Journal of Korea Multimedia Society Vol. 16, No. 12, December 2013(pp. 1413-1426)
http://dx.doi.org/10.9717/kmms.2013.16.12.1413

간단한 연산을 사용한 RFID 태그 소유권 이전 프로토콜

이 재 동†

요 약

RFID 기술은 공급망 관리를 위한 산업에 광범위하게 채택되어 사용되고 있다. 제품이 생산될 때 각 제품에 RFID 태그가 부착되며, 공장, 배급자, 소매상 및 소비자 사이에서의 공급망 관리를 위해 제품의 소유권 이전이 주의 있게 처리되어야 한다. 이와 같이 RFID 기술을 사용하여 각 제품을 식별하고 제품의 공급과정을 효율적으로 처리하기 위해 안전하고 효율적인 RFID 소유권 이전 프로토콜이 중요한 이슈이다. 이미 많은 소유권 이전 프로토콜이 제안되었다. 이들 프로토콜들은 보안 상의 문제를 가지고 있으며, 대부분의 프로토콜은 암호화 연산 및 해시함수 같은 복잡한 연산을 사용한다. Lo 등은 간단한 연산들(시프트, 덧셈, XOR 연산 및 난수 생성)을 사용한 프로토콜을 제시하였다[1]. 하지만, 이 프로토콜은 태그와 새 소유주가 공유하는 비밀키를 공격자가 획득할 수 있는 문제점과 Fraud 공격에 취약하다[2]. 본 논문에서는 Lo 등이 제시한 프로토콜을 수정하여 간단한 연산들(시프트, 덧셈 연산 및 난수 생성)을 사용하여 보안 공격에 안전한 새로운 프로토콜을 제시한다.

RFID Tag Ownership Transfer Protocol Using Lightweight Computing Operators

Jae-Dong Lee†

ABSTRACT

RFID thecnology has been widely adopted by industries for supply chain management. When a product item is manufactured RFID tag is attached to the product item and supply chain management among factories, distributors, retailers and customers needs to handle ownership transfer for the product item carefully. With RFID technology, the secure and efficient ownership transfer protocol is an important issue for the identification of product items and the overall system efficiency on supply chain. Many ownership transfer protocols have been proposed now. They have security problems and use complex operations such as encryption operation and hash function. Lo *et al.* proposed a protocol using lightweight operations such as shift, addition, XOR, and random number generation[1]. However, this protocol has a security problem in which the secret key between the tag and the new owner is disclosed to the attackers, and it is also weak against the Fraud attack[2]. In this paper, we propose a new ownership transfer protocol using lightweight operations such as shift, addition, and random number generation. This protocol is the modified version of Lo *et al.*'s protocol and is secure against the security attacks.

Key words: RFID Security(RFID 보안), Ownership Transfer Protocol, Lightweight operation(소유권 이전 프로토콜)

※ 교신저자(Corresponding Author) : 이재동, 주소 : 경남 창원시 마산합포구 월영동 449번지 경남대학교 컴퓨터공학과(631-701), 전화 : 055) 249-2214, FAX : 055) 248-2554, E-mail : jdlee@kyungnam.ac.kr

접수일 : 2013년 7월 30일, 수정일 : 2013년 11월 5일
완료일 : 2013년 11월 12일
† 정회원, 경남대학교 컴퓨터공학과

1. 서 론

RFID(Radio Frequency IDentification) 시스템은 유비쿼터스 컴퓨팅의 중요한 기반 기술로 자리 잡고 있다. 이것은 RFID 태그(tag)들을 사용하여 사물(또는 사람)에 관한 정보를 저장하고 사물과 떨어진 곳에서 리더(reader)를 사용하여 사물의 정보를 접근할 수 있는 자동 인식 시스템이다. 사물에 부착된 태그들은 인증 정보를 가지고 있으며, 이것을 사용하여 재고관리, 물품관리, 물류 및 유통, 고용자 관리, 미아 찾기 등 많은 분야에 활용되고 있으며, 미래에는 더 많은 분야에 활용될 것이다. RFID 시스템은 태그, 리더 그리고 백앤드 서버(backend server)로 구성된다.

RFID 기술은 공급망 관리(supply chain management)를 위한 산업에 광범위하게 채택되어 사용되고 있다[3]. 제품이 생산될 때 각 제품에 RFID 태그가 부착되며, 공장, 배급자(distributor), 소매상 및 소비자 사이에서의 공급망 관리를 위해 제품의 소유권 이전이 주의 있게 처리되어야 한다. 이와 같이 RFID 기술을 사용하여 각 제품을 식별하고 제품의 공급과정을 효율적으로 처리하기 위해 안전하고 효율적인 RFID 소유권 이전 프로토콜이 중요한 이슈이다.

이미 많은 소유권 이전 프로토콜이 제안되었다[4-11]. 이들 프로토콜들은 2장에서 언급하는 바와 같이 보안 상의 문제를 가지고 있으며, 대부분의 프로토콜은 암호화 연산 및 해시함수 같은 복잡한 연산 즉, 태그 및 서버에서 수행할 연산의 양이 많아진다. 이것은 태그의 설계를 복잡하게 하여 게이트(gate) 수가 증가하므로 태그의 비용이 비싸진다. Lo 등은 간단한 연산들(시프트, 덧셈, XOR 연산 및 난수 생성)을 사용한 프로토콜을 제시하였다[1]. 하지만, 이 프로토콜은 태그와 새 소유주(new owner)가 공유하는 비밀키를 공격자가 획득할 수 있는 문제점과 Fraud 공격에 취약하다[2]. Fraud 공격에 취약하기 때문에 이전 소유주(old owner)는 같은 물건을 여러 사람에게 이전 할 수 있게 되는 보안 상의 문제점이 발생한다. 본 논문에서는 Lo 등이 제시한 프로토콜을 수정하여 간단한 연산들(시프트, 덧셈 연산 및 난수 생성)을 사용하여 보안 공격에 안전한 새로운 프로토콜을 제시한다. 우리가 제시한 프로토콜은 2장에서 언급한 보안 요구 조건을 모두 만족하며, Lo 등이 제시한 프로토콜의 문제점들을 모두 해결할 수 있다.

본 논문은 다음과 같이 구성되어 있다. 2장에서는 관련 연구를 기술한다. 관련 연구로 보안 요구 조건, 표기법 및 가정, 기존 알고리즘들의 문제점, Lo 등이 제시한 프로토콜 및 보안 취약점을 기술한다. 3장에서는 새로운 프로토콜을 제시하고, 4장에서는 제시한 프로토콜이 보안 공격에 안전함을 보인다. 5장에서는 결론과 향후연구에 대한 방향을 제시한다.

2. 관련 연구

본 절에서는 보안 요구 조건, 표기법 및 가정, 기존 소유권 이전 프로토콜의 문제점, Lo 등이 제시한 프로토콜 및 보안 취약점 등을 기술한다.

2.1 보안 요구 조건

제품에 부착된 RFID 태그에는 소유자 정보 및 접근 제어를 위한 데이터가 저장되어 있으므로 완전한 태그 소유권 이전 프로토콜은 아래의 보안 및 프라이버시 요구사항을 만족시켜야 한다[1,4-6].

새 소유주 프라이버시(New owner privacy): 태그의 소유권이 새 소유주에게로 이전 되었을 때, 단지 새 소유주만이 태그를 식별하고 태그 내의 정보를 접근할 수 있다. 이전 소유주는 그 태그에 대해 더 이상 접근할 수 없다.

전 소유주 프라이버시(Old owner privacy): 태그의 소유권이 새 소유주에게 이전 되었을 때, 새 소유주는 전 소유주와 태그 사이의 과거 활동을 추적할 수 없다.

