情 報 保 護 學 會 誌
第23卷 第5號, 2013. 10

TLS 안전성 연구 : 최근 Crypto 연구 결과를 중심으로

변 진 욱*

요 약

TLS 프로토콜은 인터넷 보안에서 가장 널리 사용되는 핵심 기술로서, 핸드쉐이크(handshake) 프로토콜과 레코드 레이어(record layer) 프로토콜로 구성된다. 핸드쉐이크에서 형성된 키를 이용하여 레코드 레이어 프로토콜에서 데이터들에 대해 인증, 무결성, 기밀성과 같은 정보보호 서비스를 제공한다. 이와 같이, TLS 프로토콜은 키 공유라고 하는 고전적인 암호학 문제를 포함하고 있기 때문에, 암호학 연구자들에게 접근성이 매우 뛰어난 연구 주제가 되어 왔다. 이를 반영하듯, 최근에 암호학 분야의 최고 권위 있는 Crypto학회에 TLS 이론 연구에 관한 논문이 연속적인 주제로 게재되었다. 본 논문에서는 최근 Crypto학회 및 eprint에 올라온 TLS와 관련된 최신 연구 동향 및 결과를 분석하여, TLS 프로토콜의 안전성을 연구하였다.

Ⅰ. 서 론

TLS 프로토콜은 인터넷 보안에서 가장 널리 사용되는 핵심 기술로서, 핸드쉐이크(handshake) 프로토콜과 레코드 레이어(record layer) 프로토콜로 구성된다. 핸드쉐이크 프로토콜의 주된 목적은 참여자간의 안전한 암/복호화 키와 MAC(message authentication code) 키를 안전하게 공유하기 위함이다. 반면에, 레코드 레이어 프로토콜은 응용단(application layer)에서 내려온 데이터 패킷을 쪼개어 핸드쉐이크에서 만들어진 키들을 이용해 암호화 하고 MAC 값을 추가하여 TCP단으로 내려 보내는 것이 목적이다. 다시 말해, 레코드 레이어에서 실제적인 보안 서비스가 이루어진다.

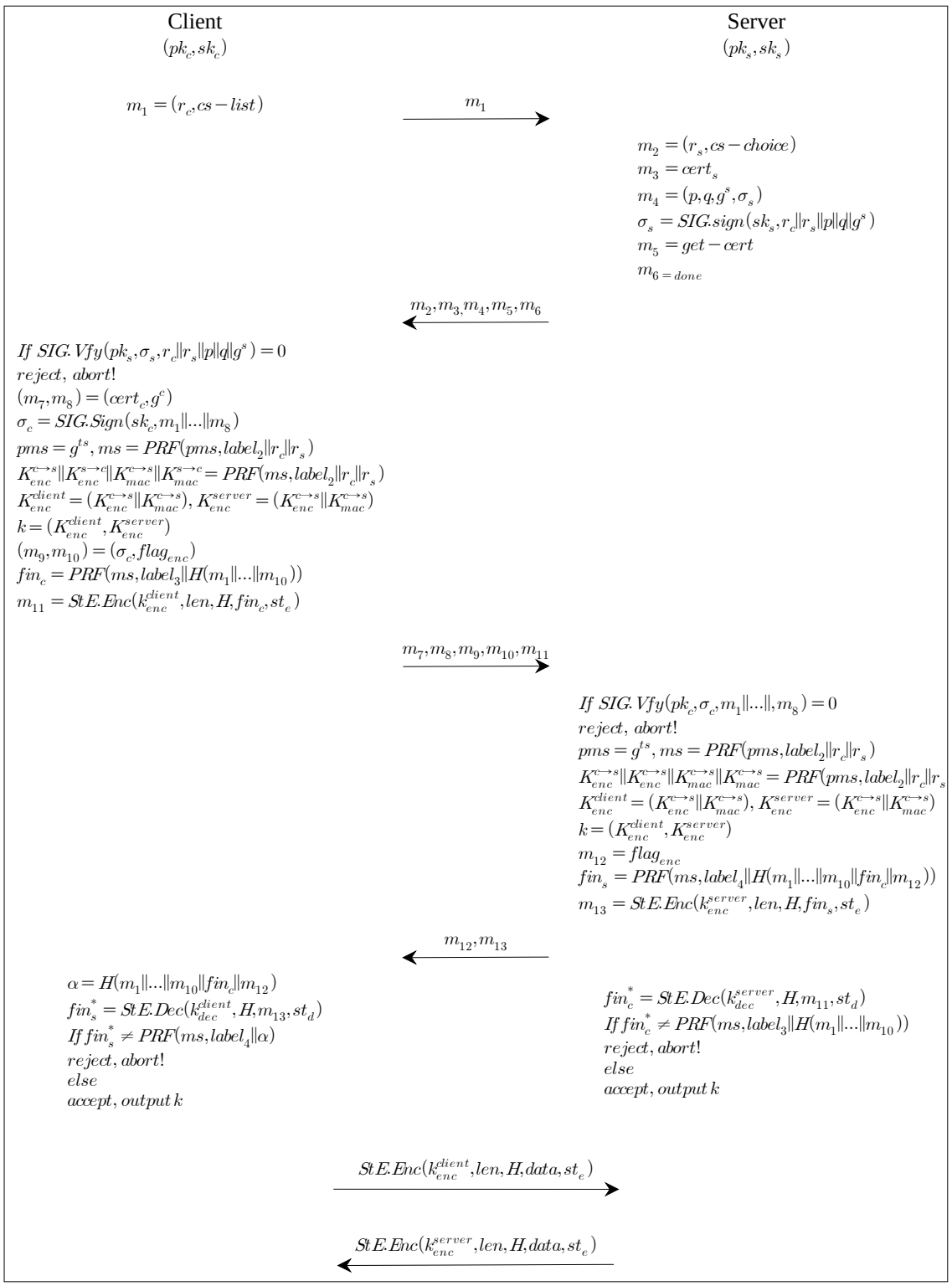
이 처럼, TLS 프로토콜은 키 공유라고 하는 고전적인 암호학적 문제를 포함하고 있기 때문에, 암호학 연구자들에게 접근성이 매우 뛰어난 연구 주제가 되어 왔다. 이와 더불어, TLS 프로토콜은 인터넷 상에서 안전한 채널을 형성하기 위한 사실상 표준 프로토콜이므로, 실제 표준 프로토콜 분석이라는 관점에서, 실제적인 공격 및 관련 이론 연구가 활발히 진행되었다. 이를 반영하듯, 최근에 암호학 분야의 최고 권위 있는 Crypto학회에 TLS 이론 연구에 관한 논문이 연속적인 주제로 게재되었다[4, 6]. 이 연구 결과들은 예전에 제기되었던 TLS에 관한 이슈들을 해결하였으며, 아직 해결하지 못한 연구 주제를 제시하기도 하였다.

