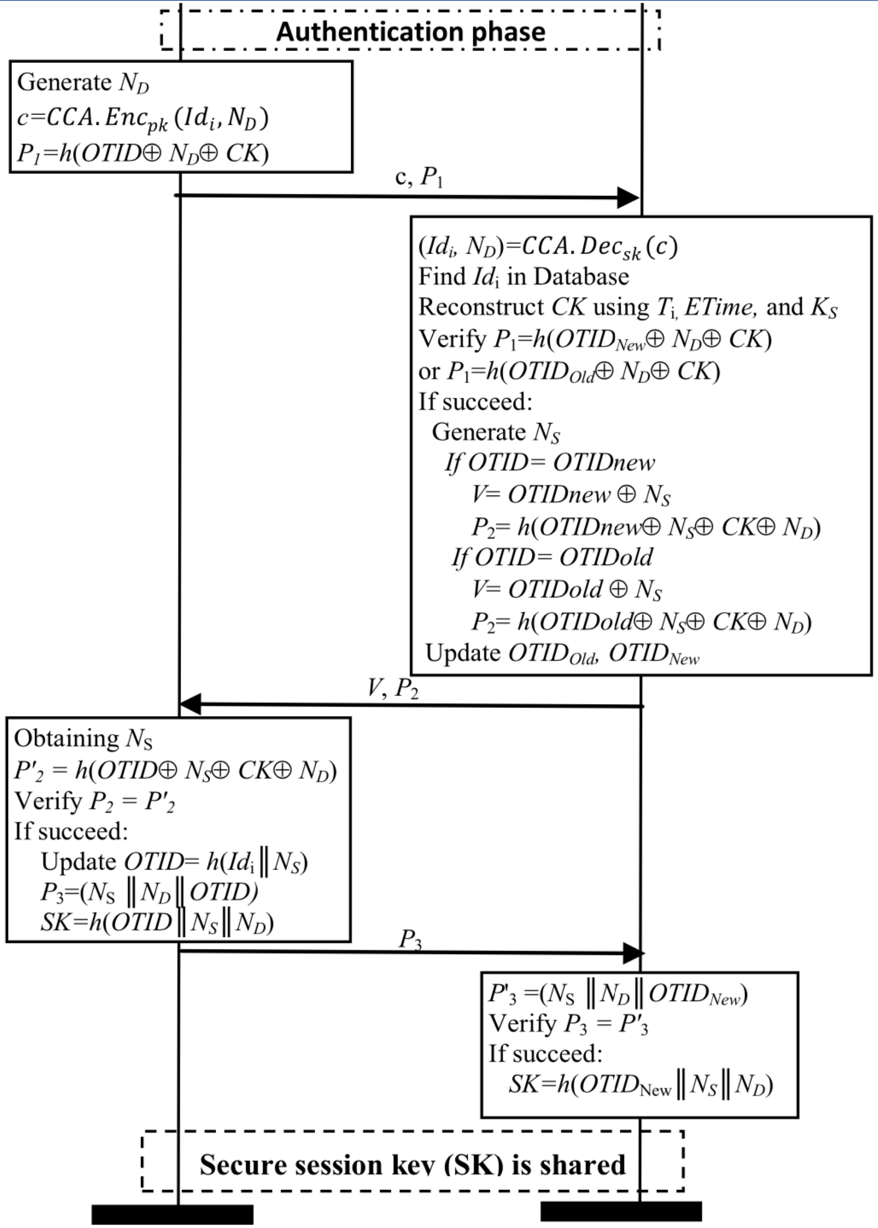
Violation of device anonymity

Violation of untraceability

ECC vs PQC



Performance



Step3. $\{V, P_2\}$ 를 받은 $V \oplus OTID$ 를 계산하여 N_S 획득

$$P'_2 = h(OTID \oplus N_S \oplus CK \oplus Id_i)$$ 를 계산

진짜 서버라 $P_2 \doteq P'_2$