권한 복구(Authorization recovery): RFID 태그가 붙은 제품에 대한 애프터 서비스(after-sales service)와 같은 상황에서, 현재의 소유주는 전 소유주가 태그 내의 정보를 접근할 권한을 잠시 동안 가지도록 태그의 소유권을 잠시 이전할 필요가 있다.

서비스 거부 공격에 안전(Resistance to Denial of Service (DoS) attack): DoS 공격 을 막기 위해서 태그와 백앤드 서버가 공유하고 있는 비밀정보에 대한 동기화 메커니즘이 요구된다.

재생 공격에 안전(Resistance to replay attack): 공격자가 리더와 태그 사이의 통신 메시지를 도청하여 리더나 태그를 속이기 위해 도청한 메시지를 사용할 수 없도록 한다.

중간자 공격에 안전(Resistance to man-in-mid-

dle attack): 공격자가 태그와 리더 사이에서 가짜 메시지 또는 수정한 메시지를 사용하여 원하는 일을 할 수 없도록 한다.

윈도잉 문제의 방지(Prevention of windowing problem)[7]: 만약 소유권 이전 프로토콜을 잘못 설계하면 새 소유주와 전 소유주가 같은 공유 비밀 키를 사용하여 태그를 접근할 수 있는 기간이 있다.

2.2 표기법 및 가정

본 논문에서 사용할 기호들의 의미는 표 1과 같다.

표 1의 기호 중, *TTP*는 신뢰할 수 있는 제 3의 서버를 의미한다. 즉, *TTP*로 RFID 시스템의 백앤드 서버를 사용할 수도 있고, 다른 신뢰 할 수 있는 서버를 사용할 수도 있다. C_j는 소유주 j를 확인하기 위해 사용되며 $C_j = h(TTPID \,//\, OID_j)$ 로 구한다. *Nun* 함수는 일방향 해시함수처럼 전송 시, 공격자가 그 내용을 추출 및 조작을 할 수 없게 간단한 연산으로 만든 함수이다.

대부분의 RFID 시스템처럼 태그와의 통신은 리더를 통해서 이루어지며, 리더와 백앤드 서버 사이의 통신은 안전한 통신 채널을 사용하고 태그와 리더 사이의 통신은 노출된(안전하지 않는) 것으로 가정한다. 따라서, O_C(또는 O_N)와 태그 사이의 통신을 위해서는 O_C(또는 O_N)에 있는 리더를 통해 이루어지며, 이 통신은 노출되어 있다. 반면, O_C(또는 O_N)와 *TTP*와의 통신은 안전한 채널로 이루어진다. 그리고, 태그와 리더 사이의 통신을 위해서는 상호 인증이 이루어져야 한다. 리더와 태그의 사이의 상호 인증 문제는 본 논문의 관심사가 아니므로 여기서는 다루지 않으며 상호 인증이 이루어진 후 통신하는 것으로 가정한다.

```
string Nun(bit string m, bit string n){
  string X = m
  int L = bit length of X value
    for(int i = 0; i < L; i++){
        X = (X >> 1) + (X << 1) + n
    }
  return X
}
```

그림 1. *Nun* 함수의 의사 코드

2.3 기존 소유권 이전 프로토콜

많은 소유권 이전 프로토콜들이 제안되었다. 이들은 *TTP*를 사용하는 프로토콜[1, 5-12]과 *TTP*를 사용하지 않고 현 소유주와 새 소유주 사이의 통신만으로 처리하는 프로토콜[4,9,11]로 나눌 수 있다. 또한, *TTP*를 사용하는 프로토콜 중에는 *TTP*와 태그가 직접 통신(당연히, 리더를 통하여 통신함)하는 프로토콜[6,8,11]과 *TTP*가 태그와 통신하기 위해서는 반드시 현 소유주(또는, 새 소유주)를 거쳐야 되는 프로토콜[1,5-8,10]로 나눌 수 있다.

표 1. 기호들의 의미

기 호	설 명
TTP	신뢰할 수 있는 서버(The trusted third party server)
O_C	현재 소유주
O_N	소유권 이전 후의 새 소유주
Tag_i	RFID 태그 i
TID_i	태그 i의 식별자(identity)
OID_j	소유주 j의 식별자
TTPID	TTP의 식별자
C_j	소유주 j의 소유증명서(credential)
KT_i	Tag_i와 *TTP*가 공유하는 비밀키
KO_i	Tag_i와 현재 소유주가 공유하는 비밀키
Nun()	그림 1에서 정의한 덧셈과 시프트 연산을 사용한 함수
$\oplus$	XOR 연산
r_x	난수
h()	일방향 해시함수(one-way hash function)

기존 프로토콜은 표 2에 나타낸 바와 같이 보안상의 문제점을 노출하고 있다[1]. 전 소유주 프라이버시 문제를 가지는 프로토콜[8,9], 서비스 거부 공격에 노출된 프로토콜[4,5,7,9,12], 재생 공격에 노출된 프로토콜[8], 중간자 공격에 노출된 프로토콜[4,9,10] 그리고 윈도잉 문제를 가지는 프로토콜[4,6,8-10,12] 등이 있다.

기존 프로토콜들에서 태그에서 수행되는 연산들을 표 3에 제시하였다[1]. 대부분의 프로토콜은 암호화 연산 및 해시함수 같은 복잡한 연산 즉, 태그 및 서버에서 수행할 연산의 양이 많아진다. 이것은 태그의 설계를 복잡하게 하여 게이트(gate) 수가 증가하므로 태그의 비용이 비싸진다.

2.4 Lo 등이 제안한 프로토콜

Lo 등은 간단한 연산들(시프트, 덧셈, XOR 연산 및 난수 생성)을 사용한 보안상 안전한 프로토콜을 제시하였다[1]. 이 프로토콜은 세 단계로 이루어져 있다. 등록 단계(registration phase), 현 소유권 보류 단계(current ownership suspension phase), 그리고 새 소유권 수립 단계(new ownership establish phase)이다.

2.4.1 등록 단계

모든 태그들은 사용 전 초기화 되고, 모든 소유주들의 식별자는 TTP에 등록 된다. TTP와 태그 i는 비밀 키 KT_i를 공유한다. KT_i는 태그 i에 미리 저장되어 있으며, 이 키는 현 소유주와 새 소유주는 알 수 없다. 각 소유주 j는 소유증명서 $C_j = h(TTPID \,//\, OID_j)$를 가지고 있다. 이것은 소유주가 TTP에게 소유권 이전 요청을 할 때, 요구한 소유주가 진짜 소유주인가를 확인하기 위해 사용 된다.

2.4.2 현 소유권 보류 단계

이 단계는 그림 2와 같이 동작한다. 동작 내용을 간단히 요약하면 아래와 같다.