본 논문에서는 최근 Crypto학회의 연구 결과와 이와 관련되어 eprint에 올라온 TLS의 최신 연구 동향을 파악하고, 관련 이슈를 분석하여, 향후 연구 주제를 간략히 제시한다. 우선, 최근에 Crypto학회 및 eprint에 제시된 연구결과들을 [1, 4, 5, 6, 7, 8] 통해 최근 이슈를 분석하고 이를 바탕으로 관련 연구 결과를 [4, 6] 요약하여 정리한다. 본 논문에 분석된 내용은 참고문헌 [4, 6] 연구 결과 및 분석 결과에 근거해서 작성되었으며, 이를 바탕으로 향후 연구 주제를 제시하였다.

Ⅱ. TLS 프로토콜

TLS 프로토콜은 총 3가지 모드를 이용해 동작한다. 첫 번째는 가장 널리 사용되는 TLS-RSA 모드로서, 사용자가 직접 암복호화 키를 생성하여 서버에게 안전하게 전달한다. 암복호화 키 전달 시, RSA PKCS #1v1.5 암호화 방식이 핸드쉐이크 프로토콜에서 사용된다. 두 번째는 TLS-DH 모드로서 Diffie-Hellman (DH) 키 교환 방식으로 암복호화 키를 생성하되, 서버의 정적(static) DH 키와 사용자의 일회성 (ephemeral) DH 키를 이용해서 암복호화 키를 만든다. 세 번째는

* 평택대학교 정보통신학과 (jwbyun@ptu.ac.kr)



〔그림 1〕 TLS 프로토콜 (핸드쉐이크, 레코드 레이어 프로토콜)

TLS-DHE 모드로서, TLS-DH 모드와 동일하나, 사용자와 서버 모두 일회성 DH 키를 이용해서 암복호화 키를 생성하게 된다. 아래 그림은 TLS-DHE 모드에 대한 설명으로서, 참고문헌 [4]에 설명되어 있는 내용을 그림으로 다시 정리하여 나타내었다.

2.1. 프로토콜 설명

[그림 1]에 사용된 표기 설명은 다음과 같다. 먼저, PRF은 안전한 pseudo random 함수이고, H는 암호학적으로 안전한 해시함수이다. 연구 결과에 따라 증명 시 랜덤 오라클 (random oracle) 로도 동작한다. $Sig.Sign$과 $Sig.Vfy$는 각각 전자서명 생성 및 검증알고리즘이다. $StE.Enc, StE.Dec$는 레코드 레이어에서 암복호화에 사용되는 SLHA(stateful length hiding authenticated encryption) 암호이다 [4,6,9]. 이에 대한 정의는 부록에 제시하였다. SLHA암호는, 참고문헌 [9]에, 메시지의 변조, 재생공격, 재배치(reordering)공격에 안전하도록 처음 소개되었으며, TLS 레코드 레이어 프로토콜의 강화된 안전성을 정의할 수 있도록 설계되었다. 참고문헌 [9] 결과에 의하면, CBC 기반의 TLS의 레코드 레이어는 SLHA의 안전성 정의를 만족시킴을 보였다. 마지막으로, 핸드쉐이크를 통해 만들어지는 키에 대한 표기를 살펴보면, 총 두 종류의 키가 있다. 첫째는 사용자를 위한 암복호화 키 쌍, $K_{enc}^{c\to s}, K_{enc}^{s\to c}$ 과 mac 코드의 생성 및 검증을 위한 키 쌍, $K_{mac}^{c\to s}, K_{mac}^{s\to c}$이 존재한다. 이에 대응되는 서버의 암복화 키 쌍은 $K_{enc}^{s\to c}, K_{enc}^{c\to s}$이고, mac 키 쌍은 $K_{mac}^{s\to c}, K_{mac}^{c\to s}$이다. 이를 이용해, 사용자의 공통된 키 k를 정의하였다. [그림 1]의 설명을 간략히 살펴보면 다음과 같다.

- 먼저, 사용자는 랜덤 값 r_c값을 생성하고 사용 가능한 암호스펙 (ciphersuits), $cs-list$를 작성하여, $m_1 = (r_c, cs-list)$ 메시지를 서버에게 보낸다.
- 서버는 $m_2,...,m_6$를 [그림 1]의 내용처럼 구성하여 사용자에게 보낸다. 가장 중요한 부분은 서버의 DH 값인 g^s 값에 대해서 자신의 개인키 sk_s로 서명한 값 인 $\sigma_s = SIG.sign(sk_s, r_c \| r_s \| p \| q \| g^s)$를 포함하고 있는 m_4 메시지이다. 이는 이후 서버의 인증을 위해 필요하다.
- 사용자는 $m_2,...,m_6$ 메시지를 받은 후 제일 먼저, m_4에 대한 서명 검증 작업을 먼저 수행한다. 사용자 역시 동일하게 자신의 DH 값 g^c에 대한 서명 $m_9 = \sigma_c$를 자신의 개인 키, sk_c를 이용해 만들고 이후 서버가 이를 이용해 사용자 인증을 수행한다. 이때 사용자가 $pms = g^{cs}$값을 만들며, 이를 이용해 마스터 키 값인 $ms = PRF(pms, label_2 \| r_c \| r_s)$를 생성할 수 있다. 더 나아가 ms값을 이용하여, 사용자의 암 복호화를 위한 키들과 mac(message authentication code)을 위한 키 값들, $K_{enc}^{c\to s} \| K_{enc}^{s\to c} \| K_{mac}^{c\to s} \| K_{mac}^{s\to c}$ 을 만든다. 사용자는, 종료 메시지 $fin_c = PRF(ms, label_3 \| H(m_1 \| ... \| m_{10}))$를 암호화한 값인, m_{11} 메시지를 만들어 $m_7,...,m_{10}$ 값과 함께 서버에게 전달한다.
- 서버 역시, 사용자의 서명 σ_c를 우선 검증한다. 그 후, pms, ms값을 사용자와 동일한 방법으로 계산한다. pms를 통해 만들어진 ms값을 이용해, $fin_s = PRF(ms, label_4 \| H(m_1 \| ... \| m_{10} \| fin_c \| m_{12})$를 계산하고 이를 암호화 한 m_{12} 메시지를 사용자에게 전달한다. m_{11}은 암호에 필요한 플래그 값이다. 물론, 서버는 사용자의 마지막 메시지인, m_{11}메시지를 복호화하여 fin_c값을 얻고, 자신이 계산한 fin_c^*값과 동일한지 검증한다.
- 사용자는 끝으로, m_{13}메시지로부터 fin_s값을 복호화하고 본인이 만든 fin_s^*값과 동일한지 검증한다.

