논문 15-40-10-13 The Journal of Korean Institute of Communications and Information Sciences '15-10 Vol.40 No.10
http://dx.doi.org/10.7840/kics.2015.40.10.1958

OneM2M 환경에서 안전한 통신을 위한 카멜레온 해쉬 기반의 상호인증 프로토콜

김 성 수◆, 전 문 석*, 최 도 현°

Chameleon Hash-Based Mutual Authentication Protocol for Secure Communications in OneM2M Environments

Sung-soo Kim◆, Moon-seog Jun*, Do-hyeon Choi°

요 약

사물지능통신(M2M or IoT) 서비스 활성화와 글로벌 업체의 OneM2M 관련 사업에 대한 적극적인 투자 및 가속화는 ICT 시장의 변화를 이끌고 있다. 그러나 다양한 해킹(데이터 노출, 도용, 변조, 삭제 등)의 가능성 때문에 안전한 통신 보안 기술이 중요 요구사항으로 이슈화되고 있다. 본 논문은 M2M 환경의 기존 연구에서 적용된 RSA, DSA 기반의 서명이 아닌 ECC 기반 카멜레온 해쉬(Chameleon Hash) 서명을 적용한다. 성능 분석 결과 효율성은 암·복호화 평균 0.7%, 연산속도는 3%(평균 0.003초) 차이로 비교대상 알고리즘과 동등한 수준으로 우수한 결과를 나타냈고, ECC(Elliptic Curve Cryptography) 기반 카멜레온 해쉬함수 서명의 키 갱신 장점을 이용하여 상호인증과 암호화 구간의 연산 효율성, 확인 가능한 충돌 메시지 특성을 이용하여 통신 구간의 강력한 보안성을 입증하였다.

Key Words : M2M, Chameleon Hash, Authentication, IoT, ECC

ABSTRACT

Things intelligence communication (M2M or IoT) service activation and global company of OneM2M-related business on aggressive investing and has led to the acceleration of change in the ICT market. But a variety of hacking security technology because of the possibility of secure communication (data exposure, theft, modification, deletion, etc.) has been issued as an important requirement. In this paper, we propose a mutual authentication protocol for secure communications chameleon hash based on the M2M environment. The results of performance analysis efficiency is encryption and decryption an average of 0.7%, calculated rate showed good results as compared to the target algorithm, equivalent to a 3%(Average 0.003 seconds) difference, mutual authentication and encryption region by using the key update advantage of ECC(Elliptic Curve Cryptography)based Chameleon hash function is signed of the operational efficiency, using a collision message verifiable properties demonstrated strong security of the communication section.

Ⅰ. 서 론

M2M(Machine-to-machine) 이란 사물과 사물, 사물과 사람 간에 정보가 수집되고 처리되는 지능형 정보 인프라를 의미[1]하며 통신과 ICT 기술을 결합하여 원격지의 사물, 차량, 사람의 상태정보 등 다양한 분

◆ First Author : Soongsil University Department of Computer Science and Engineering, indielazy@ssu.ac.kr, 정회원
° Corresponding Author : Soongsil University Department of Computer Science and Engineering, cdhgod0@ssu.ac.kr, 정회원
* Soongsil University Department of Computer Science and Engineering, mjun@ssu.ac.kr, 종신회원
논문번호 : KICS2015-08-252, Received August 13, 2015; Revised September 16, 2015; Accepted September 24, 2015

야에 걸쳐 활용되고 있다[2]. 센서 등으로부터 수집한 데이터를 가공하고 분배하는 점에서는 과거의 서비스와 유사하며, 디바이스들이 인간의 개입 없이 스스로 통신하며 정보를 교환한다는 점에서는 과거의 서비스에 비해 진일보한 측면이 있다[3].

14년 기준 인포네틱스(Infonetics) 시장조사에 따르면 M2M기술을 도입하는 기업과 관련 서비스가 증가하는 추세이고 13년 기준 M2M 서비스 매출 규모는 160억 달러, 13년에서 18년까지 연평균 성장률(CAGR)은 18% 수준으로 예상하고 있다[4]. 또한, IDC에 따르면 M2M 솔루션 시장은 13년에서 20년까지 연평균 성장률 17.5%로 20년에 7조원에 가까이 성장할 것으로 예상하고 있다[5].

M2M 서비스 확산 및 솔루션 시장의 성장과 함께 M2M 관련 사이버보안 강화가 새로운 과제로 부상하고 있다. 특히 M2M 표준 디바이스들은 현재 대체적으로 기능이 단순하고 보안성도 취약한 경우가 많아 해킹에 대한 뚜렷한 대책이 없는 상황이다[6].

따라서 본 논문은 현재 활발히 진행 중인 M2M 서비스의 안전한 통신을 위한 카멜레온 해쉬 기반의 상호인증 프로토콜을 제안한다. 2장은 관련연구, 3장은 제안 프로토콜, 4장은 성능 및 안전성 분석, 5장 결론으로 마친다.

Ⅱ. 관련 연구

본 관련연구에서는 OneM2M 아키텍처에서 정의하는 기능적 요구사항 중 보안 요구사항에 대하여 다룬다. 또한 제안하는 프로토콜의 안정성과 효율적인 통신을 위한 카멜레온 해쉬 기법에 대하여 설명한다.

2.1 OneM2M 아키텍처와 보안 요구사항

그림 1은 WG2에서 정의한 OneM2M 시스템 아키텍처를 나타내고, 4개의 대표적인 노드들로 구성된다[7].

- ADN(Application Dedicated Node) : M2M 어플리케이션을 포함, 서비스 로직만 포함(제한적)
- ASN(Application Service Node) : 어플리케이션과 공통 서비스 기능 포함
- MN(Middle Node) : 디바이스 노드와 네트워크를 연결해주는 게이트웨이
- IN(Infrastructure Node) : M2M 서비스 제공자

각 통신 노드는 최소 하나 이상의 AE(Application Entity)와 CSE(Common Service Entity)를 포함한다.

