Merkle Tree 기반의 로그인증 메커니즘 설계 및 분석★

이 정 엽*, 박 창 섭**

요 약

보안로그의 활용범위가 다양해짐에 따라 저장된 로그 데이터에 대한 무결성의 중요성이 높아지고 있다. 특히, 저장된 로그 데이터는 시스템에 침입한 공격자들이 자신의 흔적을 없애기 위해 우선적으로 조작되는 대상이다. 키 정보가 노출이 된 이후의 로그 데이터의 안전성은 보장하지 못하지만, 그 이전에 축적된 로그 데이터 무결성의 전방 안전성을 보장하는 다양한 이론적 기법들이 소개되었다. 이런 기법들의 인증태그들은 선형 해시체인을 기반으로 하고 있다.

이 경우 부분 검증이 어렵고, 인증태그 생성속도와 검증속도를 높이기 힘들다. 본 논문에서는, 부분 검증이 용이하고 멀티 스레딩이 적용 가능한 Merkle Tree 기반의 로그인증 메커니즘을 제안한다.

Design and Analysis of the Log Authentication Mechanism based on the Merkle Tree

Jung yeob Lee*, Chang seop Park**

ABSTRACT

As security log plays important roles in various fields, the integrity of log data become more and more important. Especially, the stored log data is an immediate target of the intruder to erase his trace in the system penetrated. Several theoretical schemes to guarantee the forward secure integrity have been proposed, even though they cannot provide the integrity of the log data after the system is penetrated. Authentication tags of these methods are based on the linear-hash chain. In this case, it is difficult to run partial validation and to accelerate generating and validating authentication tags. In this paper, we propose a log authentication mechanism, based on Mekle Tree, which is easy to do partial validation and able to apply multi threading.

Key words : Integrity, Log System, Merkle Tree, Audit Log, Forward Security

접수일(2017년 2월 13일), 게재확정일(2017년 3월 21일)

★ 본 연구는 미래창조과학부 및 한국인터넷진흥원의 "고용계약형 정보보호 석사과정 지원사업"의 연구결과로 수행되었음 (과제번호 H2101-16-1001).

★ 본 연구는 2016년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업의 지원을 받아 수행된 것임. (NRF-2014R1A12055074).

* 단국대학교/컴퓨터학과

** 단국대학교/컴퓨터학과

1. 서 론

소셜 네트워크 시장의 발달에 따라 하루에도 많은 양의 데이터들이 쏟아지는 시대가 왔다. 이전 보다 많아진 데이터의 양만큼 데이터들의 가치가 커지고 있다. 최근 고도화된 사이버 공격들이 늘어남에 따라 시스템 및 네트워크 장비 그리고 다양한 보안 및 응용 소프트웨어로부터 발생되는 로그들은 다른 데이터들에 비해 그 활용범위만큼이나 중요한 가치를 갖는다.

로그 데이터는 여러 가지 방식의 로그 관리시스템에 의해 수집되며, 이렇게 수집된 로그들은 기업과 기관에서 시간과 노력을 들여 관리를 해야 하는 중요한 자산이다. 수집된 로그는 기본적인 보안사고 탐지부터 비정상행위 탐지, 내부감사 및 포렌식 분석 그리고 시스템 이용분석을 통한 자원의 효율적 배분과 장기 플랜의 확보 등에도 이용된다. 또한, 국내외적으로 제정된 다양한 IT 컴플라이언스를 준수하기 위해서도 로그 시스템을 통한 전자적 기록물의 유지관리가 의무화되고 있다 [1].

국내 로그 관리 관련 규제로는 개인정보보호법, 신용정보보호법, 전자 상거래법, 전자정부법, 의료법 등 여러 분야에 걸쳐 제정되어 있다. 예를 들어, 의료정보분야에 있어서는 진단 방사선 또는 이식형 심장제세동기 등의 IT 의료기기를 통한 환자진단 및 치료에 대한 기록유지는 의료사고 예방 및 사후 분쟁조정을 위한 중요한 토대가 되고 있다 [2]. 국외 로그관리 규제로는 ISO 27001, SOA(Sarabane-Oxely Act), HIPPA 등이 있다 [3].