Ⅲ. TLS 프로토콜의 이슈 연구

TLS 프로토콜은 크게 두 가지 측면에서 이슈가 되어 왔다. 첫째는 안전성 모델 및 증명과 관련된 문제이다. 둘째는 재협상과 관련한 TLS 프로토콜의 이슈이다. 이를 세부적으로 정리하여 설명하면 다음과 같다.

3.1. 기존의 AKE 안전성 모델로는 TLS 안전성을 정의하기에 부족하다

TLS는 핸드쉐이크 프로토콜과 레코드 레이어 프로토콜 두 개로 구성되어 있다. 그러므로 키 구별 불가능성으로 (key indistinguishability) 정의된 기존 AKE (authenticated key exchange) 안전성 모델을 이용하여,

TLS 프로토콜에 포함된 두 개의 서브 프로토콜 모두의 안전성 정의를 논하는 것은 분명 한계가 있으며, 이 문제를 어떻게 해결할 것인가라는 이슈는, 최근 암호학 이론 분야에서 주된 관심사였다. 다시 말하면, 기존의 안전성 모델로는 TLS 프로토콜의 안전성을 보장할 수 없다는 점이 더 큰 문제점이었다.

기존의 키 구별 불가능성 안전성 모델을 그대로 TLS로 적용시킨다고 해보자. TLS 프로토콜에서는 마지막에 finished 메시지를 TLS에서 형성된 키로 암호화에서 보낸다. 공격자는 test 질의를 받을 때 마다, 추측에 성공해야 할 b (세션 키 sk_b) 에 대해서, 마지막 메시지를 복호화 한 값을 이용해 계산한 값이, finished 메시지인지를 검증함으로써, 실제 (real, b=1) 키 인지 랜덤한 (random, b=0) 키 인지를 구분할 수 있게 된다. 구체적으로 설명하면, 공격자는 안전성 정의로부터 주어진 sk_b값을 이용해, [그림 1]의 m_{13}메시지를 복호화해서, $(len', H', fin_s', st_e')$ 메시지를 얻게 된다.

$$m_{13} = StE.Enc(k_{enc}^{server}, len, H, fin_s, st_e)$$
$$fin_s = PRF(ms, label_4 \| H(m_1 \| ... \| m_{10} \| fin_c \| m_{12}))$$

공격자는 메시지들$m_1\|...\|m_{10}$, 공개된 값들, fin_c값을 PRF에 질의해 fin_s' 답변을 얻을 수 있고 이 값과 fin_s 값을 비교한다. 만약, 값이 일치하면, 실제 키로 간주하고, 일치하지 않으면, 랜덤 키로 간주하게 된다. 굳이, fin_s값을 비교하지 않더라도, 복호화된 값이 공개된 len, H 값들과 같은지만 비교하더라도 실제 키와 랜덤 키의 구분이 가능하다.

3.2. TLS 프로토콜을 수정하면 기존 AKE 안전성 모델로 충분하다 [4]

그렇다면, 위의 안전성 문제를 해결할 수 있는 방안은 무엇인가? 가장 손쉬운 방법은, TLS 프로토콜 중 finish 메시지와 관련되어 있는 m_{11}과 m_{13}메시지를 수정하는 것이다. 이렇게 되면, 새로운 안전성 모델이 필요없고, 기존의 AKE 안전성 모델로 충분히 증명가능하다. 사실, 마지막에 서버가 finished 메시지를 보내는 것은 중간자 공격을 방어하기 위함이다. 중간자 공격은, finished 메시지를 암호화하지 않더라도, 상호 인증만 정확히 이루어진다면 방어 가능하다. 위 [그림 1]에서, 메시지 m_{11}을 암호화형태가 아닌, 평문의 형태로 보내도록, m_{11}, m_{13}을 다음과 같이 설계할 수 있다.

$$fin_c = PRF(ms, label_3 \| H(m_1 \| ... \| m_{10}))$$
$$fin_s = PRF(ms, label_4 \| H(m_1 \| ... \| m_{10} \| fin_c \| m_{12}))$$

이렇게 부분적으로 m_{11}, m_{13}메시지를 수정한 TLS 프로토콜은 기존의 AKE 모델에서 안전하다라고 증명되었다 [4].

TLS 프로토콜을 수정하는 아이디어는 사실, 참고문헌 [7, 8] 연구 결과에서 처음 시도되었다. 즉, 2008년에 Morrissey, Smart, Warinschi는 TLS-RSA 프로토콜에 대해서 finished 메시지를 암호화 하지 않고 전달하는 truncated TLS 프로토콜을 제안하고 그 안전성을 증명하였다. 하지만, 그들은, RSA 암호화 방식을 CCA에 안전한 암호 방식으로 대체한다는 가정을 사용하였는데, 이는 현실적이지 못한 가정이다. 왜냐하면, 현재 TLS 프로토콜의 표준 암호화 방식인 RSA PKCS #1v1.5은 CCA에 안전한 암호 방식이 아니기 때문이다. 더 나아가, 제안된 truncated TLS 프로토콜은 실제 TLS 프로토콜이 아니므로, 전 세계에 널리 사용되고 설치되어 있는 TLS 프로토콜을 바꾸어 실제적으로 적용되기에 비용, 정책적인 측면에서 현실성이 떨어진다.