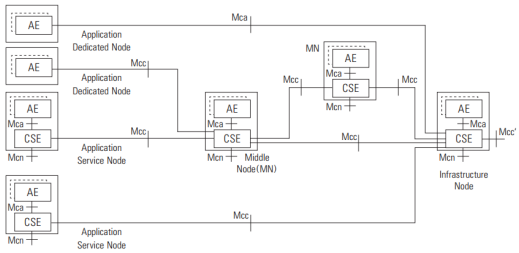

그림 1. OneM2M 아키텍처
Fig. 1. OneM2M Architecture

CSE는 M2M 노드 중 실행되는 논리적인 엔티티로 CSFs(Common Service Functions)를 공통 M2M 서비스 기능으로 포함한다.

OneM2M 아키텍처(WG2)에서 정의한 각 노드들은 등록, 연결에 대한 보안 및 프라이버시 제공, 관리 기능 등 구성요소에 대한 CSF를 정의하고 있다. CSF 표준에서 정의하는 M2M 디바이스의 구성 및 노드 위치, AE와 CSE의 유무에 따라 통신의 종류는 표 1과 같이 구분한다.

통신과정의 노드 사이는 유선통신 및 무선 통신(Zigbee, Bluetooth, WLAN) 또는 근거리 무선 통신 기술(WiFi 등의 D2D 기술), 셀룰러(WCDMA/LTE) 등의 다양한 무선 통신 기술을 사용한다.

M2M 서비스 계층의 기능적 요구사항은 ITU-T와 ETSI가 M2M 서비스와 어플리케이션 계층 등 각 표준화 그룹마다 역할을 구분하여 표준화를 진행 중에 있으며[8], 표 2는 ITU-T의 서비스 계층 보안 요구사항을 나타낸다[9].

보안 요구사항 항목은 인가되지 않은 디바이스 및 어플리케이션 인증, 신원과 위치정보 같은 개인정보의 익명 보호, 데이터의 기밀성, 무결성 보장, 다수 사용자에 대한 서비스 시나리오의 보안성을 제공해야 한다. 현재 M2M 관련 보안 표준은 M2M 관련 산업관계자들이 적합한 보안 메커니즘과 프라이버시 보호를 위해 각 표준화 그룹 별 표준화를 진행 중에 있다.

2.2 카멜레온 해쉬

일반적으로 보안프로토콜에서 해쉬함수는 메시지의 오류나 변조를 탐지할 수 있는 무결성, 전자 서명

표 1. OneM2M 통신의 예
Table 1. OneM2M Communication Example

Case1	ASN - MN(Middle) - IN
Case2	ADN - MN(Middle) - IN
Case3	ASN - Mcc - IN
Case4	ADN - Mca - IN

표 2. ITU-T 보안 요구사항
Table 2. ITU-T Security Requirement

Authentication	Service layer is required to provide authentication mechanisms for applications and devices and prevent unauthorized use of the devices.
Privacy	Service layer is required to support privacy protection capabilities, such as anonymity of identity and location, according to regulation and laws.
Confidentiality	Service layer is required to support data transfer confidentiality.
Integrity	Service layer is required to support data integrity protection.
Support of security for service scenarios involving multiple actors	Service layer is required to support security capabilities, such as supporting user access control of protected data, for M2M service scenarios involving multiple actors inside a single administrative domain and across different administrative domains (e.g., countries, operators).

에 활용된다[10]. 카멜레온 해쉬 함수 서명은 어떤 메시지의 서명이 유효한 것인지 검증하는 용도로 사용되며 각 다른 서명에 대해서는 증명할 수 없다[11]. 카멜레온 해쉬함수는 수신자가 충돌을 찾을 수 있는 특징이 있기 때문에 서명 검증 절차를 거친 메시지와 다른 메시지를 찾을 수 있는 트랩도어 함수이다. 관련연구로는 X. Chen의 Chameleon hashing without key exposure, F. Zhang의 ID-Based chameleon hashes from bilinear pairings, G. Ateniese의 On the key exposure problem in chameleon hashes 등이 있다[12-14]. 카멜레온 해쉬함수의 보안 요구사항은 표 3과 같다[15].

카멜레온 해쉬함수는 충돌 저항성, 해쉬값에 대한 역원, 충돌 검증에 대한 숨김, 키 노출에 영향을 받지 않는 4가지 보안 요구사항을 충족한다. 각 파라미터는 공개키 쌍(*SK*, *PK*), 라벨 L, 메시지 m, 보조 랜덤 파라미터 r을 나타낸다. 카멜레온 해쉬 함수의 동작과정은 다음과 같다. 원시근 g, 큰 소수 p를 생성하고 비밀값 x를 이용하여 공개키 y를 계산한다[16].

$$y = g^x \bmod p \quad (\text{단, } x \in Z_p^*) \tag{1}$$

초기 임의의 랜덤값 r_0값과 랜덤 메시지 m_0을 변수로 하고, 계산한 공개키 y를 이용하여 카멜레온 해

표 3. 카멜레온 해쉬 보안 요구사항
Table 3. Chameleon Hash Security Requirement

Collision-resistance	There is no efficient algorithm that given only PK, L, m and r, (but not the secret key SK) can find a second pair m, r
Semantic Security	The chameleon hash value C does not reveal anything about the possible message m that was hashed
Message Hiding	Assume the recipient has computed a collision using the universal forgery algorithm. By showing the second pair(m', r') without the need to open the original message it may correspond to invalid request.
Key Exposure Freeness	If a recipient with public key PK has never computed a collision under label L, then given C=Hash(PK, L, m, r) there is no efficient algorithm that can find a collision (a second pair m, r mapping to the same digest C).

쉬값을 계산한다.

$$CHAM-HASH(m,r) = g^m y^r \bmod p \tag{2}$$

이후 충돌 메시지 m'은 새로운 랜덤값 r'과 계산되는 기존의 충돌 비밀값 $x \in Z_p^*$를 알고 있는 노드에서만 계산이 가능하다.

$$\begin{aligned} m + xr &= m' + xr' \\ m' &= m + x(r - r') \end{aligned} \tag{3}$$

새로운 (m', r')쌍에 대한 해쉬값은 첫 임의의 랜덤 값(m, r)쌍에서 $m \neq m'$일 때, 두 쌍의 해쉬값은 동일한 값임을 확인 가능하다.

$$\begin{aligned} &CHAM-HASH(m', r') \\ &\quad = g^{m'} y^{r'} \bmod p \\ &\quad = g^{m + x(r - r')} g^{xr'} \bmod p \\ &\quad = g^m g^{xr} \bmod p \\ &\quad = CHAM-HASH(m, r) \end{aligned} \tag{4}$$

위와 같이 카멜레온 해쉬함수 기법은 기존 RSA나 DSA와 같은 전통적인 서명기법을 사용하기 때문에 기존 보안 프로토콜에 적용하기 쉬운 장점을 가진다.