로그 관리 인프라에도 많은 이슈들이 있지만 이를 종합해보면 다음과 같은 4가지로 나눌 수 있다. 첫째, 로그는 실시간으로 생성되지만, 이를 관리하는 리소스는 한정적이다. 즉, 제한된 로그관리 자원 내에서 지속적으로 발생되는 대용량 로그의 발생 속도와 데이터 저장 속도간의 불균형을 조정하는 것이 필요하다. 둘째, 효율적이고 유효한 로그분석을 위해서는 적절한 로그분석 정책 및 인프라의 구축이 요구된다. 셋째, 이기종 장비에서 발생하는 다양한 형태의 로그들을 통합하기 위한 로그 포맷이 필요하다. 또한, 수집 장비들 간의 연동된 타임스탬프를 필요로 한다. 넷째, 로그 데이터의 수집 및 저장에 있어서 로그의 무결성이 보장되어야 한다. 로그 데이터의 무결성은 활용유형에 관계없이 로그 데이터 전송 및 저장관리에 있어서 필수적인 요구사항이 된다. 추가적으로, 수집된 로그 데이터에는 민감한 개인정보가 포함될 수도 있고 기업의 영업기밀에 대한 정보 역시 유추가 가능하기에 활용유형에 따라서 로그 데이터의 기밀성이 요구된다.

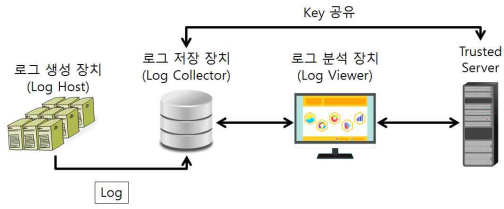
로그 인프라 이슈들 중 저장 방법 및 분석 방법에 대한 연구는 많이 진행되고 있다. 또한, 국내외 표준에서 다양한 시스템에서 생성한 로그들을 수집하여 분석하기 위한 포맷이 정의되고, 이는 대부분 syslog [3]를 기반으로 한다. 본 논문에서 중점적으로 볼 것은 상대적으로 관심을 덜 받고 있는 무결성을 지키기 위한 방법들이다.

무결성을 지키기 위한 방법들 중 로그를 수집할 때 로그 레코드에 인증태그를 추가함으로써 수집된 로그 파일들의 무결성을 보장하기 위한 기존의 방법들을 분석한다. 또한, 기존의 방법들을 보완하기 위한 Merkle Tree에 기반을 둔 기법을 제안한다.

2. 로그 시스템

2.1 로그 시스템 구성

본 논문에서 가정하고 있는 로그 인프라를 그림으로 표현하면 (그림 1)과 같은 구성을 갖는다. 다수의 로그 생성장치(Log Host)를 통해 생성된 로그 데이터는 로그 저장장치(Log Collector)에 전송/저장된다. 로그 분석장치(Log Viewer)는 정기적으로 또는 사용자 요청에 의해 로그 저장장치에 있는 데이터를 받아 로그 데이터를 분석하게 된다. 로그 저장장치는 필요에 따라 Trusted Server와의 통신을 통해 메시지 인증코드(Message Authentication Code, MAC) 또는 디지털 서명 값 생성에 필요한 키 정보를 공유한다. 로그 분석장치의 경우 인증태그 생성 및 검증방법에 따라 Trusted Server로부터 키를 다운로드 받거나, 인증태그를 서버로 전송하여 검증 결과를 받을 수도 있다. 이 때, 로그 생성장치 또는 로그 저장장치에 침투한 공격자는 로그저장 과정, 로그 분석과정에 간섭할 수 있어 보안 이슈가 발생한다.



(그림 1) 로그 시스템 구성

로그 생성장치에 있는 로그 데이터의 무결성을 보장하기 위해 가장 많이 논의되고 있는 것이 실시간 전송을 통한 백업이다. 하지만 이 경우 앞서 서론에서 언급했듯이 전송속도와 생성속도의 차이로 인해 아직까지는 완벽한 실시간 전송은 힘들다. 로그 저장장치에 있는 로그 데이터의 무결성을 보장하기 위해 가장 많이 쓰이고 있는 간단한 방법은 WORM(Write-Once Read-Many)디바이스에 로그를 저장하는 것이다. 하지만, 최근에 IBM System Journal에 발표된 것처럼 모든 WORM Storage가 안전한 것은 아니다 [5].