3.3. TLS 프로토콜을 위한 새로운 안전성 모델 제안과 표준적 가정(standard assumption) 하에서의 증명

2012년에, Jager, Kohlar, Schage, Schwenk는 TLS 전체 프로토콜 위한 새로운 안전성 모델인 ACCE (authenticated and confidential channel establishment)를 처음으로 제안하였다. 그리고 TLS-DHE 프로토콜이 ACCE 안전성을 만족함을 증명하였다 [4]. 참고문헌 [4]에서 밝혔듯이, 한 가지 이슈는, TLS-DHE를 표준적인 가정만을 이용하여 ACCE 모델 하에서 증명할 수 있는가였다. 참고문헌 [4]의 주요 결과는, TLS-DHE 프로토콜을, 표준 가정이 아닌, PRF-ODH(PRF oracle diffie-hellman) 가정을 이용하여 TLS-DHE가 ACCE 모델 하에서 안전함을 보였다는데 있다.

먼저, PRF와 PRF-ODH의 차이점에 대해 살펴보자. PRF의 안전성은 $z_0 = PRF(k, x)$와 유니폼하게(uniformly)

선택된 $z_1 \leftarrow \{0,1\}$을 공격자가 구분하되, 공격자는 질의 x_i에 대해, 답변 $z_i = PRF(k, x_i)$을 z_0, z_1 구분에 활용한다. 이에 반해, PRF-ODH는 $z_0 = PRF(g^{uv}, m)$와 유니폼하게 선택된 $z_1 \leftarrow \{0,1\}$을 공격자가 구분하는 것이 목적이되, 공격자는 $X \neq g^u$인 (X, m')을 질의하면, $PRF(X^v, m')$을 답변으로 받아 z_0, z_1 구분에 활용할 수 있다. (u, v는 순환 군에서 랜덤하세 선택된 원소이다.)

문제는, 이러한 PRF-ODH를 사용하지 않고, 오직 DDH 가정만을 이용하여 TLS-DHE 프로토콜을 ACCE 모델 하에서 안전성을 증명할 수 있는가이다. 증명 가능성 여부를 분석하기 위해서는 TLS-DHE 프로토콜의 fin_s 메시지의 구조를 먼저 살펴보아야한다. 참고문헌 [4]에 잘 설명되었듯이, 다음의 fin_s 메시지는 아래처럼 구성된다.

$$fin_s = PRF(ms, m_1 \| ... \| m_3)$$
$$ms = PRF(g^{cs}, label_1 \| r_c \| r_s)$$

만약, DDH로 안전성 증명을 귀결시킨다고 가정해보자. 그렇다면, 게임 기반의 안전성 증명 마지막 부분에, $pms = g^{cs}$를 랜덤한 값으로 대체해서 두 게임의 이점 차가 DDH 이점으로 귀결됨을 보이는 과정이 필요하다. 여기서, DDH 공격자가 입력 값 g^u, g^v에 대해 g^{uv}와 랜덤 값 구별을 시도하려 할 때 도전자(challenger)는 fin_s 메시지를 공격자에게 시뮬레이션 해야 한다. 이를 위해서는 우선 ms값을 우선 계산해야하는데, ms는 입력 받은 랜덤 값을 이용하면 두 게임의 차를 DDH 이점으로 귀결시킬 수 있다.

위 경우는 수동적인 공격자만을 가정했을 경우이다. 하지만, 입력 받은 g^c, g^s에 대해 변경을 할 수 있는 능동적인 공격자의 경우에는 공격자가 쉽게 두 게임을 구분할 수 있게 되어서 DDH 이점으로 귀결시키는 것이 어렵게 된다. 가령, 공격자가 $m_1, ..., m_6$메시지를 사용자와 서버에게 그대로 전달하고, 서버에게 전달되는 $m_7, ..., m_{11}$메시지 중에, m_7, m_8, m_9메시지를 각각 $m_7' = cert_{c'}$, $m_8' = g^{c'}$, $m_9' = \sigma_{c'}$로 변경하여 전달한다고 하자. c'은 현재 프로토콜에 참여중인 사용자가 아닌 다른 임의의 사용자이며 m_9'은 그 사용자의 개인키로 서명한 값이다. 이 개인키는 corrupt질의로 얻을 수 있으며 프로토콜에 참여하는 사용자에 대한 corrupt질의가 아니므로 안전성 정의에 위배되지 않는다. 이러한 변경이 가능한 근본적인 이유는, TLS 프로토콜에서, $m_7, ..., m_{11}$메시지를 전달하는 단계까지, 사용자에 대한 익명성을 유지하기 때문이다. 이러한 경우에 공격자는 서버로부터 m_{13}메시지를 받게 되고, 공격자는 pms값을 알고 있기 때문에 복호화 키를 계산해 m_{13}메시지를 복호화한 후 fin_s값을 알 수 있다. 이 값을 본인이 만든 값과 비교함으로써 서버가 g^s를 알고 있는지를 검증할 수 있게 된다. 만약, 본인이 만든 값과 동일하다면, 이는 도전자가 랜덤 값이 아닌 g^{cs}값으로 시뮬레이션하고 있음을 공격자가 알 수 있다. 동일하지 않다면, 공격자는 해당 게임이 서버도 모르는 랜덤 값으로 자신에게 시뮬레이션하고 있음을 눈치챌 수 있다. 이로 인해, 능동적인 공격자는 두 게임의 차를 DDH 이점보다 더 큰 확률로 충분히 구분할 수 있다.

하지만, DDH 대신, PRF-ODH라는 가정을 사용하여 ms를 계산하도록 프로토콜을 설계할 경우에는, 공격자는 서버가 g^s를 알고 있는지 검증할 수 있는 방법이 없다. 왜냐하면, PRF-ODH 가정은 공격자가 $X \neq g^c$인 (X, m')을 질의하면, $PRF(X^s, m')$ 값만을 답변으로 받기 때문이다. 그러므로, PRF-ODH의 오라클 답변 자체를 ms를 계산할 때 사용하면 된다.