1) O_C 는 난수 r_1를 생성하여 (OID_C, OID_N, TID_i, r_1, C_C)를 TTP로 보낸다.
2) 메시지를 받은 TTP는 $C_C' = h(TTPID \,//\, OID_C)$를 계산한다. C_C'와 C_C가 다르면 세션을 끝낸다. 같으면, TTP는 아래와 같은 작업을 한다.
   - Tag_i 에 대한 KT_i를 찾는다.
   - 임시 비밀키 K_{temp}를 생성한다.
   - $M_1 = KT_i \oplus K_{temp} \oplus r_1$과 $V_1 = Nun(M_1, r_1)$를 계산한다.
   - M_1과 V_1를 O_c로 전송한다.
3) O_C가 메시지를 받으면, $M_2 = M_1 \oplus KO_i$를 계산하고 (V_1, M_2, r_1)를 Tag_i로 전송하고 태그의 반응시간을 제어하기 위해 타이머를 시작한다.
4) Tag_i가 메시지를 받으면, $M_1' = M_2 \oplus KO_i$과 $V_1' = Nun(M_1', r_1)$를 구한다. V_1'과 V_1이 같으면 $K_{temp}(K_{temp} = M_1 \oplus KT_i \oplus r_1)$를 추출하고, 이것을 비밀키 KO_i로 설정한 다음, $ACK_1 = Nun(KT_i, r_1) \oplus KO_i$를 응답 메시지로 O_C로 전송한다. 그러나, V_1'과 V_1이 같지 않으면 난수 r_2를 응답 메시지 ACK_1로 설정하여 O_C로 전송한다.
5) O_C가 타임 아웃 이전에 응답 메시지를 받으면, 받은 ACK_1을 TTP로 보낸다. 타임 아웃 이후에 메시지를 받았으면 O_C는 KO_i를 사용하여 Tag_i를 액세스해 본다. 액세스를 할 수 없으면 ACK_1를 OID_C 로 하여 ACK_1를 TTP로 전송한다. 액세스가 가능하면 그림 2의 Step3부터 다시 작업을 수행한다.
6) TTP가 받은 메시지 ACK_1이 $Nun(KT_i, r_1) \oplus K_{temp}$ 또는 OID_C이면 O_C에게 "success" 메시지를 보내고, O_N에게 K_{temp}와 TID_i를 전송한다. 그렇지 않으면 TTP는 O_C에게 실패 메시지로 “go to Step3” 메시지를 보내 O_C로 하여금 Step3부터 다시 수행토록 한다.

2.4.3 새 소유권 수립 단계

이 단계는 그림 3과 같이 동작한다. 동작 내용을 간단히 요약하면 아래와 같다.

1) O_N이 K_{temp}를 받으면 아래와 같은 작업을 수행한다.
   - 새로운 비밀키 KO_{i_new}와 난수 r_3와 r_4를 생성한다.
   - $M_3 = KO_{i_new} \oplus r_3 \oplus KO_i$와 $V_2 = Nun(KO_{i_new}, r_4) \oplus KO_i$를 계산한다.
   - 메시지 (M_3, V_2, r_3, r_4)를 Tag_i로 보내고 타이머를 시작시킨다.
2) Tag_i가 메시지를 받으면 $KO_{i\_new}' = M_3 \oplus KO_i \oplus$

r_3 와 $V_2' = Nun(KO_{i_new}', r_4) \oplus KO_i$를 계산한다. V_2'와 V_2가 같으면 KO_i를 KO_{i_new}로 바꾸고, $ACK_3 = Nun(KT_i, r_4) \oplus KT_i$로 설정하여 O_N으로 전송한다. 같지 않으면, 하나의 난수를 생성하여 그것을 ACK_3로 설정하여 O_N으로 전송한다.

3) O_N이 타임 아웃 전에 태그로부터 메시지를 받으면, 받은 메시지 ACK_3을 TTP로 전송한다. 타임 아웃이 발생했으면 O_N은 이전 키 KO_i를 사용하여 Tag_i를 액세스해 본다. 액세스를 할 수 없으면 ACK_3을 OID_N으로 하여 ACK_3을 TTP로 전송한다. 액세스가 가능하면 그림 3의 Step1부터 다시 작업을 수행한다.

4) TTP가 받은 메시지 ACK_3이 $Nun(KT_i, r_4) \oplus K_i$ 또는 OID_N이면 O_N에게 "success" 메시지를 보내고, 그렇지 않으면 TTP는 O_N에게 실패 메시지로 "restart" 메시지를 보내 O_N로 하여금 Step1부터 다시 수행토록 한다.

5) O_N이 성공 메시지를 받으면 KO_i를 KO_{i_new}로 설정하고, 그렇지 않으면 O_N로 하여금 Step1부터 다시 수행토록 한다.

2.5 Lo 등이 제안한 프로토콜의 보안 상 문제점

Lo 등이 제시한 프로토콜은 보안 상 3가지 문제점을 가지고 있다[2]. 태그 키(KT_i)의 노출, 태그와 새 소유주 간의 공유 비밀키(KO_i)의 노출, 그리고 Fraud 공격에 취약한 점 등이다. 그 이유를 아래에 기술한다.

문제점1: 태그 키(KT_i)의 노출

현 소유주 O_C가 자기 자신(또는 공모한 다른 사람 O_N)에게 소유권을 이전하기 위해 이 프로토콜을 수행하면 새 소유주 수립 단계에서 TTP는 O_C(또는 공모한 O_N)에게 K_{temp}를 전달한다. 이 K_{temp}와 현 소유권 보류 단계에서 획득한 M_1과 r_1을 이용하여 $M_1 \oplus K_{temp} \oplus r_1$을 계산하여 KT_i를 획득할 수 있다 획득한 KT_i를 이용하여 해당 태그를 추적할 수 있을뿐더러 문제점2에서 설명하는 방법으로 태그와 새 소유주 간의 공유 비밀 키를 획득할 수 있다.

문제점2: 태그와 새 소유주 간의 공유 비밀 키(KO_i)의 노출

공격자는 현 소유권 보류 단계에서 (M_1, r_1)을 도청하여 문제점1에서 획득한 KT_i를 사용하여 $K_{temp} = M_1 \oplus KT_i \oplus r_1$으로 K_{temp}를 구한 다음, 새 소유주 수립 단계에서 (M_3, V_2, r_3, r_4)를 도청하여 $KO_{i_new} = M_3 \oplus r_3 \oplus K_{temp}$로 새로운 KO_i를 얻을 수 있다. 이 새로운 KO_i를 사용하여 태그의 주인으로 행세할 수 있다.

문제점3: Fraud 공격에 취약

이 프로토콜을 이용하여 O_C가 O_N에게로 소유권 이전이 끝났을 때, 'O_N은 Tag_i를 엑세스할 수 없게 하고 O_C는 Tag_i를 계속 엑세스할 수 있게' 할 수 있다. 그래서 O_C는 똑같은 태그를 여러 사람에게 이전할 수 있다. 이것은 아래와 같은 공격으로 가능하다.

(1) O_C가 TTP로부터 (M_1, r_1)메시지를 받았을 때, O_C는 Tag_i에게 어떠한 메시지도 보내지 않고 OID_C를 ACK_1로 하여 TTP에 보낸다.

(2) TTP가 ACK_1을 받았을 때, ACK_1이 OID_C이므로 success 메시지를 O_C에 보내고 K_{temp}와 TID_i를 O_N에게 보낸다.

(3) O_N이 K_{temp}와 TID_i를 받으면 Tag_i를 엑세스하기 위한 키 KO_i를 K_{temp}로 하고, 새로운 키 KO_{i_new}를 생성한 다음, (M_3, V_2, r_3, r_4)메시지를 Tag_i에게 보낸다.

(4) O_C는 (M_3, V_2, r_3, r_4) 메시지를 Tag_i가 받지 못하도록 블록(block) 시킨다.

(5) O_N은 태그로부터 응답 메시지를 받지 못했기 때문에 타임 아웃이 되어 KO_i를 사용하여 Tag_i를 엑세스하려고 한다. 하지만 O_N은 태그를 엑세스할 수 없다. 그래서 $ACK_3 = OID_N$으로 하여 (ACK_3, r_4)를 TTP로 보낸다.

(6) TTP가 이 메시지를 받아 ACK_3가 OID_N임을 확인한 후, O_N에게 success 메시지를 보낸다.

(7) O_N이 success 메시지를 받으면 KO_i를 KO_{i_new}로 바꾸어 자신이 Tag_i에 대한 소유권을 성공적으로 받았다고 확신한다.