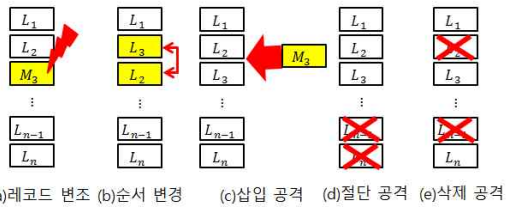
위와 같은 이유로 위해 전통적인 대칭키 및 공개키 암호 그리고 MAC 등이 활용되지만, 로그 시스템에 대한 보안위협모델은 로그저장장치 자체가 “외부공격자”에 의해 침투되어 로그 데이터를 보호하는 키 정보가 노출 될 수 있다는 가정을 하고 있다. 결국, 로그저장장치에 보관된 키 정보가 노출이 된 이후의 로그 데이터의 안전성은 보장하지 못하지만, 그 이전에 축적된 로그 데이터 무결성의 전방 안전성(Forward-Security) [4]의 확보가 기본적인 보안요구사항이 된다. 즉, 공격자에 의한 키 노출 이전에 수집 및 저장된 로그들에 대해서 무결성을 보장하는 것을 “Forward-Secure 무결성”이라고 하며, 본 논문에서는 이를 보장하기 위한 관련 연구들에 대해 살펴보고 이들의 단점을 보완할 새로운 방법을 제안할 것이다.

2.2 로그 시스템 위협모델

본 논문에서 중점적으로 살펴볼 로그 시스템 위협모델은 로그 저장장치에 수집된 로그들의 무결성을 위협하는 모델들이다 [6,7]. 수집 장치에 저장된 로그들에 가해지는 공격은 크게 3가지로 나눌 수 있다. 로그 파일들에 대한 공격은 변조, 삽입, 삭제로 나눌 수 있고, 이들 모두는 넓은 의미에서 모두 변조 공격(Modification Attack)으로 정의된다.

좁은 의미에서 로그 레코드에 대한 변조공격의 경우 다시 두 가지로 나눌 수 있다. 로그 레코드의 내용 자체를 변경함으로써 로그의 내용을 공격자에게 유리하게 만드는 레코드 변조 공격과 로그 레코드의 순서를 바꿈으로써 로그 데이터의 의미를 왜곡시키는 순서 변경공격(Re-ordering Attack)이 있다. 삽입 공격의 경우도 로그 레코드를 임의의 장소에 삽입함으로써 로그 데이터의 의미를 왜곡시킬 수 있다.

삭제공격의 경우 임의의 로그 레코드를 삭제하는 레코드 삭제 공격과 로그파일의 끝부분에 존재하는 일정 개수의 로그항목들을 임의로 삭제시키는 절단 공격(Truncation Attack)으로 구분할 수 있다. 이 둘은 레코드를 삭제한다는 점에서 물리적인 측면의 효과는 동일하지만, 이들에 대응하기 위한 보안기법 설계는 상이하기에 차별성을 두어야 한다. 즉, 전자를 위한 보안기법이 그대로 후자를 위한 보안기법으로 적용될 수는 없다. 본 절에서 제시한 위협모델들을 그림으로 표현하면 (그림 2)와 같다.



(그림 2) 로그 위협 모델

2.3 인증태그

로그 시스템 전반에 걸쳐 무결성의 보장을 위해 로그 레코드에 첨가되는 인증태그는 개별 로그항목을 보호하는 개별인증태그(Individual Authentication Tag)와 전체 로그항목들을 보호하는 누적인증태그(Aggregated Authentication Tag)가 있다. 인증태그는 생성에 쓰이는 보안기법에 따라 대칭키 방식과 공개키 방식으로 나눌 수 있다. 대칭키 방식의 경우 대칭키 K를 이용한 메시지인증코드(MAC, Message Authentication Code)를 사용하고, 공개키 방식의 경우 서명용

개인키를 이용한 디지털 서명 함수를 통해 생성된 서명을 사용한다. 공개키 방식의 경우 아직 키 교환 및 생성 등의 문제에서 대칭키 방식보다 시간이 오래 걸리므로 대용량 로그 보호에 적합하지 않다고 판단하여 본 논문에서는 대칭키 방식에 초점을 맞춘다.

인증태그 생성에 사용되는 키의 경우 2.1절에서 언급했듯이 로그 저장장치와 Trusted Server간에 공유된다. 로그 저장장치에 있는 키의 경우 개별인증태그의 생성시마다 이전의 값이 추측불가능한 일방향 해시함수, $H(.)$에 의해 갱신되며, 갱신 이전의 키는 시스템 상에서 안전하게 삭제되어야 한다. 이를 Key Evolution이라고 하며 $K_i = H(K_{i-1})$ 과 같이 표현할 수 있다.

인증태그 검증의 경우 검증 방법에 따라 몇 가지로 분류할 수 있다. 먼저, 검증의 대상에 따라 개별검증(Individual Verification)과 누적검증(Aggregated Verification)으로 구분된다. 개별 검증의 경우 개별인증태그를 통해 해당 로그 레코드의 무결성을 검증한다. 누적 검증의 경우 누적인증태그를 통해 로그 레코드들의 무결성을 검증한다. 전술한 바와 같이, 로그 분석장치에서 부분검증이 필요할 수 있으며, 본 논문에서 제안할 방법은 이에 초점을 맞추고 있다.