Ⅲ. 제안 프로토콜

본 논문에서 제안하는 프로토콜의 M2M 장치 요약어는 표 4와 같고, 프로토콜에서 생성하는 파라미터의 설명은 표 5, 6과 같다.

표 4. 디바이스 요약어
Table 4. Device Abbreviation

Parameter	Description
IN	서비스 제공자
MN	게이트웨이(중간 노드)
ASN, ADN	M2M 디바이스

표 5. 타원곡선 파라미터
Table 5. Elliptic Curve Parameter

Parameter	Description
K, Q	선택된 타원곡선상의 점
$PuKey$	Public Key
$PriKey$	Private Key
MN_C1	MN 인증 파라미터 : C1, C2
MN_C2	
IN_C1	IN 인증 파라미터 : C1, C2
IN_C2	
$S\_Session$	MN과 IN의 보안세션
M	디바이스 식별 정보
Ts	타임스탬프

표 6. 카멜레온 해쉬 파라미터
Table 6. Chameleon Hash Parameter

Parameter	Description
$g,\ p$	큰 소수
IN_x	IN 비밀값
$IN\_CEL,\ IN\_r1$	IN 해쉬 충돌 요소값, 충돌값
ASN_x	ASN 비밀값
ASN_m_i	ASN 메시지
ASN_r_i	ASN 해쉬 충돌값
ASN_y_i	ASN 공개키
ASN_Sign	ASN 카멜레온 해쉬서명
ASN_ID	ASN 식별 정보
ASN_SKEY	ASN의 세션키
MN_x	MN 비밀값
MN_m_i	MN 메시지
MN_r_i	MN 해쉬 충돌값
MN_y_i	MN 공개키
MN_Sign	MN 카멜레온 해쉬서명
MN_ID	MN 식별 정보
MN_SKEY	MN의 세션키
$CKEY$	세션키 생성 비밀값
ENC	암호화용 대칭키

3.1 등록 및 상호인증 과정

그림 2는 등록 및 상호인증 과정을 나타낸다. 본 절은 IN과 MN의 상호인증 이후 보안세션($S\_Session$)을 설정하고, ASN, ADN 노드와 MN 사이에 카멜레온 해쉬의 충돌 값의 특징을 이용하여 동일한 카멜레온 해쉬값을 생성하는 계산을 통해 상호인증을 수행한다.

MN에서 IN으로 인증요청 수행하는데, MN 노드는 종단의 ASN이나 ADN 노드의 식별 정보를 받아 IN으로 전송하는 역할을 하고, 각 노드는 상호인증 수행을 위한 초기 키 교환 과정을 수행한다.

타원곡선상의 점 K, Q를 선택한다.

$$K+Q=(x_1,y_1)+(x_2,y_2)=(x_3,y_3) \tag{5}$$

본 논문에서 사용한 타원곡선은 체의 표수 2와 3이 아닌 $y^2=x^3+ax+b$ 을 사용한다. 곡선상의 점 $K=(x_1,y_1)$, $Q=(x_2,y_2)$의 덧셈연산은 두 점 K, Q를 지나는 직선과 타원곡선과 만나는 제 3의 교점 (x,y)을 x축에 관해 대칭한 점$(x,-y)$을 $K+Q=(x_3,y_3)$ 로 정의하고, 식 (6)과 같이 MN과 IN의 서명에 사용할 개인키와 공개키를 생성하고 교

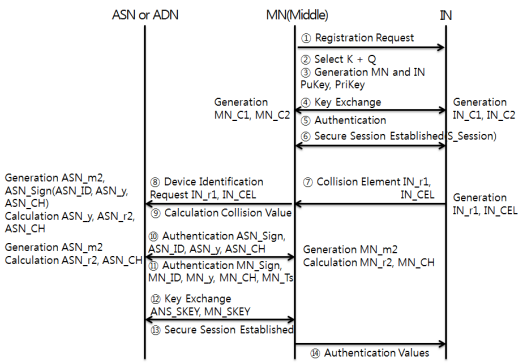

그림 2. ASN, ADN 등록 및 인증과정
Fig. 2. ASN, ADN Registration and Authentication Process

환한다.

$$PuKey = (x_3, y_3)\,PriKey$$
$$PriKey = 160bit\,Integer\,Nmber \quad (6)$$

$PriKey$(정수)와 선택된 타원곡선상의 한 점 $K+Q$를 이용하여 $C1$을 계산하여 공개키로 공개하고, 각 디바이스의 식별정보(타원곡선상의 점)와 암호화에 사용할 랜덤키를 선정하여 $C2$를 계산한다. 각 노드에서 생성되는 $C1$, $C2$ 연산과정은 동일하며 다음과 같다.

$$C1 = r(x_3, y_3)$$
$$C2 = (h(M\|Ts) + PriKey \times IN\_C1) \quad (r-1) \bmod Q \quad (7)$$

MN과 IN사이 상호인증 이후 보안세션을 설립한다. 모든 통신은 MN과 IN사이 상호인증 수행이 선행되어야 하고, 선행인증을 완료한 후 다음 단계를 수행한다.