위에서 설명한 공격에 의해 새 소유주는 소유권 이전이 성공적으로 이루어졌으므로 자신만이 이 태그(Tag_i)를 식별하고 접근 할 수 있으며, 전 소유주는 이 태그를 더 이상 식별 및 접근을 할 수 없다고 확신한다. 그러나, 이 공격은 전 소유자가 새 소유자를 속여 새 소유자는 태그에 접근할 수 없고, 전 소유자

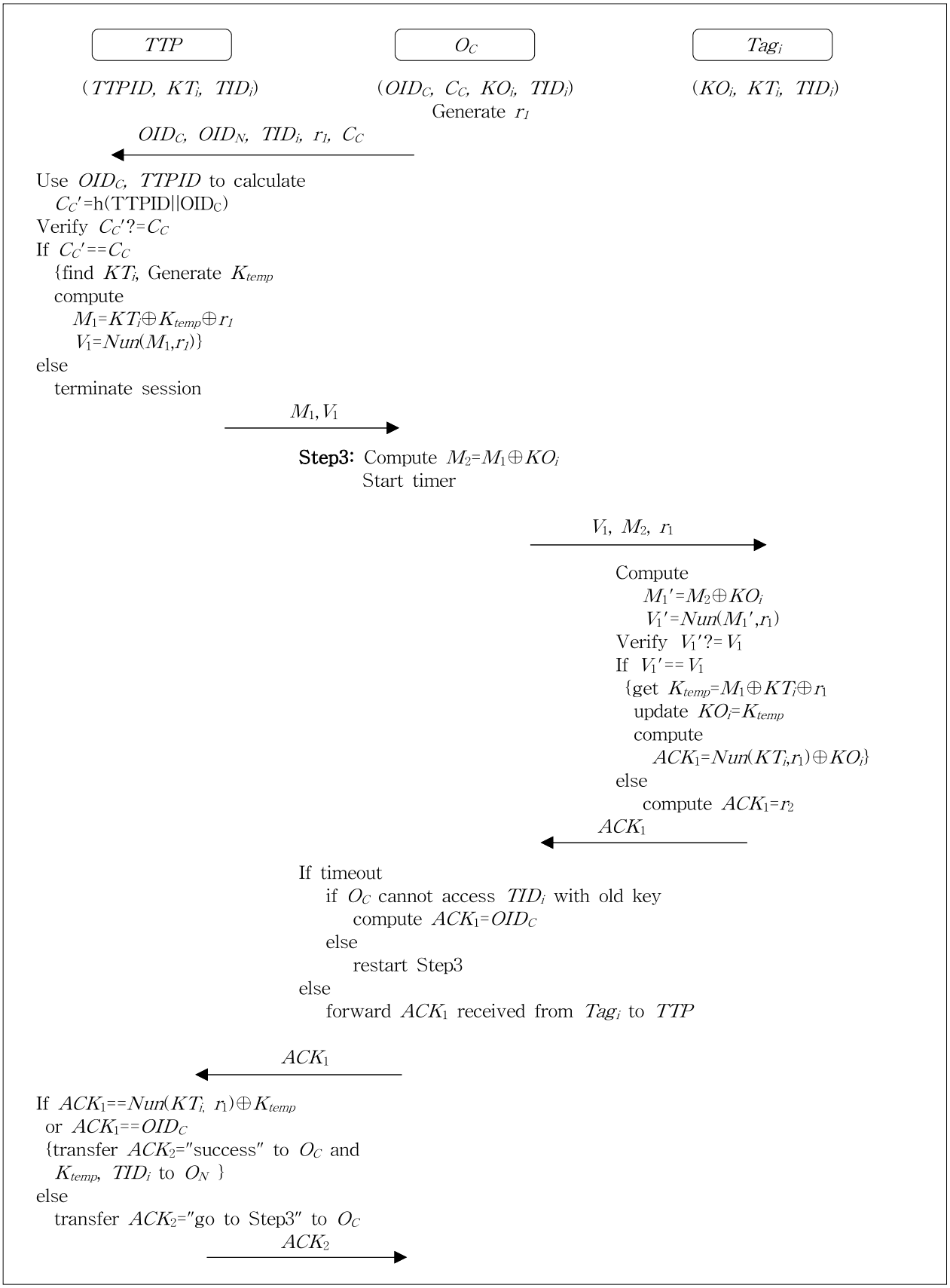

그림 2. 현 소유권 보류 단계

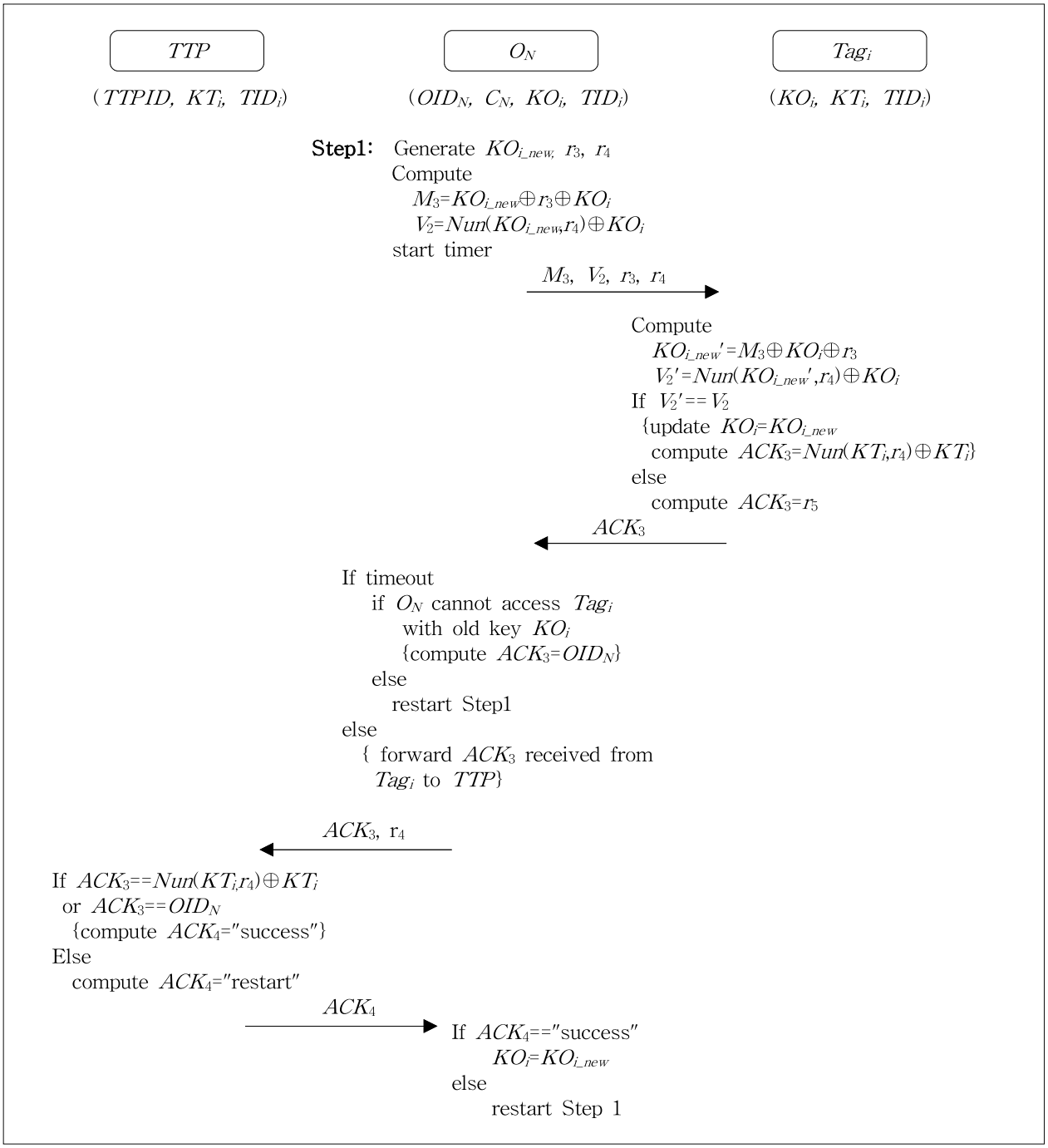

그림 3. 새 소유권 수립 단계

는 다른 사람에게 다시 이 태그의 소유권을 넘길 수 있음을 보여준다.