두 번째로 검증 과정에서 키의 공유 방법에 따라 내부 검증(Private Verification)과 공개 검증(Public Verification)으로 구분된다. 공개키 방식의 경우, 서명 확인용 공개키가 누구에게나 공개 가능함에 따라 공개 검증이 가능하다. 그러나 대칭키 방식의 경우 로그 분석장치에 해당키가 전달되면 인증태그 생성키와 검증용 키가 동일하기 때문에 조작이 가능하므로 공개 검증이 허용되지 않는다. 이에 따라 대칭키 방식의 경우 인증태그를 Trusted Server에 보내어 내부 검증된 결과를 로그 분석장치에서 받는 방식으로 검증이 진행된다.
마지막으로, 무결성 검증에 있어서 온라인으로 연결된 별도의 서버의 도움을 받는지에 따라 온라인 검증(Online Verification)과 오프라인 검증(Offline Verification)으로 나눌 수 있다. 대칭키 방식의 경우 전술한 바와 같이 검증시 Trusted Server의 도움을 받으므로 온라인 검증이 되고, 공개키 방식의 경우 로그 저장장치로부터 검증용 공개키를 함께 받으면 외부의 도움 없이 자체적으로 검증이 가능하므로 오프라인 검증이 가능하다.

3. 기존 연구

3.1 표기법

이후 설명할 기존 기법 들은 아래 <표-1>과 같은 표기법을 따른다.

<표 1> 기존 연구 표기법

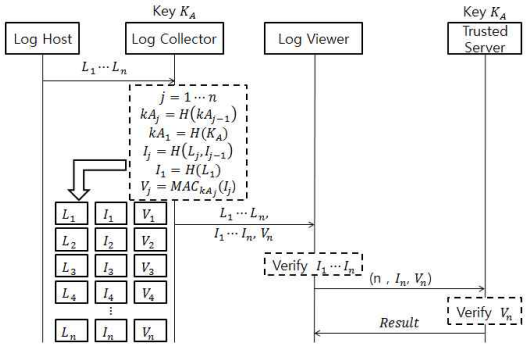
표기법	정의
Log Host	로그 생성장치
Log Collector	로그 저장장치
Log Viewer	로그 분석장치
Trusted Server	Key 공유를 위한 안전한 서버
L_j	로그생성장치에서 생성된 로그 레코드
kA_j, kB_j	인증태그 생성에 사용되는 키
K_A, K_B	키 생성에 사용되기 위해 공유된 Initial Key
I_j	개별인증태그
V_j, T_j	누적인증태그
$H(.)$	일방향 해시함수
$MAC_K(.)$	메시지 인증 코드 생성 함수
$Result$	Trusted Server에 의해 검증된 결과 값
$Request$	Trusted Server에 키를 요청

3.2 Schneier-Kelsey 기법

대칭키 기반의 Forward-Secure 무결성 보장 기법 중 가장 간단하면서도 효율적인 방법은 Kelsey와 Schneier에 의해 연구된 기법이다 [8]. 로그 시스템 작동에 앞서 로그 저장장치는 Trusted Server와 안전한 방식으로 초기 키(Initial Key) K_A 를 공유한다. 공유된 K_A는 Key Evolution에 의해 인증태그가 생성될 때마다 일방향 해시함수를 사용해 갱신되며, 이전 키는 시스템 상에서 안전하게 제거된다.

로그 저장장치는 전달받은 로그 레코드에 대하여 이전 개별인증태그와 같이 해시함수를 적용하여 개별인증태그를 만든다. 이렇게 만들어진 개별인증태그는 갱신된 키와 함께 메시지 인증 코드 함수를 적용하여 누적인증태그를 생성한다. 이렇게 저장된 로그 파일은 분석시 로그 레코드, 개별인증태그, 마지막 누적인증태그를 로그 분석기에 전달하여 분석작업 전에 무결

성 검증을 받게 된다. 개별인증태그의 경우 로그분석기에서 자체적으로 검증하고, 누적인증 태그의 경우 Trusted Server에 로그의 개수, 마지막 개별인증태그, 마지막 누적인증태그를 전달하여 검증결과를 받는다. 이 기법을 그림으로 표현하면 (그림 3)과 같다.