MN과 ASN, ADN의 상호인증을 위한 충돌 요소값과 충돌값을 IN에서 계산한다. IN은 초기 랜덤값 IN_r0과 메시지 IN_m0, IN_m1을 생성하고, 선택한 큰 소수 g, p중 카멜레온 해쉬함수의 충돌값을 계산하기 위한 비밀값 IN_x을 선택한다.

$$IN\_x \in Z_p^* \quad (8)$$

초기 랜덤값 IN_r0과 메시지 IN_m0, IN_m1을 이용하여 충돌 요소값과 충돌값을 IN_CEL과 IN_r1을 다음 식 (9)와 같이 계산한다.

$$IN\_CEL = IN\_x \times IN\_m1$$
$$IN\_r1 = IN\_r0 + (IN\_x \times IN\_m0 \quad - IN\_CEL) \quad (9)$$

IN이 생성한 IN_CEL과 IN_r1을 MN을 통해 ADN과 ASN에 전송하고, 이를 수신한 ASN, ADN은 인증을 수행하기 위해 다음과 같이 계산한다.

$$ASN\_x \in Z_p^*$$
$$ASN\_m2 = h(ASN\_ID\|ASN\_Ts) \quad (10)$$

ASN은 IN_CEL과 IN_r1 값을 저장하고, ASN 노드는 충돌값을 계산하기 위한 비밀값 ASN_x을 선택하여, 메시지 ASN_m2를 생성한다.

큰 소수 g, p와 선택한 비밀값 ASN_x과 메시지 ASN_m2, 서비스 제공자가 생성한 IN_CEL과 IN_r1을 이용하여 ASN의 ASN_r2 충돌값과 공개키 ASN_y, 카멜레온 해쉬값 ASN_CH을 계산한다.

$$ASN\_r2 = IN\_r1 + (IN\_CEL - ASN\_x \quad \times ASN\_m2) \quad (11)$$

$$ASN\_y = g^{ASN\_x} \bmod p$$
$$ASN\_CH = g^{ASN\_r2} y^{ASN\_m2} \bmod p \quad (12)$$

ASN 노드의 개인키로 ASN_ID, ASN_y, ASN_CH, ASN_Ts에 대해 서명함으로서 전송과정에 데이터 변조나 위조의 위협에 대해 검증하고, 보안세션에 사용할 공개키 ASN_y를 MN으로 전송한다. 또한 정상적인 ASN 노드에서 생성한 동일한 카멜레온 해쉬 값임을 증명하기 위해 ASN_CH와 ASN의 식별값 ASN_ID를 전송한다.

$$ASN\_Sign(ASN\_ID\|ASN\_y\|ASN\_CH \quad \|ASN\_Ts) \quad (13)$$

MN는 ASN, ADN 노드로부터 받은 각각의 값을 저장하고, MN_x를 선택한다. 또한 MN은 충돌값을 계산하기 위한 메시지 MN_m2를 생성한다.

$$MN\_x \in Z_p^*$$
$$MN\_m2 = h(MN\_ID\|MN\_Ts) \quad (14)$$

IN으로부터 수신한 IN_CEL과 IN_r1값을 이용하여 MN의 충돌값 MN_r2, 공개키 MN_y, 카멜레온 해쉬 MN_CH를 계산한다.

$$MN\_r2 = IN\_r1 + (IN\_CEL - MN\_x \quad \times MN\_m2) \quad (15)$$

$$MN\_y = g^{MN\_x} \bmod p$$
$$MN\_CH = g^{MN\_r2} y^{MN\_m2} \bmod p \quad (16)$$

IN에서 생성한 IN_CEL과 IN_r1 값은 정상적인 노드에서만 충돌값을 생성할 수 있으므로, MN은 ASN, ADN 노드로부터 수신한 ASN_CH 값과

MN_CH 값이 서로 동일한 값인지를 식 (17)과 같이 비교한다. 또한 수신한 공개키 ASN_y를 이용하여 ASN_Sign을 검증한다.

$$\begin{aligned} &ASN\_CH \\ &= g^{ASN\_r2} y^{ASN\_m2} \bmod p \\ &= g^{IN\_r1+(IN\_x \times IN\_m1 - ASN\_x \times ASN\_m2)} \\ &\qquad g^{ASN\_x \times ASN\_m2} \bmod p \\ &= g^{MN\_r2} g^{MN\_x \times MN\_m2} \bmod p \\ &= MN\_CH \end{aligned} \tag{17}$$

$$Verify\, ASN\_Sign \tag{18}$$

카멜레온 해쉬값이 동일하고, ASN의 서명에 대한 검증을 수행하면, 정상적인 ASN 노드로부터 생성된 값임을 인증한다. 인증에 대한 MN의 응답으로 ASN_Sign, MN_ID, MN_CH, MN_y, MN_Ts를 MN의 개인키로 서명한 값 MN_Sign과 MN_ID, MN_CH, MN_y, MN_Ts를 ASN 노드로 전송한다.

$$\begin{aligned} MN\_Sign(&ASN\_Sign \| MN\_ID \| MN\_CH \\ &\| MN\_y \| MN\_Ts) \end{aligned} \tag{19}$$

수신한 값들을 저장하고 MN_Sign에 대해 MN_y 값으로 서명을 검증하고, MN_CH와 ASN_CH를 비교함으로써 ASN 노드에서의 MN을 인증함으로써 상호인증을 수행한다.

상호인증 다음 단계로 키교환을 수행한다. ASN 노드와 MN 간의 키교환을 수행함으로써 보안세션을 설정한다. 키 교환은 인증 과정에서 ASN 노드와 MN의 송수신 한 공개키와 카멜레온 해쉬의 비밀값, IN의 충돌 요소값 IN_CEL를 이용하여 세션키를 다음과 같은 식으로 계산하여 생성한다.

$$CKEY = IN\_CEL \bmod p \tag{20}$$

$$\begin{aligned} &ASN\_SKEY \\ &= (g^{MN\_x} \bmod p)^{ASN\_x (IN\_CEL \bmod p)} \bmod p \\ &MN\_SKEY \\ &= (g^{ASN\_x} \bmod p)^{MN\_x (IN\_CEL \bmod p)} \bmod p \end{aligned} \tag{21}$$

$$ASN\_SKEY = MN\_SKEY \tag{22}$$

세션키의 생성으로 키교환 과정을 완료하고, 보안 세션을 설정한다. MN는 ASN, ADN 노드와 상호인증 과정에서 사용한 인증 파라미터 값들을 IN에게 전송한다. 세션 및 키폐지 과정을 위해 파라미터 값이 필요하며 상호인증과 키교환 과정을 수행한 모든 ASN, ADN 노드와 MN의 상태확인을 위해 전송한다.