3. 간단한 연산을 사용한 안전한 소유권 이전 프로토콜

본 장에서는 Lo 등이 제시한 프로토콜을 수정하여 간단한 연산들(시프트, 덧셈 연산 및 난수 생성)을 사용하여 보안 요구 조건을 만족하며, 2.5절에서 언급한 기존 프로토콜의 문제점들을 해결한 프로토콜을 제시한다.

3.1 기본 구조

새 프로토콜은 등록 단계와 소유권 이전 단계

(registration phase)로 구성되어 있다. 등록 단계는 Lo 등의 프로토콜과 마찬가지로, 모든 태그들은 사용 전 초기화 되고, 모든 소유주들의 식별자는 TTP에 등록 된다. TTP와 태그 i는 비밀 키 KT_i를 공유한다. KT_i는 태그 i에 미리 저장되어 있으며, 이 키는 현 소유주와 새 소유주는 알 수 없다. 각 소유주 j는 소유증명서 $C_j = h(TTPID \,||\, OID_j \,||\, KO_i)$를 가지고 있다. 이것은 소유주가 TTP에게 소유권 이전 요청을 할 때, 요구한 소유주가 진짜 소유주인가를 확인하기 위해 사용된다. 소유주와 태그는 비밀키(KO_i)를 공유하며, 소유주는 이 비밀키를 사용하여 태그를 엑세스한다. 새 프로토콜에서는 TTP가 KO_i를 저장하고 있는 것으로 가정한다. 이것은 그림 4의 윗부분에 나타나 있다. 즉, TTP는 ($TTPID$, KT_i, TID_i, KO_i)을, O_C는 (OID_C, C_C, KO_i, TID_i)을, 그리고 Tag_i는 (KO_i, KO_i^{old}, KT_i, KT_i^{old}, TID_i)을 저장하고 있다. KO_i^{old}와 KT_i^{old}는 TTP와 Tag_i가 공유하는 키 KO_i와 KT_i의 동기화를 위해 KO_i와 KT_i의 이전 값을 가진다. 초기값으로 첫 번째 KO_i와 KT_i의 값으로 설정되어 있다.

새 프로토콜에서는 [6,8,11]의 프로토콜처럼 TTP와 태그가 직접 통신(당연히, 리더를 통하여 통신함)하는 것으로 설계하였으며, 소유권 이전이 성공하면 KT_i와 KO_i는 새로운 값으로 바뀌게 설계하였다. KT_i는 $Nun(KT_i, TID_i)$로, KO_i는 $Nun(KO_i, KT_i)$로 계산된 새로운 값으로 변경된다. 이 때, TTP는 다른 모든 TTP에게 변경된 KT_i와 KO_i를 전송하여 키들의 동기화를 이루도록 설계하였다.

3.2 소유권 이전 프로토콜

새 소유권 이전 프로토콜은 그림 4와 같다. 등록 단계가 수행된 후, 소유권 이전 단계가 수행된다. 소유권 이전 단계는 아래와 같이 수행된다.

Step1: ($O_C \Rightarrow TTP$). 현 소유주(O_C)가 소유권을 새 소유주(O_N)에게 이전하기 위해 소유권 이전 요청 메시지(OID_C, OID_N, TID_i, C_C)를 TTP에게 전송한다. TTP는 메시지를 보낸 자가 실제 소유주임을 확인하기 위해 $C_C' = h(TTPID||OID_C||KO_i)$를 구하여 C_C와 비교한다. 만약, 같지 않으면 실제 소유주가 아니므로 세션을 끝낸다. 같으면 실제 소유주이므로 소유권 이전 작업을 수행한다. 태그의 식별자 TID_i를 사용하여 KT_i와 KO_i를 찾은 다음, 소유권 이전 이후에 사용할 새 $KO_i(K_{temp} = Nun(KO_i, KT_i))$와 새 $KT_i(KT_i^{new} = Nun(KT_i, TID_i))$를 계산한다. 다음으로 태그의 반응시간을 제어하기 위해 타이머를 시작하고, 소유권 이전 작업의 시작을 알림과 동시에 TTP의 진위를 나타내기 위해 난수 r_1을 생성하여 $V_1 = Nun(KT_i, r_1)$을 계산하여 메시지(V_1, r_1)을 태그(Tag_i)로 전송한다.

Step2: ($TTP \Rightarrow Tag_i$). 메시지(V_1, r_1)을 수신한 후, 이 메시지가 진짜 TTP에서 보낸 것인지 여부와 이 메시지가 소유권 이전을 요구하는 첫 번째 메시지인지를 구별하기 위해 $V_1' = Nun(KT_i, r_1)$을 계산하여 V_1과 비교한다. 같으면, 올바른 메시지이며 소유권 이전을 위한 첫 시도임을 나타낸다. 소유권 이전을 위해 KT_i와 KO_i를 변경시키고, DoS 공격이나 전송 에러로 인하여 소유권 이전이 완료되지 않을 경우를 대비하여 KT_i^{old}와 KO_i^{old} 변수에 이전 KT_i와 KO_i를 저장해 둔다. KT_i와 KO_i가 모두 변경되었음을 TTP에 알리기 위해 난수 r_2와 $V_2 = Nun(KT_i, r_2)$를 구하여 메시지 (V_2, r_2)를 TTP로 전송한다.

Step3: (Tag_i). Step2에서 V_1과 V_1'이 같지 않으면, 수신한 메시지가 잘못된 것이거나 소유권 이전을 위한 이전의 시도가 DoS공격 또는 전송 에러 등의 이유로 실패하였음을 나타낸다. 이를 확인하기 위해, $V_1'' = Nun(KT_i^{old}, r_1)$를 계산하여 V_1과 비교한다. 다르면, 잘못된 메시지이므로 아무 일도 하지 않는다. 같으면, 이전 소유권 이전 시도들이 실패한 것이므로 다시 소유권 이전을 위해 KT_i와 KO_i를 변경시킨다. 이 때, 기존의 KT_i와 KO_i값이 KT_i^{old}와 KO_i^{old}에 저장되어 있으므로 이를 사용한다. Step2와 마찬가지로 KT_i와 KO_i가 모두 변경되었음을 TTP에 알리기 위해 난수 r_2와 $V_2 = Nun(KT_i, r_2)$를 구하여 메시지 (V_2, r_2)를 TTP로 전송한다.

Step4: ($Tag_i \Rightarrow TTP$). TTP가 메시지 (V_2, r_2)를 수신하면 메시지의 진위 여부를 확인하기 위해 $V_2' = Nun(KT_i^{new}, r_2)$를 구하여 수신한 V_2와 V_2'이 같은지 확인한다. 같으면 올바른 태그가 보낸 진짜 메시지이므로 KT_i와 KO_i를 변경하고 새 소유주를 위한 소유증명서 $C_N = h(TTPID||OID_N)$을 계산한다. 같지 않으면, 태그 또는 메시지를 신뢰할 수 없으므로 Step1의 난수 생성(그림 4의 R)부터 다시 수행한다. 만약 타임아웃이 되면 <start timer>부터 작업을 다시 수행한다.