(그림 3) Schneier-Kelsey 기법

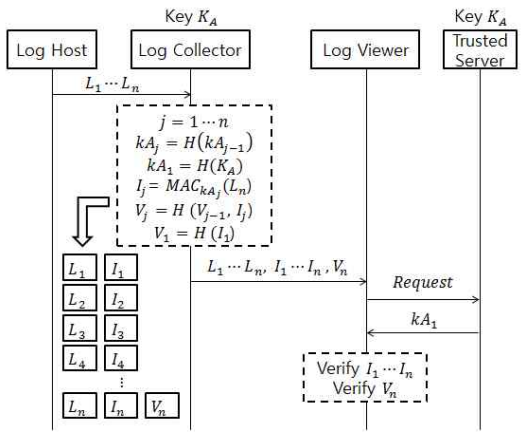
이 기법의 경우, 개별인증태그끼리 선형해시체인을 형성함으로써 순서변경 공격으로부터 인증태그들의 무결성을 보장한다. 또한, 개별인증태그에 대한 메시지 인증 코드를 생성함으로써 각 개별인증태그에 대한 무결성을 확보한다. 이 기법의 가장 큰 특징은 대칭키 기반의 Forward-Secure 무결성 보장 기법이면서 누적검증을 Trusted Server에 맡김으로서 공개검증이 가능하다는 점이다.

이 방법의 장점은 비교적 적은 계산량으로 인증태그들을 생성할 수 있으며, 이에 따라 검증시간이 비교적 짧다는 것이 장점이다. 단점은 누적인증태그와 개별인증태그를 모두 저장함으로써, 다른 기법보다 2배 정도의 인증태그 저장공간을 필요로 하며, 그만큼 인증태그 생성시간이 길다는 것이다.

3.3 Ma-Tsudik 기법

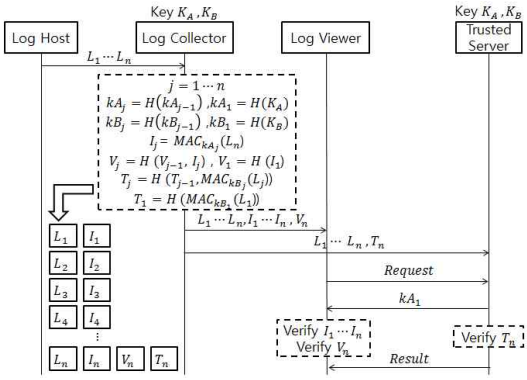
다른 대칭키 기반 Forward-Secure 무결성 보장 기법은 Ma 와 Tsudik에 의해 발표된 기법이다 [9]. 이 기법은 앞의 3절에서 설명한 Schneier-Kelsey 기법과는 반대되는 방법으로 인증태그를 형성하는 기법이라고 할 수 있다. 이 기법 역시 Trusted Server와 초기키를 공유하며 Key Evolution에 의해 인증태그 생성시 키가 갱신된다. 로그 저장장치는 전달받은 로그레코드를 갱신된 키를 사용하여 메시지 인증코드를 만든다. 이렇게 생성된 개별인증태그는 이전 누적인증태그와 함께 해시함수를 적용하여 누적인증태그를 생성한다.

이렇게 저장된 로그 파일은 분석시 로그 레코드, 개별인증태그, 마지막 누적인증태그를 로그 분석기에 전달하여 분석작업 전에 무결성 검증을 받게 된다. 로그 분석기는 Trusted Server로부터 키를 다운받아 개별 및 누적인증태그를 자체적으로 검증한다. 이를 그림으로 표현하면 (그림 4)와 같다.



(그림 4) Ma-Tsudik 기법 1

이 기법의 경우 앞의 3.2절에서 소개한 Schneier-Kelsey 기법과는 반대로 로그항목에 메시지 인증 태그를 생성하여 개별인증태그로 사용하고 개별인증태그를 누적하여 해시함으로써 누적인증태그를 생성한다. 즉, 누적인증태그는 개별인증태그를 해시체인으로 연결한 전체 개별인증태그들의 인증 값이 된다. 하지만, 이 기법의 경우 로그 분석장치가 직접 인증태그 생성에 사용된 키를 다운받아 검증하기 때문에 보안 요구사항 중 내부자 위협에 취약하다는 단점이 있다. 인증태그 생성에 사용된 키와 검증을 위해 로그 분석장치에게 제공된 키가 같은 키이기 때문이다. 그래서 Ma와 Tsudik는 이에 대한 보완책을 제시한다 [10].이를 그림으로 표현하면 (그림 5)와 같다.