검증이 완료되었다면, Device는 $OTID$ 를 업데이트

$$OTID = h(Id_i \parallel N_S)$$

새롭게 업데이트한 $OTID$, 와 Nonce 값들을 XOR 하여 P_3 계산

$$P_3 = (N_S \oplus N_D \oplus OTID)$$ 서버에 전송

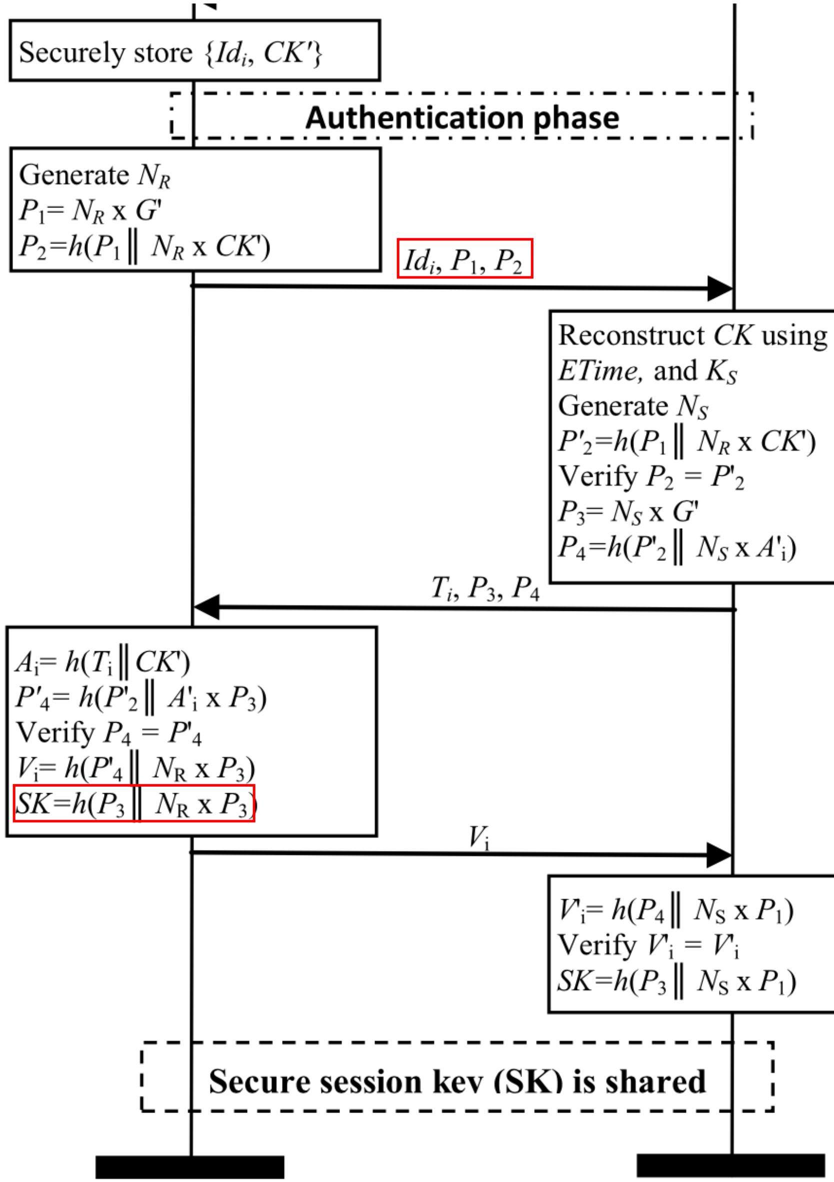
통신을 위한 세션 키 $SK = h(OTID \parallel N_S \parallel N_D)$ 생성