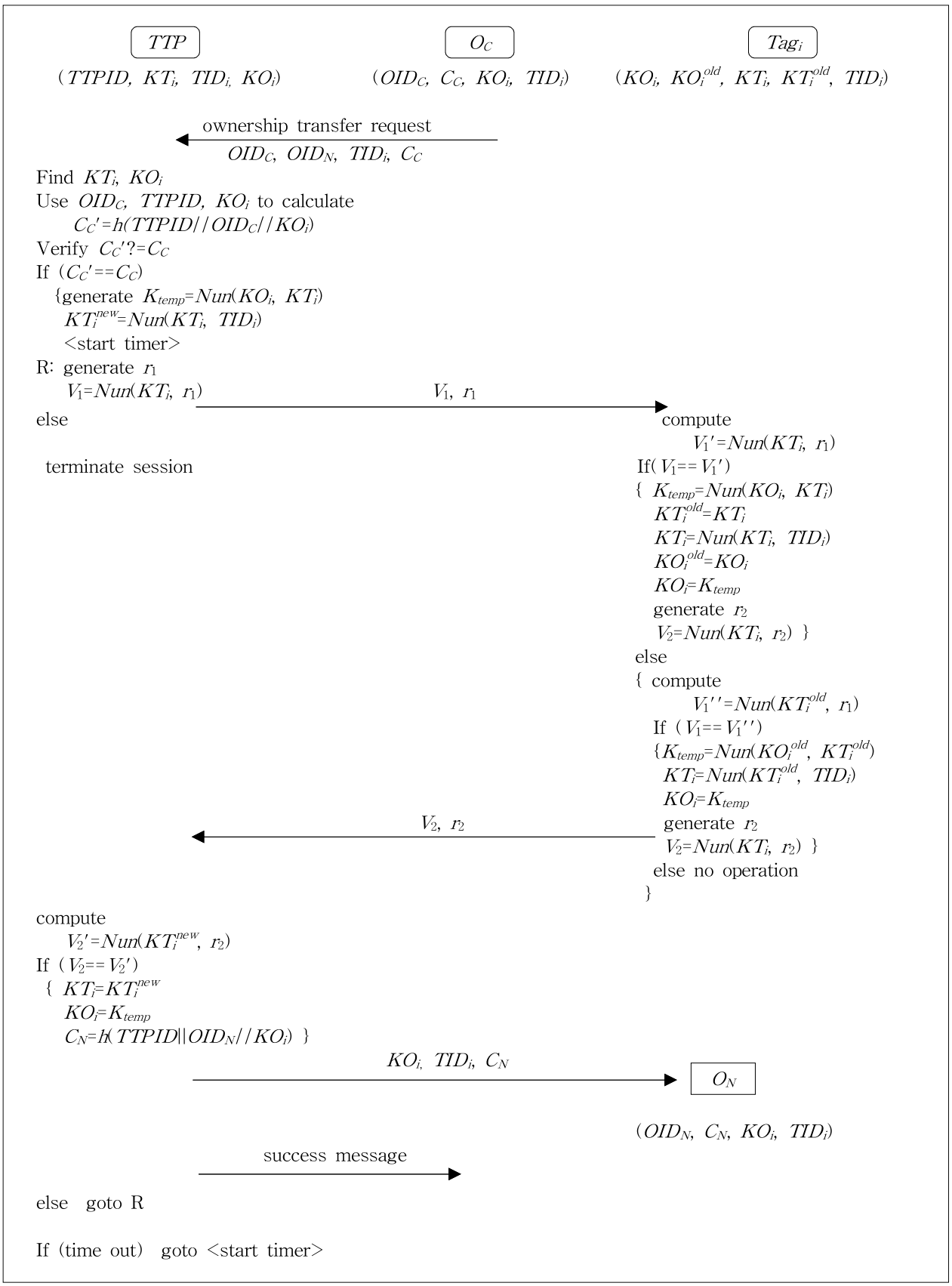

그림 4. 새 소유권 이전 프로토콜

Step5: ($TTP \Rightarrow O_N$). TTP는 새 소유주(O_N)에게 태그를 엑세스할 비밀키(KO_i), 태그의 식별자(TID_i) 그리고 소유증명서(C_N)으로 구성된 메시지(KO_i, TID_i, C_N)를 전송한다. 새 소유주는 비밀키(KO_i)를 사용하여 태그를 엑세스할 수 있으며 소유증명서(C_N)을 사용하여 소유권을 주장할 수 있다. 또한, TTP는 이전 소유주(O_C)에게 '성공' 메시지를 보내 소유권 이전이 성공적으로 이루어졌음을 알린다.

Step6: (TTP). 소유권 이전이 성공적으로 이루어지고 나면 다른 모든 TTP(백앤드 서버 포함)들에게 변경된 태그의 KT_i와 KO_i를 전송하여 동기화를 이룬다. 이 부분은 그림 4에는 제시하지 않았다.

4. 보안 요구 조건 분석 및 성능 분석

본 장에서는 3장에서 제안한 소유권 이전 프로토콜(이후 '수정 프로토콜'이라 함)이 2.1절에서 기술한 보안 요구 조건을 만족하는지를 분석하고, 2.5절에 기술한 문제점들을 해결함을 보인다. 또한, 태그와 서버에서 수행되는 연산들을 비교 분석한다.

4.1 보안 요구 조건 분석

새 소유주 프라이버시(New owner privacy): 수정 프로토콜에서는 TTP가 Tag_i에게 소유권 이전하도록 메시지를 보내면 KO_i를 변경한다. 변경된 KO_i는 KT_i를 알고 있어야만 구할 수 있다. 그러나, KT_i는 TTP와 Tag_i만 알고 있으므로 전 소유주 O_C는 알 수 없다. Tag_i에서 KO_i의 변경이 끝나면 TTP는 같은 방법으로 변경된 KO_i를 안전한 통신으로 새 소유주 O_N에게 전달한다. 따라서 전 소유자는 더 이상 태그를 접근 할 수 없고, 단지 새 소유주만이 접근할 수 있다. 그러므로 새 소유주의 프라이버시가 보호된다.

전 소유주 프라이버시(Old owner privacy): 새 소유주 O_N는 TTP로부터 변경된 KO_i, TID_i와 C_N만을 받는다. 따라서 O_N는 이전 소유자 O_C의 프라이버시를 보호할 수 있고, O_C와 Tag_i 사이의 트랜잭션들을 추적할 수 없다.

권한 복구(Authorization recovery): 이전 소유주에게 권한을 복구시키기 위해 다음과 같은 절차로 처리 가능하다. 먼저, 현 소유주가 TTP에게 권한 복구 요청을 한다. TTP는 이전 소유주에게 태그를 접근할 수 있는 비밀키(KO_i)를 전송한다. 이전 소유주는 이 비밀키로 태그를 접근할 수 있다. 사용이 끝나면, 이전 소유주는 TTP에게 사용이 끝났음을 알린다. TTP는 우리가 제시한 수정 프로토콜을 이용하여 소유권을 현 소유주에게 넘길 수 있다.