(그림 5) Ma-Tsudik 기법 2

이 기법의 경우, 기본적인 태그의 생성 원리는 이전 방법과 동일하다. 다른 점은 로그 저장장치가 Trusted Server와 추가적인 키를 공유함으로써 로그 분석장치에 제공되는 누적인증태그 V_n과 별도로 T_n을 생성한다는 점이다. 추가적으로 공유된 키 K_B 역시 Key Evolution에 의해 매번 갱신된다. 그렇게 되면 로그 분석장치에 있는 내부공격자 또는 크래커에 의해 키 kA_1이 노출되어 로그 분석장치의 V_n 값 및 로그 항목들이 변조 되더라도 Trusted Server에 의해 검증될 T_n은 변조할 수 없으므로 로그 저장장치에 저장된 로그항목들의 Forward -Secure 무결성이 보장된다.

또한, 검증용으로 공개될 수 있는 키가 존재함으로 대칭키 방식이지만 공개검증이 가능하다. 이 방식의 단점은 T_n의 검증에 걸리는 시간과 긴 인증태그 생성시간이다. 두 개의 키를 통해 두 개의 누적인증태그를 생성하기 때문에 인증태그 생성시간이 타 기법에 비해 길다. 또한, T_n의 검증시 V_n의 I_n에 해당하는 $MAC_{kB_j}(L_j)$값을 구해야 하므로 검증시간이 타 기법에 비해 오래 걸린다.

4. 제안 기법

로그 분석장치에서 분석 결과 공격자의 침입 및 공격이 의심될 경우 해당 분석에 사용된 로그들에 대해서만 부분적으로 재검증이 필요할 수 있다. 이 제안기법의 원형이 되는 Schneier-Kelsey 기법의 경우 인증태그들이 선형 해시체인을 형성함에 따라 부분검증시 처음부터 해당 부분까지 모두 검증해야 해당 부분의 인증태그에 대한 검증이 가능하다. 본 절에서는 이를 보완하기 위한 Merkle Tree 기반의 인증태그 생성기법을 제안한다.

4.1 표기법

본 장에서 제안할 제안기법의 표기법은 기본적으로 이전 장에서 소개한 표기법을 따른다. 추가적으로 Merkle Tree 방식에 의해 생성된 노드 $M_{a,d}$는 $H(M_{a,b}, M_{c,d})$와 같이 표현할 수 있다. ***a, b, c, d*** 는 노드를 구성하는데 사용된 로그 레코드의 번호를 나타내며, ‘,’앞의 숫자는 해당 레코드가 포함하는 가장 낮은 레코드의 번호이고 뒤의 숫자는 가장 높은 번호이다. 로그 레코드를 최하위 Leaf 노드로 하며, 로그 레코드의 개수가 ***n***개일 때 이 최하위 Leaf 노드로부터 Root 노드인 $M_{1,n}$이 생성될 때 까지 같은 방식을 반복한다. 이렇게 생성된 상위노드들의 집합을 $Merkle(L_1 \cdots L_n)$라고 표기한다.

4.2 Merkle Tree

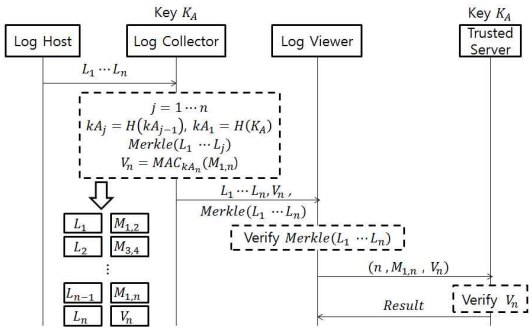
먼저 제안기법의 소개에 앞서 제안기법의 기본 아이디어가 되는 Merkle Tree에 대해 소개하고 그 원리를 알아보도록 한다. Hash Tree 라고도 불리는 Merkle Tree는 일회용 서명인 Lamport 서명을 효과적으로 다루기 위해 Ralph C. Merkle에 의해 제안되었다[11].

현재는 여러 블록으로 나뉜 데이터의 무결성을 보장하는 용도로 주로 사용된다. 트리 구조의 일종이지만 그 생성과정에 있어 일반적인 트리와는 다른 모습을 보인다. Root 노드로부터 파생되어 상위노드로부터 자식노드로 연결되는 일반적인 트리와 달리 Merkle Tree는 파일 등의 데이터를 최하위 Leaf 노드로 하여 최상위 노드인 Root 노드를 구해내는 방식이다. 상위 노드는 각각 자식 노드들의 해시 값이 된다.

예를 들어, $M_{1,2}$ 는 $H(M_{1,1}, M_{2,2})$과 같이 표현할 수 있다. 이때, 해시함수는 어떤 것이든 사용할 수 있지만, 보통 SHA-1과 같은 충돌저항성이 보장된 해시함수를 사용한다.