그렇다면, DDH 가정을 이용하여 증명할 수 있는 방안은 없는가? 이를 위해서는 위에서 묘사한 능동적 공격자의 게임 구별에 대한 공격을 방어할 수 있어야 한다. 가장 간단한 방법은 공격자에게 corrupt질의를 못하게 하여, $m_7' = cert_{c'}$, $m_8' = g^{c'}$, $m_9' = \sigma_{c'}$메시지를 서버에게 전달 할 수 없게 하는 방안이 있다 [4]. 즉, 참여자가 아닌 다른 사용자들의 롱텀 키를 얻지 못하도록 corrupt 질의를 원천적으로 봉쇄하는 방안이다. 이는 안전성 모델을 크게 약화시킨다라는 단점이 있지만, TLS 프로토콜 자체에 대한 변형이 없다는 장점도 있다.

사용자와 서버의 아이디를 랜덤 넌스 (nounce) 값, r_c, r_s에 직접 넣는 방안도 있다 [4]. r_c, r_s는 이후에 사용자와 서버에 의해 서명되므로 위의 능동적 공격을 방어할 수 있다. 하지만, 이 방법은 기존 TLS 표준 프로토콜을 변형한 형태이고, 모든 적용된 프로토콜을 변경해야 하므로 프로토콜 실제 적용에 어려움이 있다는 단점이 있는 반면에, 표준적인 가정 (DDH) 을 이용하여 증명 가능하다는 장점을 지닌다.

3.4. TLS-DHE를 제외한 다른 모드에 대한 연구 이슈

참고문헌 [4] 연구 결과는 TLS-DHE에 대한 결과이다. 실용적인 관점에서, TLS-DHE는 TLS-RSA 모드에 비해 실제적인 사용빈도가 높지 않다. 하지만, 참고문헌 [4]는 TLS를 위한 안전성 모델을 처음으로 제안하였고, TLS-DHE 모드의 안전성 증명을 처음으로 시도하였다는 점에서 공헌도가 높다. 이후, 나머지 두 개의 모드 TLS-RSA, TLS-DH에 대해서 참고문헌 [4]에서 제안된 ACCE 모델 하에서 안전성 증명 연구가 이루어졌다. 흥미로운 사실은, 총 2개의 연구가 [6][5] 거의 비슷한 시기에 수행되어, 올해 2013 Crypto학회에 제출되었지만, 참고문헌 [6] 결과만 게재되었다.[1)]

3.5. TLS 재협상에 관한 이슈

핸드쉐이크 프로토콜이 완료되고 안전한 통신이 레코드 레이어에서 이루어진 후에, 사용자는 언제든지 프로토콜의 재협상을 요구할 수 있다. TLS 프로토콜의 재협상은 새로운 세션 키 (암복호화키, 인증키 포함)를 만들거나 혹은 암호학적 파라미터 혹은 알고리즘들을 변경하고 싶을 때 서버와 사용자에 의해 각각 요청된다. 사용자는 ClientHello 메시지를 현재의 레코드레이어에서 현재의 암호 키로 암호화해서 서버에게 요청할 수 있고, 서버는 HelloRequest 메시지를 레코드레이어에서 사용자에게 보내고 (trigger의 역할), 이를 받은 사용자는 새로운 ClientHello 보내면서 새로운 재협상을 수행하게 된다.

2009년에 Ray와 Dispensa는 처음으로 TLS 재협상에 대한 문제점을 지적했다 [10]. 사용자 Alice, 서버 Bob, 공격자 Eve를 가정하자. Alice와 Bob이 TLS 세션을 맺으려할 때 Eve는 Alice가 보내려는 초기 ClientHello메시지를 보내지 않고, Eve 자신이 Bob과 TLS 세션을 직접 맺는다. 그리고 Eve 자신의 m_0메시지를 Eve-Bob간의 레코드 레이어에 전달한다. 이후, Eve는 Alice의 초기 ClientHello 메시지를 Eve-Bob간의 레코드 레이어에 전달한다. 그러면, TLS 프로토콜은 Alice에 의한 재협상 요청으로 인식하고 Alice와 Bob간의 핸드쉐이크가 수행된다. (그러므로 Eve는 이때의 Alice와 Bob간의 레코드 레이어에는 접근할 수 없다.) 그 후 Alice는 자신의 메시지 m_1을 Alice-Bob간의 레코드 레이어에 전달한다.

문제는 Bob이 https, smtps와 같은 응용 환경에서, m_0,m_1을 연결하여 같은 메시지 $m_0 \| m_1$로 인식한다는 점이다. 그 이유는 Eve가 m_0메시지를 보내고, 이후 ClientHello 메시지를 보내어 재협상이 이루어지므로 같은 메시지의 묶음으로 인식하게 된다. 비록, ClientHello 메시지는 Alice에 의해 만들어진 것이고 이후 협상이 Alice와 Bob간에 이루어지는 것이지만 보낸 이가 Eve이므로 https, smtps의 응용환경에서, 같은 메시지로 인식되는 것이 TLS 프로토콜의 취약점이다. 같은 메시지로 인식되면, 이후 공격자가 m_0값을 조작하여, m_1값과 관계없이 특수한 목적 (예: 전자결제 및 지급)에 악용할 가능성이 충분히 존재한다.

위 취약점을 방어하기 위해, IETF TLS 워킹 그룹에서, RFC 5746를 통해 대처법을 내놓았다 [11]. ClientHello와 SeverHello 메시지에 확장 필드를 추가하여 직전 핸드쉐이크로부터 형성된 키에 대한 확인 값을 (client_verify_data, server_verify_data) 추가적으로 할당하는 방안이다. 직전 핸드쉐이크에 대한 키 값을 알아야 하므로, 위와 같은 취약점을 방어할 수 있다.

2012년에, Giesen, Kohlar, Stebila에 의해, 재협상가능한 (renegotiable) TLS의 안전성 모델이 정립되었다 [1]. 또한, 위에서 설명한, 재협상과 관련한 공격에 [10] 대한 해결책을 담고 있는 IETF RFC 5746의 TLS가 제안된 안전성 모델에서 안전함을 증명하였다 [1].

Ⅳ. TLS 프로토콜의 최근 연구 결과

2012년, 2013년에, Crypto학회에서 TLS와 관련해서, 두 편의 연구결과가 [4, 6] 연속적으로 게재되었다. 이에 대한 결과를 간략히 요약하면 다음과 같다.

4.1. Jager, Kohlar, Schage, Schwenk의 연구 결과 [4]

참고문헌 [4]의 가장 큰 공헌도는 처음으로 TLS를 위한 안전성 모델과 안전성을 정의했다는 점이다. 지면 관계상, 안전성 모델의 핵심부분만 간단히 소개한다.

