클라우드 시스템에서 배타적 논리합을 이용한 키 생성 및 관리의 문제점

채승재, 노종선

서울대학교 전기정보공학부 뉴미디어통신공동연구소

chae950104@snu.ac.kr, jsno@snu.ac.kr

Weakness of XOR key generation & management in cloud system

Chae Seung Jae, No Jong-Seon

INMC, Department of ECE, Seoul National Univ.

요 약

본 논문에서는 클라우드 시스템에서의 파일 전송 및 저장에 필요한 key에 대한 요구가 증가함에 따라서 새롭게 제안된 방식의 문제점을 제기한다. 기존 제안된 방식은 keystore seed를 이용하여 적은 양의 정보 저장으로도 많은 key들을 만들어 낼 수 있고, 생성된 key들은 서로 information-theoretically하게 안전하다고 주장한다. 중복 사용된 keystore seed들에 의한 문제점들을 소개하고 공격에 취약함을 설명한다.

Ⅰ. 서 론

최근 클라우드를 이용한 파일 저장 혹은 파일 전송의 사용이 많아지면서 그에 필요한 많은 scheme들이 제안되고 있다. 시스템에서 사용자가 비밀번호만 이용하는 방식에는 dictionary attack같은 공격들에 취약하기에 [1], 암호화된 key를 이용하는 방식이 필요하다. Key를 사용하는 방식은 사용자에 대한 추가적인 정보를 드러내지 않는 장점을 갖고 있다. 암호화된 key들은 file encryption, virtual private networks에 대한 접근 그리고 user authentication 같은 다양한 분야에서 사용 할 수 있다.[2] 클라우드 시스템은 장기간 저장하는 것과 유저들이 빈번하게 접속하는 특성을 갖고 있기에 각 파일들이 서로 다른 random key로 encrypted 돼야 ciphertext-only attack이나 chosen-plaintext attack 같은 공격들에 안전하다.[3] 클라우드를 이용하는 유저들이 증가함에 따라서 필요한 key의 개수도 증가하는데 앞서 말한 조건들을 만족해야한다. 기존에 제안한 방식은 위의 문제들을 해결하기 위해서 key를 만드는 기반이 되는 keystore seed로부터 배타적 논리합 연산을 이용하여 많은 양의 key를 안전하게 생성할 수 있다고 주장한다.[4] 본 논문에서는 이 방식은 각 key들이 keystore-seed에 대한 정보들을 담고 있다는 문제점이 있고 공격에 취약하다는 것을 밝혀낸다.

Ⅱ. Information-Theoretically Secure한 key 생성 및 관리

A. Key 생성

기존에 제안된 key를 생성하기 위한 방식을 소개한다. 적은 양의 keystore seed으로 key index값을 알면, 많은 key들을 생성 할 수 있는 방식이다.

K = Keystore Seed

k_i = Key

i = Key Index

여기서 keystore seed K는 L bits인 random string이다.

$K = K(0)K(1)\cdots K(L-1)$

Key of length l :

$$k(m_1, m_2, \cdots, m_t) = \sum_{i=1}^{t} K(m_i)K(m_i+1)\cdots K(m_i+l-1)$$

(Summation of Strings is Binary Addition)

Keystore ψ :

$$\Psi = \{k(m_1, m_2, \cdots, m_t) : 0 \le m_1 < m_2 < \cdots < m_t \le L-1\}$$

The Number of Random Keys : $\Lambda = \binom{L}{t}$

B. Key 관리

1) Encryption Key k_i를 랜덤하게 하나 뽑은 뒤, file을 encrypt한다.

2) 암호화된 ciphertext와 key index i를 같이 전송한다.

3) K는 secret하게 공유하고 있으므로, key index를 받은 유저가 K와 i를 통해 k_i를 추출해내고, 이것으로 decrypt를 한다.

C. Information-Theoretically ϵ-secure

생성된 l길이의 key들은 $0 \le \epsilon < 1$에 대하여 다음과 같은 성질을 가진다.

1) 주어진 i에 대해서 $1 \le i \le \Lambda, k_i$ 는 random 하고 $\{0,1\}^l$로 uniformly distributed 하다.