4.3 제안기법

본 기법은 앞에서 설명한 Merkle Tree와 Schneier-Kelsey 기법을 응용하여 구성하였다. 본 기법에서는 계산량을 줄이기 위해 $M_{1,1} = H(L_1)$ 이 아닌 $M_{1,1} = L_1$으로 하여 구성하였다. 이를 통해 n번의 해시함수 실행을 줄일 수 있다. 기본 원리는 Schneier-Kelsey 기법과 비슷하나 개별인증태그 대신 상위노드의 집합을 생성하고, 누적인증태그는 Merkle Tree의 Root 노드에 메시지 인증 코드 함수를 Key Evolution에 의해 갱신된 키를 사용하여 적용한다. 분석장치에서의 무결성 검증은 자체적으로 상위노드의 집합을 검증 후 로그의 개수, Root 노드 값, 누적인증태그를 Trusted Server로 전송하여 검증 결과를 받는다. 이를 그림으로 표현하면 (그림 6)과 같다.



(그림 6) 제안 기법

이 방법의 경우, Schneier-Kelsey 기법처럼 대칭키 방식이지만, 공개 검증이 가능하다. 또한, 필요에 의해 부분검증이 필요할 경우 해당 부분과 관련된 노드들만을 사용하여 부분검증이 가능하다. 예를 들어, 10,000 개의 로그 레코드 중 500 ~ 600번, 800 ~ 900번에 있는 로그 레코드의 검증이 필요하다고 가정하자. 제안기법은 $M_{499,500} \cdots M_{599,600}$, $M_{799,800} \cdots M_{899,900}$의 검증을 통해 부분 검증이 가능하다. 그러나 Schneier-Kelsey 기법의 경우, 개별인증태그가 선형의 해시체인 형태를 갖기 때문에 해당 부분의 로그 레코드들만으로 개별인증태그의 일부를 검증할 수 없으므로 처음부터 해당 부분의 마지막인 900번 레코드까지의 모든 개별인증태그를 검증해야 한다. Ma-Tsudik 기법들은 해시체인을 이루는 부분이 누적인증태그이기 때문에 부분검증이 가능하나, 메시지 인증코드 검증을 위해 900번의 키 갱신이 필요하여 더 많은 시간이 걸리게 된다. 다음 장에서는 지금까지 소개 및 제안한 기법들을 분석하고 실제 구현을 통한 성능 비교를 해보도록 하겠다.

5. 비교 및 분석

5.1 보안 요구사항 비교 및 분석

본 절에서는 2장에서 언급한 보안 위협모델에 대해 기법들 간의 장,단점을 비교 및 분석할 것이다. 먼저, 이전 장들에서 소개 및 제안한 기법들의 경우 모두 개별 및 누적인증태그를 통해 기본적인 변조, 삽입, 삭제 공격에 대해 Forward-Secure 무결성을 보장한다. 또한, 로그들의 순서를 무작위로 바꿈으로서 로그파일의 의미를 변조하는 순서변경 공격의 경우, 모든 기법들이 각자만의 해시체인을 통해 Forward -Secure 무결성을 보장한다. 제안 기법의 경우 트리형 해시체인을 사용하며, 기존 기법들은 선형 해시체인을 사용한다. Schneier-Kelsey 기법의 경우, 공개검증이 가능하며, 개별인증태그와 누적인증태그가 모든 레코드에 있으므로 부분검증 및 부분누적검증이 가능하다.

그로 인해 절단공격에 취약한 면을 보인다. Ma-Tsudik 기법 1은 공개검증이 불가능 하지만 Ma-Tsudik 기법 2의 경우 이를 보완하여 공개검증이 가능하다. 개별인증태그로 부분검증은 가능하지만, 마지막 누적인증태그 값만을 저장하기 때문에 부분누적검증은 불가능하다. 제안기법의 경우 Schneier-Kelsey 기법을 기반으로 함으로써 공개검증이 가능하고 트리구조의 특성상 부분검증 및 부분누적검증이 용이하다. 대신 인증태그간의 트리관계를 저장할 인덱스 파일 또는 추가 계산량을 필요로 하는 단점이 있다. 이를 표로 정리하면 <표 2>와 같다. 표의 기존연구 1은 Schneier-Kelsey 기법을 기존연구 2는 Ma-Tsudik 기법 1을 기존연구 3은 Ma-Tsudik 기법 2를 의미하다.