1) 참고문헌 [5]의 저자들은 자신의 결과가 Crypto 2013 학회에 제출하여 게재가 되지 않았음을 직접 밝히고 있다 [5].

4.1.1 안전성 모델

먼저, 모든 참여자, $P_i \in \{P_1,..,P_l\}$는 각자 롱텀 키 쌍 (pk_i, sk_i)를 소유하고 있다. 각 참여자 P_i는 오라클 $\pi_i^1,...,\pi_i^d$로 모델링 되며, d는 참여자 P_i가 열 수 있는 세션이다. 각 오라클은 총 다섯 개의 $(\Lambda, k, \Pi, \rho, st)$ 내부 상태 값을 유지한다.

- $\Lambda \in \{accept, reject\}$: 오라클의 수락, 거절 상태를 의미한다.
- $k \in K$: 키 값을 의미한다.
- $\Pi \in \{1,...,l\}$: 통신 파트너를 의미한다.
- $\rho \in \{Client, Server\}$: 통신 파트너의 ID를 의미한다.
- st : 상태 값으로서, DH 값 혹은 주고받은 임시 메시지들을 보관한다.

공격자는 다음의 네 개 질의에 대해 답변을 각각 받는다.

- $Send^{pre}(\pi_i^s, m)$: 공격자는 오라클 π_i^s에 대해 메시지 m을 보내기 위해 이 질의를 이용한다. 이 질의에 대한 답변 값은 프로토콜의 절차에 대한 이후 응답 내용이다. 단, 오라클 π_i^s의 상태가 수락(accept)인 경우에는 에러 메시지인 $\perp$를 출력하여, 수락 이후의 프로토콜 메시지들에 대해서는 $Send$ 오라클이 처리하지 않는다. 대신, 수락 상태 후의 $Send$질의는 $Decrypt$질의에서 처리한다. 이 점이 기존 AKE 안전성 모델의 주요 차이점이다.
- $Reveal(\pi_i^s)$와 $Corrupt(P_i)$: 공격자는 이 질의들을 통해 각각 대응되는 세션 키와 롱텀 키를 얻을 수 있으며, 이는 기존 AKE 안전성 모델과 동일하다.
- $Encrypt(\pi_i^s, m_0, m_1, len, H)$: m_0, m_1, len, H 값을입력으로 받는다. 만약 π_i^s의 상태가 수락 (accept)이 아닌 경우에는 에러 메시지인 $\perp$를 출력한다. 만약 수락이라면, 아래 [그림 2]의 $Encrypt$ 과정처럼 동작한다.
- $Decrypt(\pi_i^s, C, H)$: 입력으로 C, H 값을 받는다. 만약 π_i^s의 상태가 수락 (accept)이 아닌 경우에는 에러 메시지인 $\perp$를 출력한다. 만약 수락이라면, 위

$Encrypt(\pi_i^s, m_0, m_1, len, H)$:
$(C^0, st_e^0) \leftarrow StE.Enc(k_{enc}^p, len, H, m_0, st_e)$
$(C^1, st_e^1) \leftarrow StE.Enc(k_{enc}^p, len, H, m_1, st_e)$
$If\ C^0 = \perp \text{ or } C^1 = \perp\ then,\ return \perp$
$u_i^s = u_i^s + 1$
$(C_i^s[u_i^s], st_e) = (C^{b_i^s}, st_e^{b_i^s})$
$Return\ C_i^s[u_i^s]$

$Decrypt(C, H)$:
$(j,t) = \theta_i^s$
$v_i^s = v_i^s + 1$
$If\ b_i^s = 0, then\ return \perp$
$(m, st_d) \leftarrow StE.Dec(k_{dec}^p, H, C, st_d)$
$If\ v_i^s > u_j^t \text{ or } C \neq C_j^t[v_i^s], then\ phase = 1$
$If\ phase = 1, then\ return\ m$

〔그림 2〕 ACCE 안전성 모델에서 $Encrypt, Decrypt$ 오라클 동작 원리

[그림 2]의 $Decrypt$ 과정처럼 동작한다.

위 두 $Encrypt$, $Decrypt$질의는 기존의 AKE 안전성 모델과 구별되는 질의로서, 오라클 π_i^s가 수락(accpet)이 되어야 동작한다. 이것은 공격자가 TLS의 레코드 레이어에서 임의의 메시지를 암복호화 하여 전달하는 행위를 모델링한 것이다. $Encrypt, Decrypt$질의를 위해, π_i^s는 추가적인 내부 변수 $(u_i^s, v_i^s, C_i^s, \theta_i^s)$를 둔다. u_i^s, v_i^s는 $Encrypt, Decrypt$에 사용되는 카운터 값이고, C_i^s는 암호문 리스트이며, $C_i^s[u]$를 u번째 암호문이라 표기한다. θ_i^s는 상대방 파트너의 오라클 인덱스 값을 보관한다. 만약, π_i^s가 π_j^t와 통신 파트너라면, $\theta_i^s = (j,t)$라 정의한다.

4.1.2 안전성 정의

안전성은 도전자와 참여자로 구성된 게임으로 정의된다. 도전자는 우선, l개의 롱텀 키 쌍 (pk_i, sk_i)을 생성한다. 그리고 랜덤한 b_i^s값을 선택한다. 공격자 A는 공개키 $pk_1,...,pk_l$을 입력으로 받는다. 공격자는 앞에서 정의한 질의들을 할 수 있으며, 도전자는 선택한 b_i^s값을 이용해 $Encrypt, Decrypt$에 대한 질의를 답변한다. 공격자는 궁극적으로 b'값을 추측한다.

안전성은 크게 두가지를 정의한다. 첫째는, 주어진 프로토콜이 안전한 인증 프로토콜임을 보장한다. 둘째는 주어진 프로토콜이 인증된 기밀 채널임을 보장한다. 구체적으로, 공격자 A가 다음의 두 경우 중 한 경우라

도 만족시킬 수 있다면, 주어진 프로토콜을 깼다라고 $((t,\epsilon)-breaks)$ 정의하였다. 첫 번째 경우는 공격자가 인증을 깨는 경우로서, 사용자 π_i^s와 매칭 대화 관계에 있는 π_j^t가 존재하지 않게 만드는 경우이다. 두 번째 경우는, 공격자가 b'을 정확히 추측하는 경우로서, SLHA의 인증된 기밀성 채널을 깨는 경우이다.