2) 두 개의 독립인 $i, j, 1 \le i, j \le \Lambda$에 대해서

$$\Pr\{k_i = k_j\} \le (1-\epsilon) \times 2^{-l} + \epsilon$$

을 만족한다.

3) 두 개의 독립인 $i, j, 1 \le i, j \le \Lambda$에 대해서

$$H(k_j|i, j, k_i) \ge H(k_j|j) \times (1-\epsilon) = l(1-\epsilon)$$

을 만족한다.

Ⅲ. 배타적 논리합 방식 키 생성 및 관리의 문제점

기존 논문에서는 keystore seed를 이용한 배타적 논리합 key 생성 및 관리 방식을 사용하면, 공격자가 K를 모르기 때문에 위의 C에서 3)과 같이 index만 전송해도 각 유저들이 정보 이론적으로 안전한 key를 얻을 수 있다고 주장한다. 즉, key를 알아내도 다른 key에 대한 정보를 알아낼 수 없다는 것이다. Key 생성 방식에 문제점을 설명하기 위해 하나의 key k_i가 t=5일 때, 어떻게 생성되는지를 보겠다. encryption key k_i를 하나 뽑은 뒤, 그에 해당하는 index를 ciphertext와 같이 전송할 때, key index m_1, m_2, m_3, m_4, m_5 값을 알 수 있는 것이다. 그리고 이 index로부터 l만큼의 길이들이 합쳐서 k_i가 되는 것인데, 그 5개의 합쳐진 결과 값도 알 수 있다. 이 경우에서 만약 m_1, m_2, m_3, m_4로부터 시작되는 $4l$만큼의 값을 알고 있다면, m_5부터 시작되는 l만큼의 값은 다 더해진 값을 알고 있기에 자동적으로 알게 된다. 이는 각 key들은 keystore seed의 l bits 만큼의 정보를 담고 있는 것이다.

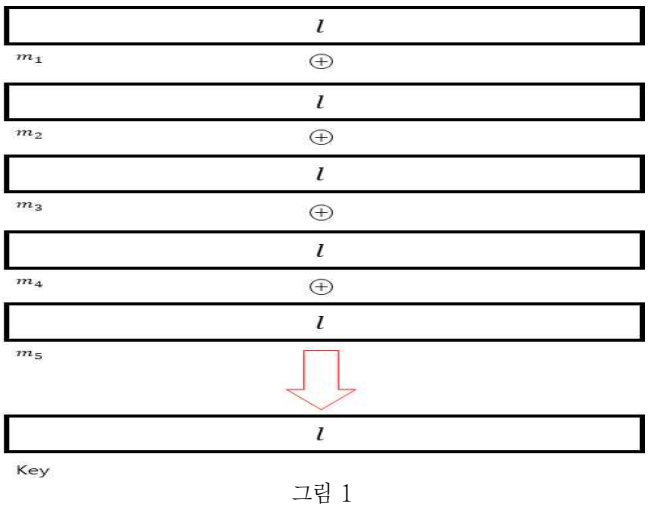

그림 1

위의 그림1을 통해 자세히 설명하면, 한 개의 key는 각 5개의 index로부터 시작되는 keystore seed의 l만큼의 정보들의 합으로 이뤄져있다. $4l$만큼의 known value를 갖고 있을 때, 모르는 하나의 index 값을 $m_5 = 340$이라 한다면, keystore seed $K = K(0)K(1)\cdots K(L-1)$에 대해서 $K(340)K(341)\cdots K(340+l-1)$의 정보를 자동적으로 구할 수 있는 것이다. 같은 방식으로 여러 개의 key에 대해서 정보를 알아내면 일정 개수의 key를 넘어 서면 keystore seed 전부를 알아낼 수 있다. 이는 논문에서 주장하는 아래 식에서 n개만큼의 key에 대해 그 값과 index 값을 알아도 index j에 대한 k_j의 엔트로피가 $l(1-\epsilon)$보다 큰 것이 아니라, 주어진 index에 대해 하나의 값으로 정해지기에 엔트로피가 0이 되는 것이다.