Step4. P_3 를 받은 서버는 검증

$$P'_3 = (N_S \oplus N_D \oplus OTID)$$

$P_3 \doteq P'_3$ 라면 세션 키 생성 $SK = h(OTID_{New} \parallel N_S \parallel N_D)$

Wang vs Proposed



Violation of device anonymity

Violation of untraceability

ECC vs PQC

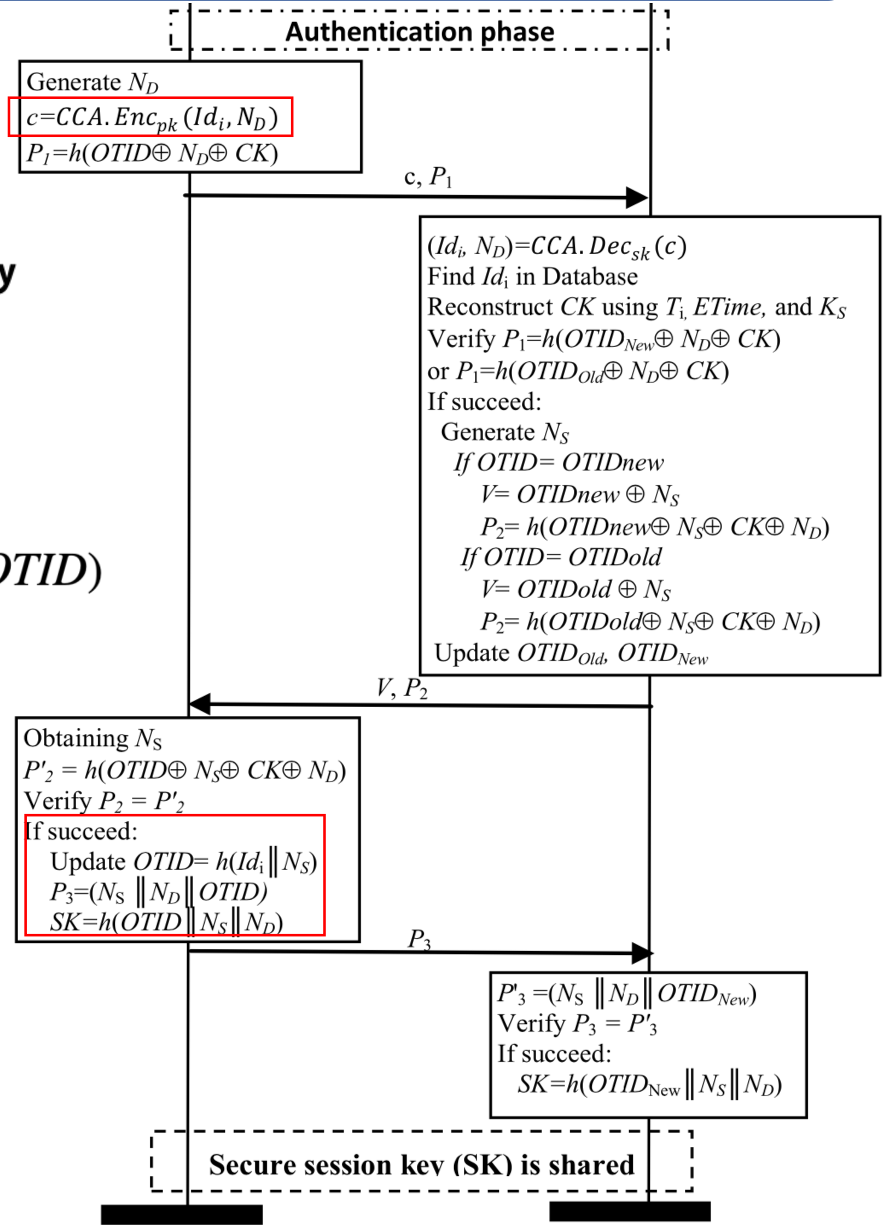
+

Violation of forward secrecy

Device 메모리의 $(Id_i, CK, OTID)$
가 탈취되어도

Update $OTID = h(Id_i \| N_S)$

때문에 이전 세션의
$OTID$ 를 알 수 없음



Security Analysis

different. In our protocol, we resolved this issue by using the old one-time identifier $OTID_{old}$, and thus, the device's authentication is successful.