서비스 거부 공격에 안전(Resistance to Denial of Service (DoS) attack): 수정 프로토콜에서 DoS공격은 (V_1, r_1)메시지와 (V_2, r_2)메시지에서 이루어질 수 있다. 하지만 수정 프로토콜은 소유권 이전이 성공할 때까지 계속 수행되므로 DoS공격에 대해 안전하다. 만약, 수정 프로토콜을 변경하여 타임 아웃이 되면 O_C와 O_N에게 실패 메시지를 보내고 세션을 종료한다고 하면 동기화 문제가 발생한다. 첫 번째, (V_1, r_1) 메시지의 전송이 이루어지지 않도록 공격하는 경우를 생각해보자. 이 경우는 Tag_i의 키들(KO_i와 KT_i)이 변화되지 않았으므로 O_C가 기존 KO_i를 사용하여 태그에 접근할 수 있다. 두 번째, (V_1, r_1)메시지는 전송되었으나 (V_2, r_2)메시지의 전송이 이루어지지 않도록 공격하는 경우에는 Tag_i의 KO_i와 KT_i가 새로운 값으로 변경되었다. 이 때, O_C가 Tag_i를 기존 KO_i로 접근하려고 할 때 키가 달라 접근할 수 없다. 이를 위한 해결책이 필요하다. 그림 5와 같은 간단한 프로토콜을 추가하여 해결할 수 있다. O_C가 KO_i를 사용하여 Tag_i를 접근하려고 할 때 접근을 할 수 없으면 O_C는 TTP에게 새 키를 요구하는 메시지 (C_C, OID_C)를 전송한다. TTP는 C_C를 이용하여 태그의 소유주임을 확인한 후, KO_i와 KT_i를 수정 프로토콜에서 구한 방법과 같이 새로운 값으로 수정한 후, KO_i를 O_C

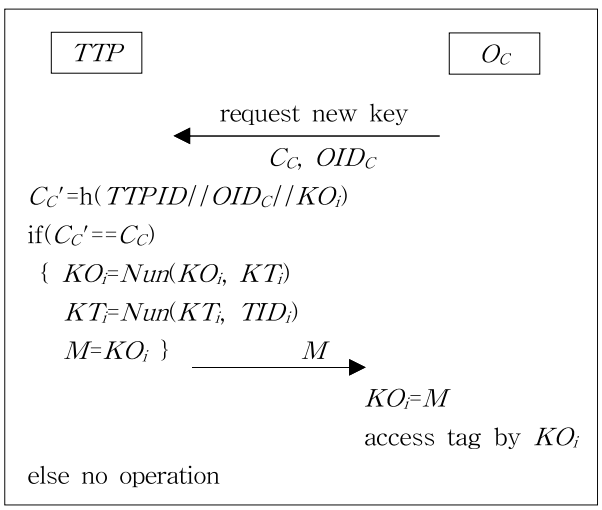

그림 5. TTP와 OC 간의 키 동기화

에게 전송한다. Oc는 수신한 KO_i를 새 KO_i로 하여 태그를 엑세스할 수 있다. 따라서, 수정 프로토콜은 서비스 거부 공격에 안전하다.

• 재생 공격에 안전(Resistance to replay attack): 수정 프로토콜에서 도청할 수 있는 메시지는 (V_1, r_1)과 (V_2, r_2) 2개이다. 먼저, 공격자가 (V_1, r_1)을 도청하여 이 메시지를 차후에 Tag_i에 전송한다고 가정하자. 이 메시지를 수신했을 때, Tag_i의 KO_i와 KT_i는 이미 변경되었고, KO_i^{old}와 KT_i^{old}는 이전 KO_i와 KT_i를 가지고 있다. 따라서, 공격자가 보낸 메시지 V_1에는 이전 KT_i(즉, KT_i^{old})의 정보만 들어 있다. 이 메시지를 받은 태그는 $V_1 == V_1''$임을 확인하여 응답 메시지(V_2, r_2)가 TTP에 전달되지 않은 것으로 판단하여 KT_i와 KO_i를 KT_i^{old}와 KO_i^{old}를 사용하여 다시 계산한다. 이렇게 새롭게 구해진 값은 이미 저장된 KO_i와 KT_i와 같다. 따라서 이 공격은 태그에 영향을 주지 않는다. 공격자가 메시지 (V_2, r_2)를 도청하여 차후에 이 메시지를 TTP에 전송한다고 가정하자. TTP가 이 메시지를 수신했을 때는 이미 세션이 종료되었으므로 TTP나 태그에 영향을 주지 않는다. 따라서, 수정 프로토콜은 재생 공격에 안전하다.

• 중간자 공격에 안전(Resistance to man-in-middle attack): 공격자들은 TTP와 태그 사이에 전송된 메시지를 도청하여 이 메시지를 수정하여 사용할 수 있다. 수정 프로토콜에서는 메시지가 수정되었는지를 쉽게 판별하기 위해 간단한 연산을 통해 구현된 Nun 함수를 사용한다. 공격자들이 도청한 메시지를 수정하여 TTP나 태그에 보냈을 때 수신자는 쉽게 이 메시지의 진위를 알 수 있다. 그 이유는 공격자들이 KT_i의 값을 알 수 없을뿐더러 Nun 함수로 계산된 값에서 KT_i의 값을 유추할 수 없기 때문이다. 따라서, 수정 프로토콜은 중간자 공격에 안전하다.

• 윈도잉 문제의 방지(Prevention of windowing problem): 수정 프로토콜에서는 소유권 이전 시, 태그를 접근하기 위한 키 KO_i를 태그 자체 내에서 새로운 키로 바꾼다. 이 새로운 키는 TTP가 새 소유주에게만 전달한다. 따라서 이전 소유주와 새 소유주가 동시에 태그를 접근할 수는 없다.

표 2는 여러 프로토콜들과의 보안 요구 조건을 비교 분석한 것이다.

4.2 기타 보안 사항

2.5절에 기술한 Lo 등이 제시한 프로토콜의 문제점들 중 '문제점1: 태그 키(KT_i)의 노출'은 수정 프로토콜에서는 발생하지 않는다. 2.5절에서 기술하였듯이 Lo 등의 프로토콜에서는 XOR로 사용하여 KT_i를 구할 수 있었지만, 수정 프로토콜에서는 $Nun(KT_i, r_1)$로 계산된 V_1을 메시지로 전송하므로 $Nun(KT_i, r_1)$ 값에서 KT_i를 구하는 것은 불가능하다[1].

표 2. 각 프로토콜들의 보안 요구 조건 분석

보안요구조건 / 프로토콜	NP	OP	RA	DoS	MA	WP
[8]	O	X	X	O	O	X
[4]	O	O	O	X	X	X
[5]	O	O	O	X	O	O
[10]	O	O	O	O	X	X
[6]	O	O	O	O	O	X
[11]	O	O	O	O	O	O
[9]	O	X	O	X	X	X
[7]	O	O	O	X	O	O
[1]	X	O	O	O	O	X
수정 프로토콜	O	O	O	O	O	O
NP: 새 소유주 프라이버시		OP: 전 소유주 프라이버시		RA: 재생 공격에 안전		
DoS: 서비스 거부 공격에 안전		WP: 윈도잉 문제의 방지		MA: 중간자 공격에 안전		
O: 성공		X: 실패				

'문제점2: 태그와 새 소유주 간의 공유 비밀 키(KO_i)의 노출' 역시 수정 프로토콜에서는 발생하지 않는다. 수정 프로토콜에서는 비밀키(KO_i)가 TTP, O_C 그리고 Tag_i에 저장되어 있고, 메시지에는 관련 정보가 노출되지 않는다. 따라서, 비밀키(KO_i)를 획득하기 위해서는 TTP, 소유주(O_C) 또는 Tag_i를 물리적으로 확보하여야 함으로 이는 불가능하다. 또한, KO_i는 $Nun(KO_i, KT_i)$로 변경되므로 KT_i를 알아야 변경된 KO_i를 알 수 있다. 하지만 위에서 언급한 바와 같이 KT_i는 노출되지 않으므로 변경된 KO_i를 알 수 없다.