$$H(k_j|i_1,\cdots,i_n,j,k_{i_1},\cdots,k_{i_n}) \geq H(k_j|j)\times(1-\epsilon) = l(1-\epsilon)$$

K를 전부 알아내게 되면 1차적으로는 해당 index값만 알아도 그에 맞는 key를 도출해 낼 수 있기에 논문에서 주장하는 각 key들이 정보 이론적으로 안전하다는 것이 틀림을 밝힐 수 있다. 2차적으로는 index값을 몰라도 생성할 수 있는 모든 index조합에 대한 모든 key들을 알아낼 수 있기에 시스템 전체가 위협을 받을 수 있다.

위의 방식과 달리 한 개의 key에 대해서 특별한 경우에는 가능한 후보의 개수를 줄일 수 있다. t=5일 때에 대하여 index 값들 중, $0 \leq m_1 < m_2 < m_3 < m_4 < m_5 \leq l-1$을 만족하는 경우에 $X = m_5 - m_1$이라고 하면, $3 < X < l$을 만족하고, 이때 해당하는 모든 index에 대한 keystore seed들의 가능한 후보군은 2^X개가 된다. 이 값은 일반적인 경우에서의 가능한 후보군인 2^{4l}보다 매우 적은 양인데, 이는 중복 사용으로 인한 또 하나의 문제점이라고 볼 수 있다. 위에서 설정한 $3 < X < l$ 인 경우에 대해서 확인해보면, 하나의 key를 생성할 때, 각 값들에 대해서 사용하는 빈도수는 아래의 표1과 같다.

5회 사용	$l - m_5 + m_1$
4회 사용	$(m_5 - m_4) + (m_2 - m_1)$
3회 사용	$(m_4 - m_3) + (m_3 - m_2)$
2회 사용	$(m_3 - m_2) + (m_4 - m_3)$
1회 사용	$(m_2 - m_1) + (m_5 - m_4)$

표 1

총 사용된 변수 $5l$에서 위의 표에 중복돼서 사용된 횟수만큼을 빼주면, $3 < X < l$일 때의 실제 독립인 변수의 개수는 $l + m_5 - m_1$개 즉, $l + X$개가 된다. 이 경우에서도 변수 $l + X$개에 대해서 후보군은 2^X개이므로, l bits만큼의 정보는 담고 있는 것이다. 이런 특별한 경우에서는 key가 한 개만 있어도 적은 양의 연산으로 keystore seed에 대한 후보군을 얻을 수 있는 것이다. 여러 개의 key에서도 index값을 확인하여 이런 특별한 경우에 만족하는지를 먼저 확인한 뒤, 그 결과를 토대로 다른 값들도 확인해 나가면 더 이른 시간에 공격이 가능한 장점이 있다.

Ⅳ. 결 론

본 논문에서는 기존에 제안한 방식인 적은 양의 정보저장으로도 keystore seed에서의 배타적 논리합을 이용하여 많은 양의 key를 생성 할 때, 중복 사용되는 keystore seed값들에 의해 문제점이 생기고 생성된 key들이 제안한 방식만큼의 독립성을 갖지 않는 것을 밝혀냈다.

ACKNOWLEDGMENT

이 논문은 2016년도 정부(미래창조과학부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (R-20160229-002941, IoT 및 클라우드 컴퓨팅을 위한 경량 포스트 양자 암호 시스템 연구)

참 고 문 헌

[1] Morris, Robert, and Ken Thompson. "Password security: A case history." *Communications of the ACM* 22.11 (1979): 594-597.

[2] Monrose, Fabian, et al. "Cryptographic key generation from voice." *Security and Privacy, 2001. S&P 2001. Proceedings. 2001 IEEE Symposium on*. IEEE, 2001.

[3] Menezes, Alfred J., Paul C. Van Oorschot, and Scott A. Vanstone. Handbook of applied cryptography. CRC press, 1996.

[4] Yang, En-Hui, and Xin-Wen Wu. "Information-theoretically secure key generation and management." *Information Theory (ISIT), 2017 IEEE International Symposium on*. IEEE, 2017.