In addition, Maarof et al. scheme [34] does not resist desynchronization attack, although it used synchronized secret numbers as it updated this one on the device before the server. When the attacker blocks the communication *D-S* after performing the update in the device, the new value of secret data in *D* does not exist in the database of server, neither old nor new secret data. Then, we avoid this weakness by doing the update on the server before the device. In our protocol, we updated the secret data in the server before the device to avoid the desynchronization attack.

6.2 Formal security analysis by using AVISPA tool

6.2.1 Verification tool

In order to validate the safety and the robustness of our proposed protocol, we use the AVISPA tool (Automated Validation of Internet Security Protocols and Applications) [3]. We choose this tool for the following reasons:

- It can detect passive and active attacks, such as replay and MITM attacks.
- It consists of four back ends that based on several approaches of formal validation: SATMC (SAT-based Model Checker), OFMC (On-the-fly Model Checker), CL-ATSE (Constraint-Logic-based Attack Searcher), and TA4SP (Tree Automata based on Automatic Approximations for the Analysis of Security Protocols).
- HLPSL language is the specification language that used by different back ends.
- CL-ATSE and OFMC validate the protocols based on exclusive-or operator.
- AVISPA is widely agreed by several researchers for the formal verification of cryptographic protocols in different domains, such as Internet, mobile, RFID, and WSN.

SPAN (Security Protocol ANimator) tool is a simulator that permits to animate the cryptographic protocols that are specified by HLPSL and validated by AVISPA tool. It also permits to simulate scenarios of attacks.

To validate a security protocol, we have two essential steps. First one, specify the protocol by using HLPSL that include, roles, messages transmitted, intruder capacity, initial assumptions, and the protocol goals. Last one, we carry out the specification code in AVISPA tool that translates into an intermediate format IF. This is the input of different back ends that confirm if the protocol is either safe or has failed. In the last case, the tool shows the attack trace.

6.2.2 Specification of our scheme

AVISPA tool provides the language HLPSL (High-Level Protocol Specification Language) [51] to specify cryptographic protocols. HLPSL is a formal, expressive, and role-based language. The specification of protocol consists of two parts: basic roles and composed roles. Basic roles illustrate the actions of one single agent in

the execution of the protocol. Others instantiate basic roles to model an entire protocol run, a session of the protocol between multiple agents, or the protocol model itself. HLPSL can specify two important properties, the data confidentiality, and the authentication. The Dolev–Yao model [22] is the intruder model that supported by HLPSL.

In our work, we specify the mutual authentication phase of our proposed scheme. Basic roles are shown in Fig. 3. The honest entities are the server *S* and the IoT device *D*. So, we have two basic roles, the server and the device.

Composed roles are shown in Fig. 4. They consist of three parts: session, environment, and goal. The session role describes the initial state of the protocol. The environment role illustrates sessions of the protocol between honest agents. In the end of the specification, we specify the security properties that we want to check them. HLPSL can specify the authentication and the confidentiality requirements.