[인증 공격자에 대한 안전성] : 공격자 A가 프로토콜 종료 시, ϵ의 확률로 다음을 만족하는 오라클 π_i^s가 존재하는 경우

- 공격자 A가 파트너 P_j에게 τ_0번째 질의를 던졌을 때 오라클 π_i^s가 수락한다고 했을 때, P_j는 $\tau_j(\tau_0 < \tau_j)$ 시점에서 $corrupt$ 되어야 한다.
- π_j^t는 프로토콜 상에서 수락(accept) 했을 경우에, 공격자는 π_j^t에게 (또한 매칭 대화 관계에 있는 π_j^t에게도) $Reveal$질의를 수행하지 않아야 한다.
- π_i^s와 매칭 대화 관계에 있는 π_j^t가 존재하지 않는다.

만약, 오라클 π_i^s가 위의 조건으로 수락된다면, 이를 악의적 (malicious) 수락 (accept) 이라고 정의한다.

[암호화 내용에 대한 안전성] : 공격자 A가 다음의 조건을 만족하며 b'을 출력하며 종료하는 경우

- 공격자 A가 파트너 P_j에게 τ_0번째 질의를 던졌을 때 오라클 π_i^s가 수락한다고 했을 때, P_j는 $\tau_j(\tau_0 < \tau_j)$ 시점에서 $corrupt$ 되어야 한다.
- π_j^t는 프로토콜 상에서 수락(accept) 했을 경우에, 공격자는 π_j^t에게 (또한 매칭 대화 관계에 있는 π_j^t에게도) $Reveal$질의를 수행하지 않아야 한다.

b'이 b_i^s가 되는 확률이 ϵ보다 클 경우, $\Pr[b_i^s = b'] - 1/2] \geq \epsilon$, A가 정확히 b'을 추측했다라고 정의한다.

안전성 정의 중 가장 큰 특징은 두 번째의 조건이다. 이는 공격자가 TLS 프로토콜에서 매우 간단하게 공격에 성공할 수 있는 경우를 배제시킨 것이다. 예를 들어, [그림 1]에서, 서버가 프로토콜에서 마지막 메시지, m_{13}을 보내는 경우를 보자. 서버는 이 메시지가 사용자에게 잘 전달되었는지 확인하지 않고, 수락 단계에 들어가게 된다. 이 때에 공격자 A가 $Reveal$질의를 하게 되면, 키 값 k_{enc}^{server}을 통해 finished 메시지인 fin_s값을 쉽게 알 수 있다. 이후, A는 키 값과 랜덤 값을 이용해 m_{13} 메시지를 보내지 않고, m_{13}'메시지 값을 전달한다. 즉, π_i^s는 π_j^t와 매칭 대화 관계에 있지 않으면서 수락하는 상황이 된다. 이러한 경우를 제외하였다.

4.2. Krawczyk, Paterson, Wee의 연구 결과 [6]

참고문헌 [6]의 연구 핵심은 TLS 프로토콜을 key encapsulation mechanism (KEM)을 이용하여 표현했다는 점이다. 참고문헌 [6]의 결과에 의하면, TLS 프로토콜은 KEM이라는 암호 프리미티브로 잘 포장되어 정의할 수 있다는 사실이다. 그리하여, 기반이 되는 KEM의 안전성 성질 만족 정도에 따라 최종적인 TLS의 ACCE 안전성을 만족할 수 있도록 하는 프레임워크를 제시하였다. 사실, 이러한 접근법은 참고문헌 [3]에 의해 처음 연구되었다. 하지만, 그 범위가 RSA로 키를 분배하는, TLS-RSA 모드에 국한되었으며, 참고문헌 [6]에서, 처음으로 이를 TLS-RSA, TLS-DH, TLS-DHE 모든 모드에 적용한 점이 가장 큰 공헌도이다. 먼저, TLS 프로토콜을 KEM으로 표현한 후, KEM으로 표현된 TLS가 IND-CCCA[2] 안전성을 만족하면, 참고문헌 [4]에서 제안한 ACCE를 만족함을 보였다. 그 후 KEM으로 표현된 세 개의 TLS-RSA, TLS-DHE, TLS-DH가 각각 IND-CCCA를 만족함을 보임으로, 궁극적으로 세 개의 모드들이 ACCE 모델 하에서 안전함을 증명하였다. 해당 프로토콜이 각각의 모드에 대한 연구결과를 간략히 정리하면 다음과 같다.

- TLS-RSA: TLS 표준상의 RSA 암호화 알고리즘은 PKCS #1v1.5를 사용한다. 이 암호알고리즘이 OW-PCA [3] 에 안전하다면, 랜덤 오라클 모델 하에서, IND-CCCA임을 보였다.

2) constrained IND-CCA로 제약된 복호화 오라클을 사용한다. 지면관계상, 이에 대한 자세한 정의는 [2]를 참고하기 바란다.

- TLS-CCA: TLS-CCA는 TLS-RSA에서 RSA알고리즘을 IND-CCA에 안전한 RSA-OAEP로 바꾼 모드를 의미한다. TLS-CCA가 ACCE모델 하에 안전함을 오직 표준적 가정만을 사용하여 증명하였다.
- TLS-DH, TLS-DHE: TLS-DH, TLS-DHE에 대해서 PRF-ODH 가정을 이용해서 ACCE 안전성을 만족함으로 보였고, TLS-DH는 SDH(strong DH)가정을 이용해서 랜덤오라클 모델에서 ACCE 안전성을 만족함을 보였다.