수정 프로토콜에서는 소유권 이전이 끝났을 때, 소유증명서와 변경된 비밀키를 새 소유주에게 전달함으로 이전 소유주는 태그를 접근할 비밀키도 없고 소유증명서도 효력이 없다. 따라서, 이전 소유주가 여러 명에게 소유권을 이전할 수 없다. 즉, Lo 등의 프로토콜의 '문제점3: Fraud 공격에 취약'을 해결한다.

4.3 성능 분석

태그와 서버에서 수행되는 컴퓨팅 비용을 비교 분석 하였다. 표 3은 태그에서 수행되는 연산의 종류와 수행 횟수를 나타내며, 표 4는 서버(TTP)에서 수행되는 연산의 종류와 수행 횟수를 나타낸다. 일반적으로 대칭 암호화 연산은 해시함수나 Nun 함수에 비해 많은 컴퓨팅 자원과 시간을 요구한다. 우리가 사용한 Nun 함수는 덧셈과 비트 시프트 연산을 사용한 것으로 그 기능은 일방향 해시함수와 같은 기능을 수행하지만 해시함수보다 컴퓨팅 비용은 작다[1].

표 2에서 보안 요구조건을 모두 만족하는 Kapoor와 Piramuthu가 제시한 프로토콜[11]과 우리가 제시한 수정 프로토콜을 비교해 보자. 표 3과 표 4에 나타난 바와 같이 [11]의 프로토콜은 태그와 서버에서 많은 암호화 연산뿐만 아니라 여러 연산을 많이 수행한다. 수정 프로토콜에 비해 훨씬 많은 컴퓨팅 비용이 요구된다. [4], [8]과 [9]의 프로토콜은 대체적으로 적은 컴퓨터 비용이 요구되지만 표 2에서 지적한 것처럼 보안에 취약하다.

5. 결 론

RFID 기술은 공급망 관리를 위한 산업에 광범위하게 채택되어 사용되고 있다. 제품이 생산될 때 각 제품에 RFID 태그가 부착되며, 공장, 배급자, 소매상

표 3. 태그에서 수행되는 연산

	[8]	[4]	[5]	[10]	[6]	[11]	[9]	[1]	수정 프로토콜
XOR 연산	0	1	5	2	9	6	4	7	0
일방향 해시함수	0	0	0	2	6	2	2	0	0
난수 생성	0	0	1	0	2	2	3	2	1
대칭 암호화 연산	1	0	2	0	0	2	0	0	0
시프트 연산	0	0	0	0	4	0	0	0	0
Nun 연산	0	0	0	0	0	0	0	4	4

(숫자는 처리되는 연산의 횟수임)

표 4. 서버(TTP)에서 수행되는 연산

	[8]	[4]	[5]	[10]	[6]	[11]	[9]	[1]	수정 프로토콜
XOR 연산	0	1	5	2	10	12	4	6	0
일방향 해시함수	0	0	0	3	5	4	0	1	2
난수 생성	0	0	1	0	3	3	4	4	1
대칭 암호화 연산	1	1	2	1	0	5	0	0	0
시프트 연산	0	0	0	0	4	0	0	0	0
Nun 연산	0	0	0	0	0	0	0	2	4

(숫자는 처리되는 연산의 횟수임)

및 소비자 사이에서의 공급망 관리를 위해 제품의 소유권 이전이 주의 있게 처리되어야 한다. 이와 같이 RFID 기술을 사용하여 각 제품을 식별하고 제품의 공급과정을 효율적으로 처리하기 위해 안전하고 효율적인 RFID 소유권 이전 프로토콜이 중요한 이슈이다.

이미 많은 소유권 이전 프로토콜이 제안되었다. 이들 프로토콜들은 보안상의 문제를 가지고 있으며, 대부분의 프로토콜은 암호화 연산 및 해시함수 같은 복잡한 연산 즉, 태그 및 서버에서 수행할 연산의 양이 많아진다. 이것은 태그의 설계를 복잡하게 하여 게이트 수가 증가하므로 태그의 비용이 비싸진다. Lo 등은 간단한 연산들(시프트, 덧셈, XOR 연산 및 난수 생성)을 사용한 프로토콜을 제시하였다. 하지만, 이 프로토콜은 태그와 새 소유주가 공유하는 비밀키를 공격자가 획득할 수 있는 문제점과 Fraud 공격에 취약하다.

본 논문에서는 Lo 등이 제시한 프로토콜을 수정하여 간단한 연산들(시프트, 덧셈 연산 및 난수 생성)을 사용하여 보안 공격에 안전한 새로운 프로토콜을 제시하였다. 우리가 제시한 프로토콜은 보안 요구 조건을 모두 만족하며, Lo 등이 제시한 프로토콜의 문제점들을 모두 해결함을 보였다.

참 고 문 헌

[1] N.W. Lo, S.H. Ruan, and T.C. Wu, “Ownership Transfer Protocol for RFID Objects Using Lightweight Computing Operators,“ *The 6th International Conference on Internet Technology and Secured Transactions*, pp. 484-489, 2011.

[2] M. Safkhani, N Bagheri, M Naderi, and A. Mahani, “On the Security of Lo et al.'s Ownership Transfer Protocol,” *Cryptology ePrint Archive*, Report 2012/023, 2012.

[3] 배석태, “역 공급사슬관리를 위한 RFID 기반의 서비스 플랫폼 적용에 관한 연구,” 멀티미디어학회논문지, 제12권, 제11호, pp. 1671-1679, 2009.

[4] K. Osaka, T. Takagi, K. Yamazaki, and O. Takahashi, "An Effcient and Secure RFID Security Method with Ownership Transfer," *International Conference on Computational Intelligence and Security*, pp. 1090-1095, 2006.

[5] S. Fouladgar and H. Afifi, "An Efficient Delegation and Transfer of Ownership Protocol for RFID Tags," *The First International EURASIP Workshop on RFID Technology*, pp. 59-62, 2007.

[6] S. Song, "RFID Tag Ownership Transfer," *Workshop on RFID Security*, 2008.

[7] Y. Zuo, "Changing Hands Together: A Secure Group Ownership Transfer Protocol for RFID Tags," *Hawaii International Conference on System Sciences*, pp. 1-10, 2010.

[8] J. Satio, K. Imamoto, and K. Sakurai, "Reassignment Scheme of an RFID Tag's Key for Owner Transfer," *Lecture Notes in Computer Science*, Vol. 3823, pp. 1303-1312, 2005.

[9] L. Kulseng, Z. Yu, Y. Wei, and Y. Guan, "Lightweight Mutual Authentication and Ownership Transfer for RFID Systems," *IEEE INFOCOM*, pp. 251-255, 2010.

[10] H. Lei and T. Cao, "RFID Protocol enabling Ownership Transfer to Protect Against Traceability and DoS Attacks," *The First International Symposium on Data, Privacy, and E-Commerce,* pp. 508- 510, 2007.

[11] G. Kapoor and S. Piramuthu, "Single RFID Tag Ownership Transfer Protocols," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, Vol. 42, No. 2, pp. 164-173, 2012.

[12] C.L. Chen, Y.L. Lai, C.C. Chen, Y.Y. Deng, and Y.C. Hwang, "RFID Ownership Transfer Authorization Systems Conforming EPCgobal class-1 Generation-2 Standards," *International Journal of Network Security*, Vol. 13, No. 1, pp. 41-48, 2011.

이 재 동

1983년 2월 서울대학교 계산통계학과 이학사
1985년 2월 서울대학교 전산과학전공 이학석사
1995년 2월 서울대학교 전산과학전공 이학박사

1986년~현재 경남대학교 컴퓨터공학과 교수
관심분야 : 실시간 시스템, 임베디드 시스템, 컴퓨터 보안