3.2 키 갱신 및 암호화 과정

그림 3은 키 갱신 및 암호화 과정을 나타낸다.

키 갱신 과정은 통신 중에 나타날 수 있는 재인증 요청, 연결 시간 만료 등 기존 보안 프로토콜에서 성능의 오버헤드가 큰 영역이다. 새로운 비밀값을 선택함으로써 동일한 카멜레온 해쉬값을 계산할 수 있다. 동일한 카멜레온 해쉬 값으로 재인증과 세션키 갱신이 가능하고, 암호화키 또한 매번 갱신할 수 있다.

IN은 MN에게 ASN과 ADN의 키 갱신 수행 요청을 전송한다. 요청을 수신한 ASN, ADN은 새로운 충돌값을 생성하기 위한 비밀값 x_i를 선택한다.

$$ASN\_x_i \in Z_p^* \tag{23}$$

x_i에 따른 공개키 y_i와 충돌값 r_i, 메시지 m_i를 계산하여 기존의 상호인증에 사용한 카멜레온 CH_i와 같은 값임을 확인 할 수 있다.

$$\begin{aligned} &ASN\_y_i = g^{ASN\_x_i} \bmod p \\ &ASN\_m_i = h(ASN\_ID \| ASN\_Ts) \\ &ASN\_r_i \\ &\quad = ASN\_r_{i-1} + (ASN\_x_{i-1} \times ASN\_m_{i-1} \\ &\qquad - ASN\_x_i \times ASN\_m_i) \end{aligned} \tag{24}$$

$$ASN\_CH_{i-1} = ASN\_CH_i = MN\_CH_i \tag{25}$$

또한 MN에서도 키 갱신을 수행하여 양방향 새로운 키 갱신에 따른 새로운 보안세션의 설립이 가능하다.

식 (26)와 같이 ASN, ADN과 MN간의 메시지 암호화 전송을 위한 대칭키 ENC를 계산하고, IN으로 키 갱신에 관한 정보들을 전송한다.

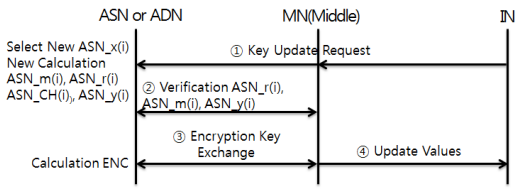

그림 3. 키 갱신 및 암호화 과정
Fig. 3. Key Update and Encryption Process

$$ENC = g^{ASN\_SKEY} \bmod p = g^{MN\_SKEY} \bmod p \quad (26)$$

상호인증에서부터 키 갱신 과정의 파라미터 값을 수신 받아 ASN, ADN과 MN간의 보안 세션 및 메시지 암호화 전송뿐만 아니라, MN과 IN, IN과 ASN, ADN간의 메시지에 대한 암호화 전송이 가능하다.

3.3 세션 및 키 폐지 과정

상호인증을 통해 연결된 보안세션을 해제 하거나 폐지하는 과정은 그림 4와 같다.

MN에서 IN으로 해당 ID에 대해 세션만료 요청을 전송한다.

상호인증 과정에서 저장한 ID 값을 비교하여 연결 설정이 되어있는 단일 ASN, ADN 노드 또는 MN 전체 노드를 폐지할 것인지를 확인하고, 다음과 같이 ENC 키로 폐기 메시지에 단일 노드 ASN_ID나 전체 노드 MN_ID의 파라미터 값을 암호화한다.

$$E_{ENC}(ASN\_ID \| Ts \| END) \quad (27)$$

세션만료 요청에 대한 응답으로 폐기 암호문을 MN으로 전송하여 MN 또는 ASN, ADN으로 전송한다. 각 ASN, ADN 노드는 암호화된 메시지를 복호화하여 자신의 ID와 비교하여 동일하면 세션을 끊고, MN 또한 해당 노드와 세션을 만료한다.

$$D_{ENC}(ASN\_ID \| Ts \| END) \quad (28)$$

세션만료와 사용한 세션키에 대한 폐기 성공 메시지를 IN에게 전송한다.

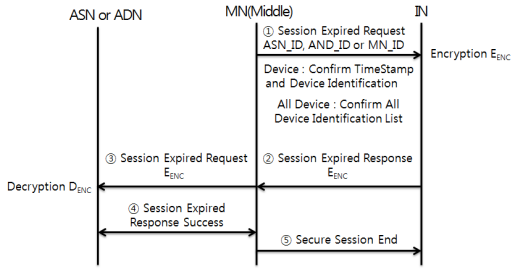

그림 4. 세션 및 키 폐지 과정
Fig. 4. Session and Key Expired Process

Ⅳ. 성능 평가

본 논문에서 제안한 프로토콜은 충돌 요소값으로 충돌 메시지를 생성하여, 동일한 카멜레온 해쉬값을 계산함으로써 양방향 상호인증을 수행하였다.

M2M 환경으로 PKI와 같은 과중한 기술을 요구하지 않고, 초기 타원곡선의 덧셈연산으로만 선행인증을 수행하여 네트워크의 과도한 트래픽 증가 및 디바이스의 오버헤드를 감소시키는 장점이 있다. 또한 트랩도어 기반의 카멜레온 해쉬 보안요구 사항을 만족하고, 정상적인 디바이스에서 계산한 충돌값을 카멜레온 해쉬값으로 확인하여 ASN, ADN과 MN 또는 IN간의 양방향 상호인증 수행을 제공한다.

4.1 알고리즘 검증

표 7은 타원곡선 알고리즘과 카멜레온 해쉬함수에서 연산 파라미터인 $C1$, $C2$의 암호화와 복호화를 나타낸다.

· 타원곡선 암호화와 복호화

타원곡선 공개키 암호에서는 공개키, 평문, 암호문 $C1$, $C2$ 모두 타원곡선 위의 점들이다. 키교환 과정에서 공개된 K, a, b를 이용하여 각 선택된 랜덤값 r과 선택된 점을 이용하여 공개키 $C1$을 생성하고, r값과 $C1$을 사용하여 $C2$를 생성한다. 수신자는 전송받은 $C2$의 역원을 구하여 $C2$를 더하는 $M = C2 - (d \times C1)$ 수식으로 복호화된 메시지 M을 얻어낼 수 있다. 유한체 위에서 정의된 타원곡선 위에 있는 점들 간의 덧셈은 실수의 경우와 동일하게 수행되어 지기 때문에 성능의 효율성이 좋은 장점을 가진다.