6.2.3 Verification results

The validation results of the proposed protocol with OFMC and CL-AtSe back ends are shown in Figs. 5 and 6, respectively. These results are found to be SAFE. By using these results, we can conclude that proposed protocol accomplishes the goals of the confidentiality and mutual authentication and ensures resistance to the MITM attack and the replay attack over the Internet of Things environment under the test of AVISPA.

```
role device  ( D,S: agent,
               ID,OTID,  CK: text,
               H : hash_func,
               PK: public_key,
               Snd,Rec:  channel(dy))
    played_by  D
    def=
      local   State   : nat,
              Ns, Nd   : text,
              SK:  symmetric_key
init State  := 0
transition
1. State  = 0 /\ Rec(start)  =|> State'  := 1
/\ Nd'  := new()
/\
Snd({ID.Nd'}_PK.H(xor(xor(OTID,Nd'),CK)))
/\ witness(D,S,device_auth,Nd')
/\ secret({ID},sec_id,  {D,S})
/\ secret({Nd},sec_nd,  {D,S})
2. State  = 1 /\
Rec(xor(OTID,Ns').H(xor(xor(xor(OTID,Ns'),
CK),Nd)))
=|> State'  := 2 /\
request(D,S,server_auth,Ns')
/\  OTID':=H(ID.Ns')
/\ Snd(H(Ns.Nd.OTID'))
/\  SK'  := H(OTID'.Ns.Nd)

end role

role server  (S,D: agent,
               ID,OTIDnew,OTIDold,  CK:
text,
               H : hash_func,
               PK: public_key,
               Snd,Rec:  channel(dy))
    played_by  S
def=
      local   State   : nat,
              Ns, Nd   : text,
                  SK:  symmetric_key
init State  := 0
transition

% Case:  OTID = OTIDnew
1. State  = 0 /\
Rec({ID.Nd'}_PK.H(xor(xor(OTIDnew,Nd'),CK)
))
=|>  State'  := 1 /\ Ns'  := new()
/\
Snd(xor(OTIDnew,Ns').H(xor(xor(xor(OTIDnew
,Ns'),CK),Nd)))
/\ witness(S,D,server_auth,Ns')
/\ secret({Ns},sec_ns,  {S,D})
/\  OTIDold':=OTIDnew  /\
OTIDnew':=H(ID.Ns')

% Case:  OTID = OTIDold
1. State  = 0 /\
Rec({ID.Nd'}_PK.H(xor(xor(OTIDold,Nd'),CK)
))
 =|>  State'  := 1 /\ Ns'  := new()  /\
Snd(xor(OTIDold,Ns').H(xor(xor(xor(OTIDold
,Ns'),CK),Nd)))
/\ witness(S,D,server_auth,Ns')
/\ secret({Ns},sec_ns,  {S,D})
/\  OTIDnew':=H(ID.Ns')

2. State  = 1 /\ Rec(H(Ns.Nd.OTIDnew'))
=|> State'  := 2
/\ request(S,D,device_auth,Nd)
/\  SK'  := H(OTIDnew.Ns.Nd)

end role
```

Fig. 3 The basic roles in HLPSL

Result

Classic McEliece

2.7 Encapsulation

The sender generates a session key K and its ciphertext C as follows:

1. Generate a uniform random vector $e \in \mathbb{F}_2^n$ of weight t.
2. Use the encoding subroutine on e and public key T to compute C_0.
3. Compute $C_1 = \mathsf{H}(2, e)$; see Section 2.9 for H input encodings. Put $C = (C_0, C_1)$.
4. Compute $K = \mathsf{H}(1, e, C)$; see Section 2.9 for H input encodings.
5. Output session key K and ciphertext C.

weight 의 조건이 있는 벡터를 암호화

Protocol

Generate N_D

$c = CCA.Enc_{pk}(Id_i, N_D)$

$P_1 = h(OTID \oplus N_D \oplus CK)$

Device ID, Nonce값 암호화

1. KEM 구조의 Classic McEliece, 암호화에 조건이 따름 → Device ID는 암호화 불가능?
2. Memory 문제
3. ROLLO?

Result

	Code	Security history	Structure	Security level	Performance
Classic McEliece	Goppa	long	KEM	IND-CCA2	low
BIKE	QC-MDPC	short	KEM	IND-CCA	high
NTS-KEM	Goppa	long	KEM	IND-CCA	low
HQC	QC	short	KEM	IND-CCA2	high
RQC	QC	short	KEM	IND-CCA2	high
ROLLO	Rank Metric	short	PKE/KEM	IND-CPA(KEM) IND-CCA2(PKE)	high
LEDAcrypt	QC-LDPC	short	PKE/KEM	IND-CCA2	low

감사합니다