Ⅴ. 결론

본 논문을 통해, TLS 최근 연구 결과들을 살펴보았다. TLS에 대한 연구가 오랫동안 진행되어왔지만, 최근에 처음으로 ACCE 안전성 모델이 정립되고 관련 세 개의 모드에 대한 증명 가능성이 밝혀졌다. TLS가 간단하지 않은 프로토콜이기 때문에, 현재 제안된 ACCE 안전성은 TLS와 관련한 다양한 공격에 대해서 완전한 모델을 제공하지 못할 뿐더러, 프로토콜 전체에 대해서도 안전성 정의가 미흡한 부분이 있다. 가령, 최근 이슈가 되어온, 핸드쉐이크 프로토콜에서 발생되는 alert 및 error 메시지에 대해, 공격자가 얻을 수 있는 정보의 양은 많다. 기존의 AKE안전성 모델만 이용하여 이러한 alert, error메시지를 포함하는 TLS 프로토콜의 안전성을 정의하는 것은 분명 한계가 있다. 또한, TLS에서 제공하는 재 인증(reauthentication), 새로운 키 만들기(rekyeing) 세션 재개 (session resumption)와 같은 이슈들을 함께 포함하는 안전성 모델 개발도 필요하다.

참고문헌

[1] Florian Giesen, Florian Kohlar, and Douglas Stebila, “On the Security of TLS Renegotiation”, Cryptology ePrint Archive, Report 2012/630, http://eprint.iacr.org/2012/630

[2] D. Hofheinz and E. Kiltz, “Secure hybrid encryption from weakened key encapsulation”, In Proc. of Crypto, pages 553-571, 2007.

[3] J. Jonsson and B. S. Kaliski Jr., “On the security of RSA encryption in TLS”, In Proc. of Crypto 2002, LNCS 2442 pages 127-142, Springer, August, 2002.

[4] Tibor Jager, Florian Kohlar, Sven Schage, and Jorg Schwenk, “On the Security of TLS-DHE in the standard model” In Proc. of Crypto 2012, LNCS vol. 7417, pages 273-293, Springer, August, 2012.

[5] Florian Kohlar, Sven Schage, and Jorg Schewenk, “On the Security of TLS-DH and TLS-RSA in the Standard Model”, Cryptology ePrint Archive, Report 2013/367, http://eprint.iacr.org/2013/367

[6] Hugo Krawczyk, Kenneth G. Paterson, and Hoeteck Wee, “On the Security of the TLS Protocol: A Systematic Analysis”, In Proc. of Crypto 2013, LNCS vol. 8042, pages 429-448, Springer, August, 2013.

[7] Paul Morrissey, Nigel P. Smart, and Bogdan Warinschi, "A modular security analysis of the TLS handshake protocol", In Proc. of Asiacrypto 2008, LNCS vol. 5350, pages 55-73. Springer, December 2008.

[8] Paul Morrissey, Nigel P. Smart, and Bogdan Warinschi, "The TLS handshake protocol: A modular analysis", Journal of Cryptology, 23(2): 187-223, April 2010.

[9] Kenneth G. Paterson, Thomas Ristenpart, and Thomas Shrimpton, “Tag size does matter: attacks and proofs for the record protocol. In Proc. of Asiacrypt 2011, LNCS 7073, Springer, December, 2011.

[10] Marsh Ray and Steve Dispensa, "Renegotiating TLS" November 2009. http://extendedsubset.com/Renegotiating_tls.pdf

[11] Eric Rescorla, Marsh Ray, Steve Dispensa, and Nasko Oskov, "Transport layer security(TLS) renegotiation indicatin extension", February 2010, RFC 5746.

A. 부록

A.1. SLHA (stateful length-hiding authenticated encryption) 암호

SLHA는 $(StE.Init, StE.Enc, StE.Dec)$ 총 3개의 알고리즘으로 구성된다. $StE.Init$ 알고리즘은 암복호화에 사용되는 상태 값 (st_e, st_d)을 우선 초기화하고, 다음의 과정으로 암복호를 수행한다.

$$(st_d, st_e) = StE.Init(),$$
$$(C, st'_e) \leftarrow StE.Enc(k, len, H, m, st_e),$$
$$(m', st'_d) \leftarrow StE.Dec(k, H, C, st_d)$$

k는 암복호화 키 값이고, $H \in \{0,1\}^*$는 헤더 값이며, len은 암호문 크기이다. 암호화는, 입력 메시지 (k, len, H, m, st_e)에 대해 암호문 C와 업데이트 된 상태 값 st_e'를 출력한다. 복호화는, 입력 메시지 (k, H, C, st_d)에 대해 평문 m과 업데이트 된 상태 값 st_d'을 출력한다. 공격자 A가 다항식 시간 t 동안 다음 실험을 통해 b값을 추측할 확률이 무시할 수 있는 확률이라면, 즉, $\Pr[b = b'] \leq \epsilon_{slhae}$이라면, (t, ϵ_{slhae})-secure SLHA이라 정의한다.

$$b \leftarrow \{0,1\},\ k \leftarrow \{0,1\}^{|k|},\ set(st_e, st_d) \leftarrow StE.Init$$
$$\text{run } b' \leftarrow A^{Encrypt, Decrypt}$$
$$Encrypt(m_0, m_1, len, H):$$
$$u = u + 1$$
$$(C^0, st_e^0) \leftarrow StE.Enc(k, len, H, m_0, st_e)$$
$$(C^1, st_e^1) \leftarrow StE.Enc(k, len, H, m_1, st_e)$$
$$If\ C^0 = \perp \text{ or } C^1 = \perp\ then,\ return \perp$$
$$(C_u, st_e) = (C^b, st_e^b)$$
$$Return\ C_u$$
$$Decrypt(C, H):$$
$$v = v + 1$$
$$If\ b = 0, then\ return \perp$$
$$(m, st_d) \leftarrow StE.Dec(k, H, C, st_d)$$
$$If\ v > u \text{ or } C \neq C_v, then\ phase = 1$$
$$If\ phase = 1, then\ return\ m$$
$$Return\ \perp$$

〔그림 3〕 SLHAE 실험 및 암복호화 오라클 동작 과정

〈著者紹介〉

변 진 욱 (Byun Jin Wook)
종신회원

2001년 2월 : 고려대학교 전산학과 이학사
2003년 2월 : 고려대학교 정보보호대학원 공학석사
2006년 8월 : 고려대학교 정보보호대학원 공학박사
2006년 10월~2007년 11월 : Royal Holloway University of London, ISG 박사후연구원
2009년 ~ 현재 : 한국정보보호학회 이사
2008년 3월 ~ 2010년 2월 : 평택대학교 정보통신학과 전임강사
2010년 3월 ~ 현재 : 평택대학교 정보통신학과 조교수
관심분야 : 정보보호 프로토콜, 인증, 프라이버시 보호 기술, 데이터베이스 보안