표 7. 타원곡선 파라미터 검증
Table 7. Elliptic Curve Parameter Verification

Elliptic Curve
$C1 = r(x_3, y_3)$ $C2 = (h(M \| Ts) + PriKey \times IN\_C1)(r-1) \bmod Q$ $M = C2 - (d \times C1)$

· 카멜레온 해쉬(충돌값과 키 갱신)

본 논문에서 제안한 상호인증에 사용한 카멜레온 해쉬 기법은 표 8과 같이 값을 계산하여 동일한 CH값을 연산한 결과를 확인할 수 있다. 키 갱신을 통해 비밀값을 새롭게 선택하여 세션키와 암호화키 또한 새로운 값으로 갱신이 가능하다.

표 8. 카멜레온 해쉬 파라미터 검증
Table 8. Chameleon Hash Parameter Verification

Chameleon Hash					
g	313		IN_CEL	20221	
p	443		IN_r1	1657	
ASN or ADN	x	103	MN	x	137
	y	112		y	235
	r_1	145		r_1	506
	m_1	211		m_1	156
	CH	351		CH	351
	x_i	251		x_i	233
	y_i	242		y_i	322
	r_i	1296		r_i	442
	m_i	82		m_i	92
	CH	351		CH	351
	$CKEY$	286		$CKEY$	286
	$SKEY$	5		$SKEY$	5
	ENC	317		ENC	317

최초 IN에서 생성한 IN_CEL 값에 의해 IN_r1을 생성하고, MN과 ASN, ADN으로 전송을 시작으로 각 ASN, ADN과 MN에서 선택한 비밀값 x로 하여금 서로 다른 r_{i-1}, m_{i-1} 값을 가지고 있지만, 충돌값을 생성하는 IN_CEL과 IN_r1 값을 기반으로 동일한 CH값을 계산하여 정상적인 노드에서 생성한 CH값을 인증하는 장점을 가진다.

$$\begin{aligned} ASN\_CH_i &= ASN\_CH_{i\pm1} \\ &= MN\_CH_i = MN\_CH_{i\pm1} \end{aligned} \tag{29}$$

최초 IN_CEL과 IN_r1 값은 타원곡선 기반의 안전한 세션으로 전송하고, 두 값으로 인해 비밀값의 유추나 유출이 어려운 장점이 있고, 키 갱신과 암호화 검증에서 전송한 값을 중간자 공격에서 획득을 하여도 ASN, ADN과 MN이 선택한 비밀값 x에 대해서 유추하기 힘들다.

4.2 안전성 분석

· 데이터 및 키 값 노출

초기 노드 인증을 위한 $C1$, $C2$ 사이의 타원곡선 점을 얻어내는 방법은 Q를 구할 때까지 K, $2K = K + K$, $3K = K + K + K$ 등 연산을 연속적으로 계산해야 한다. 이는 K의 값이 큰 소수일 경우 이를 알아내기 어려운 타원곡선 이산 멱승 연산의 문제를 해결해야하기 때문에 공개키 교환과 암호문 교환에 안전성을 제공한다. 또한 암호화 시 계속 다른 r값을 이용하기 때문에 같은 메시지 M과 같은 키값을 사용해도 암호문의 값이 계속 변하는 장점이 있다.

· 사용자 위장 공격

암호화 과정에서 초기 IN_r0과 IN_m0, IN_m1 값을 노드간의 전송 없이 충돌값을 계산하여 상호인증을 수행하기 때문에 정상적인 사용자와 같이 카멜레온 해쉬값이나 충돌요소 값 및 충돌값에 대해 생성이 불가능하다.

· 재전송 공격

ASN, ADN 노드와 MN 노드 사이의 보안 세션키를 재사용하여 키교환을 하는 시도가 있을 수 있다. 하지만 식 (20), (21)과 같이 ASN, ADN과 MN이 선택한 비밀값 x에 대해 알아내거나 추측하기 어렵고, 이를 계산하기 위한 값 또한 추측하기 어렵다.

ASN_y와 MN_y를 가로채 재사용을 시도할 수 있다. 키 갱신 단계에서 ASN, ADN과 MN은 새로운 비밀값 x_i를 선택하기 때문에 세션키에 사용하는 $CKEY$ 뿐만 아니라 ASN_SKEY, MN_SKEY를 생성할 수 없고, 이산 멱승 연산의 문제를 가진다.

· 중간자 공격

ASN, ADN과 MN 사이의 중간자 공격을 시도할 수 있지만, 동일한 카멜레온 해쉬값을 인증요소로 이용하여 공개키에 대한 인증과 세션키를 제공하기 때문에 중간자 공격으로부터 안전하고, 또한 공격자는 동일한 카멜레온 해쉬값을 생성하지 못한다.

· 카멜레온 해쉬 비밀값 공격

기존의 카멜레온 해쉬 기법은 동일한 비밀값를 이용한 충돌값과 카멜레온 해쉬값을 계산하기 때문에 노드와 노드간의 전송하는 과정에서 키쌍 (m', r')과 (m'', r'')쌍을 수집한다면, 다음과 같은 식으로 비밀값 x에 대해 알아낼 수 있다.

$$\begin{aligned} &CHAM-HASH(m'', r'') \\ &\quad = CHAM-HASH(m', r') \\ &g^{m''}g^{xr''} \bmod p = g^{m'}g^{xr'} \bmod p \\ &\quad m'' + xr'' = m' + xr' \\ &\quad x = \frac{m' - m''}{r' - r''} \end{aligned} \tag{30}$$

제안하는 프로토콜은 초기 IN_CEL과 IN_r1값을 전송하지만, 초기 IN_r0과 IN_m0, IN_m1 값을 노드간의 전송 없이 충돌값을 계산하여 상호인증

을 수행하기 때문에 비밀값에 대한 공격은 어렵다.

초기 전송하는 IN_CEL과 IN_r1값으로 ASN, ADN에서 ASN_r2, ASN_m2를 생성하는데, ASN_r2에서 IN_CEL의 계산과 다른 ASN_x를 선택하므로 전송하는 ASN_CH 값을 수집하고 계산할 수 있는 카멜레온 키쌍을 추측하여 공격하기는 어렵다.

또한 식 (10)과 같이 키 갱신과 암호화 단계에서 전송하는 ASN_m_i는 매번 생성하는 타임스탬프 값 ASN_Ts가 포함되어 있고, 식 (31)과 같이 ASN_r_i는 기존에 선택한 ASN_x_{i-1}와 새롭게 선택한 ASN_x_i로 계산하기 때문에 비밀값 공격은 어렵다.

동일한 ASN, ADN 노드 내에서 생성한 x_i와 x_{i-1}에 대한 공격은 이원일차연립방정식의 문제가 되고, 동일한 ASN_CH_{i-1}, ASN_CH_i, MN_CH_i와 공개키 ASN_y, MN_y, ASN_y_i, MN_y_i는 새롭게 선택한 비밀값 x_i에 의해 생성되고, 카멜레온 해쉬와 공개키 값은 이산 멱승 연산으로 생성하기 때문에 각각 계산에 사용한 각각의 비밀값 x_{i-1}와 x_i에 대해 추측하거나 공격하기는 매우 어렵다.

$$\begin{aligned} &ASN\_r_i \\ &\quad = ASN\_r_{i-1} + (ASN\_x_{i-1} \times ASN\_m_{i-1} \\ &\qquad\qquad - ASN\_x_i \times ASN\_m_i) \\ &\qquad\qquad \Downarrow \\ &ASN\_r_i + ASN\_x_i \times ASN\_m_i \\ &\quad = ASN\_r_{i-1} + ASN\_x_{i-1} \times ASN\_m_{i-1} \\ &\qquad\qquad \Downarrow \\ &\qquad x \neq \frac{m' - m''}{r' - r''} \end{aligned} \tag{31}$$

· 강력한 신분 확인

상호인증과 키 갱신 및 암호화에 사용하는 파라미터 값은 초기에 비밀값와 합한 값을 계산한 충돌 요소값과 충돌값으로 프로토콜이 전개되고, 정상적인 노드에서만 생성할 수 있는 동일한 카멜레온 해쉬값으로 인증을 수행한다.

이후 새로운 비밀값으로 동일한 카멜레온 해쉬값을 계산하고, 이에 따른 세션키와 암호키 또한 새롭게 갱신되기 때문에 ASN, ADN 또는 MN에 대한 강력한 상호인증을 제공한다.

4.3 효율성 분석

분석 환경은 리눅스(Ubuntu 12.04 32bit), Intel(R) Core(TM) Duo T2450 2.00GHz, 2.00GB 메모리, java 코드로 구현된 127바이트 크기(IEEE 802.15.4 CoAP 메시지 포맷 최대 크기) 텍스트의 암·복호화 성능을 비교하였다. 보다 폭넓은 비교분석을 위해 표준 블록암호(DES, AES 등)이외에 국산 블록암호 SEED와 DES의 대안으로 공개된 BLOWFISH 등을 추가 비교분석 하였다.

표 9는 일반적으로 알려진 암호화 용도의 알고리즘과 제안하는 카멜레온 해쉬 함수의 성능분석 결과(단위 : nanosecond)를 나타낸다.

연산 방법은 비교대상 알고리즘 표준에서 최소 요구되는 블록 및 키길이, ECB(Electronic CodeBook) 운영모드를 사용하였고, 제안 프로토콜의 카멜레온 해쉬함수 기법은 비교적 비슷한 크기인 64비트 블록과 128비트 키길이를 적용하였다.

성능 분석 결과는 키 생성 부분을 제외한 각 암호화 및 복호화 100회 연산에 대한 시간의 최대, 최소, 평균을 비교하였고, 그림 5는 각 암호화 및 복호화와

표 9. 제안 알고리즘 성능 비교(시간 - 나노세컨드)
Table 9. Proposal Algorithms Performance Comparison(Time - Nanosecond)

	DES	3DES	SEED	AES	BLOWFISH	Proposal: Chameleon
Block Size	64	128	128	128	64	64
Key Size	56	168	128	128	112	128
Time(Max)	110786	870593	425032	285871	1181941	98143
	245119	844632	446768	135842	260212	174976
Time(Min)	18716	52224	38640	8150	35017	32817
	9962	53733	36224	9358	36224	21965
Time(Avg)	35321	93696	82085	21412	66613	44168
	13118	91897	78713	28232	57319	27519
Total(Avg)	48439	185593	160798	49644	123932	71687

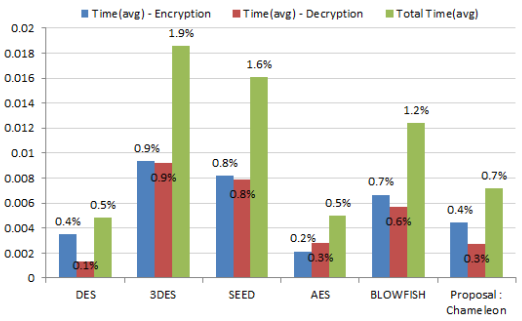

그림 5. 평균 수행 시간에 대한 결과
Fig. 5. The Results of the Average Execution Time

전체 평균시간의 비율(정규화 범위 : 최소 0.1 기준)을 나타낸다.

연산성능 순으로 정렬했을 때 3DES(1.9%), SEED(1.6%), BLOWFISH(1.2%), 제안 카멜레온 해쉬함수(0.7%), AES, DES(0.5%) 순으로 나타났다.

DES의 경우 블록, 키 크기(64, 56)가 낮아 연산 성능은 가장 높게 나타났지만 Brute Force 공격에 대해 취약성이 알려져 있기 때문에 암호화 알고리즘 보다는 무결성 검사용도로 사용된다.

제안 알고리즘은 동일한 수준인 AES, DES와 전체 평균 0.2%차이로 성능이 비교적 비슷한 것으로 나타났으며 3DES, SEED, BLOWFISH에 비해 전체 평균 각 1.2%, 0.9%, 0.5% 차이로 효율이 높은 것으로 나타났다. 암호화 부분에서는 오히려 DES 0.4%와 동일한 수준으로 연산속도가 효율적인 것을 확인할 수 있다.

그림 6과 같이 성능 효율이 낮은 3DES(29%), SEED(25%), BLOWFISH(19%)를 제외하고 성능이 비교적 동등하게 분석된 3DES(8%), AES(8%)와 비교 하였을 때 제안 알고리즘(11%)은 해쉬함수 특성상 연산 속도가 빠른 결과를 나타내며, 100회 연산에 대한 평균 연산속도 3% 차이는 실제 약 0.003초

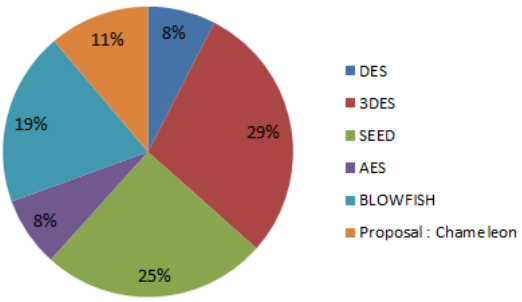

그림 6. 제안 기법에 대한 효율성 비교 결과
Fig. 6. The results of the Proposed Scheme Efficiency Compare

(Seconds)로 비교적 동등한 수준을 확인하였다.

V. 결 론

본 논문에서는 타원곡선 키교환 기법과 카멜레온 해쉬를 이용하여 M2M 환경에 안전한 상호인증 프로토콜을 제안하였다. 프로토콜 초기 키교환 과정에서 타원곡선 기반으로 보안세션을 설립하고, 이후 암호화 과정에서 카멜레온 해쉬 기반으로 새로운 비밀값로 갱신하는 특징을 가진다. 결론적으로 두 알고리즘의 적절한 응용을 통해 상호인증에서 효율적인 재인증과 새로운 세션키를 자동으로 갱신하는 강한 이점을 가진다.

또한 카멜레온 해쉬 함수는 동일한 해쉬값을 생성하는 충돌값은 새로운 비밀값에 의해 다른 충돌값을 생성하기 때문에 사용자 위장 공격, 재전송 공격, 중간자 공격과 비밀값 등 다양한 공격에 안전하다는 것을 안전성 분석에서 입증하였다.

References

[1] H. Kwon and N. Kang, “Analysis on energy consumption required for building DTLS session between lightweight devices in internet of things,” *J. KICS*, vol. 40, no. 8, pp. 1588-1596, 2015.

[2] S. Choi, *M2M / IoT service practices and development prospects*, 2013, from http://www.tta.or.kr.

[3] U. Jeon and S. Bak, “KT’s M2M / IoT services platform,” *J. KICS*, vol. 30, no. 8, pp. 40-45, Jul. 2013.

[4] Infonetics Research, *As businesses turn to the internet of things for growth, M2M WAN connections set to triple by 2018*, Retrieved Jul. 30, 2015, from http://www.infonetics.com/pr/2014/2H13-M2M-Connections-and-Services-by-Vertical-Market-Highlights.asp

[5] S. Tiazkun and M. Kumar, *Worldwide internet of things spending by vertical market 2014 - 2017 forecast*, IDC, 2014, from http://www.idc.com.

[6] KISA, *Internet threat trend things,* Korea Internet & Security Agency, 2014, from http://www.kisa.or.kr.

[7] oneM2M-TS-0001, o*neM2M functional architecture technical specification*, v0.2.1, 2013.
[8] Z. Kim, J. Kim, S. Yoo, and J. Lee, "Wireless technology for M2M / IoT services," *J. KICS*, vol. 30, no. 8, pp. 11-19, Jul. 2013.
[9] oneM2M-TR-0008, *Analysis of security solutions for oneM2M system*, v0.2.1, 2013.
[10] H. Yoo and K. Sung, "Analysis and implementation of digital signature algorithm using hash function," *J. KITS*, vol. 6, no. 3, pp. 129-142, Jun. 2011.
[11] KISA, *The trend of project related to technology for personal information protection*, Korea Internet & Security Agency, 2006.
[12] X. Chen, F. Zhang, and K. J. Kim, "Chameleon hashing without key exposure," *Information Security Conf.*, pp. 87-98, Palo Alto, CA, USA, Sept. 2004.
[13] F. Zhang, S. N. Reihaneh, and W. Susilo. *ID-based chameleon hashes from bilinear pairings,* IACR Cryptology ePrint Archive, Report, 2003.
[14] A. Giuseppe and D. M. Breno, "On the key exposure problem in chameleon hashes," in *Proc. Security in Commun. Netw.*, pp. 165-179, Amalfi, Italy, Sept. 2004.
[15] A. Giuseppe and D. M. Breno, "Identity-based chameleon hash and applications," in *Proc. Financial Cryptography*, pp. 164-180, Key West, FL, USA, Feb. 2004.
[16] H. Krawczyk and T. Rabin, "Chameleon signatures," in *Proc. Netw. and Distrib. Syst. Security Symp.*, pp. 143-154, San Diego, California, USA, Feb. 2000.

김 성 수 (Sung-soo Kim)

2008년 2월 : 영동대학교 컴퓨터공학과 졸업
2010년 2월 : 숭실대학교 컴퓨터학과 석사
2010년 3월~현재 : 숭실대학교 컴퓨터학과 박사과정
<관심분야> 정보보호, 인증 이론, 암호 알고리즘

전 문 석 (Moon-seog Jun)

1981년 2월 : 숭실대학교 전자계산학과 졸업
1986년 2월 : University of Maryland Computer Science 석사
1989년 2월 : University of Maryland Computer Science 박사
1991년 3월~현재 : 숭실대학교 컴퓨터학과 정교수
<관심분야> 정보보호, 네트워크 보안, 암호학

최 도 현 (Do-hyeon Choi)

2008년 2월 : 동서울대학교 컴퓨터소프트웨어학과 졸업
2010년 8월 : 숭실대학교 컴퓨터학과 석사
2010년 9월~현재 : 숭실대학교 컴퓨터학과 박사과정
<관심분야> 모바일 보안, PKI, 시큐어코딩, 가상화