<표 2> 보안 요구사항 비교 및 분석

	기존 연구1	기존 연구2	기존 연구3	제안 기법
변조, 삽입, 삭제	○	○	○	○
순서변경	○	○	○	○
절단공격	×	○	○	○
공개검증	○	×	○	○
부분검증	○	○	○	○
부분누적검증	○	×	×	○
온라인/오프라인 검증	온라인	오프라인	온라인	온라인

5.2 구현 및 성능분석

앞서 소개한 기법들은 저장 구조와 사용 함수들에 따라 많은 성능 차이를 보인다. 본 절에서는 이러한 기법들의 처리시간 및 메모리 효율을 비교한다. 구현 환경은 프로세서는 Intel i5-5200U 2.20GHz를 사용하였고 메모리는 8GB이다. 개발언어로는 JAVA를 사용하였고, JDK 1.8.0_91을 사용하였다. 해당 기법들의 구현에 사용되는 해시함수는 SHA-256으로 통일하였고, 메시지인증코드 함수 또한 Hmac-SHA-256 방식을 사용하였다. 이렇게 구성할 경우 함수들에 의해 생성된 결과물의 길이가 같아지므로 공격자는 해당 태그가 어떤 방식으로 생성되었는지 추측이 힘들어 진다.

syslog 표준상 한 로그레코드의 길이는 2048Byte로 제한하고 있다 [3]. 그러나 사실상 로그레코드가 이 길이를 다 쓰는 경우는 드물다. 이에 따라, 본 구현환경의 테스트를 위한 로그파일의 레코드 1개는 1024Byte로 하여 구성하였다.

5.2.1 메모리 요구사항

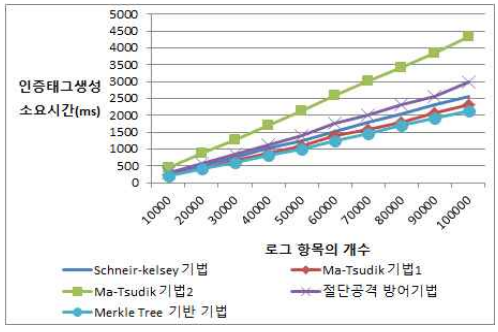
가장 먼저 분석할 내용은 각 기법의 메모리 요구사항이다. Schneier-Kelsey 기법의 경우 누적인증태그와 개별인증태그를 모두 저장하기 때문에 타기법에 비해 2배정도의 추가 메모리를 소모한다. 제안기법의 경우 키 및 인증태그의 저장에는 가장 적은 메모리를 소모하지만 인증태그의 관계를 따로 파일로 저장할 경우 추가적인 메모리를 소모하게 된다. 이를 표로 정리하면 <표 3>과 같다.

<표 3> 기법별 메모리 요구사항

로그 항목 = n 개	기존 연구1	기존 연구2	기존 연구3	제안 기법
개별인증태그	n	n	n	n-1
누적인증태그	n	1	2	1
키	1	1	2	1
합계	$2n$+1	n+2	n+4	n+1

5.2.2 인증태그 생성시간 비교

다음은 각 기법의 인증태그의 생성 및 저장에 필요한 시간에 대한 비교이다. 측정은 로그 저장장치가 로그항목들을 받은 후 개별인증태그의 생성부터 누적인증태그의 생성까지를 측정하였으며, 키 갱신 시간도 여기에 포함된다. 로그 항목의 개수별 100번을 돌린 값의 평균을 측정하였으며, 이를 정리하면 (그림 7)과 같다.

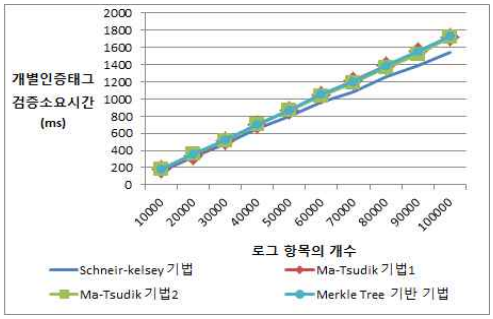


(그림 7) 인증태그 생성시간 비교

5.2.3 인증태그 검증시간 비교

인증태그 검증의 경우 3가지 유형의 검증에 대해 진행할 것이다. 먼저, 개별 검증의 경우 로그 분석장치에서 누적검증 이전에 로그 저장장치로부터 전달된 개별인증태그의 검증 시간을 측정한다. 이때, 제안 기법의 경우는 개별인증태그 대신 전달되는 상위노드 집합의 검증 시간을 측정한다. 누적검증의 경우 로그 분석장치 또는 Trusted Server에서 누적 검증태그 검

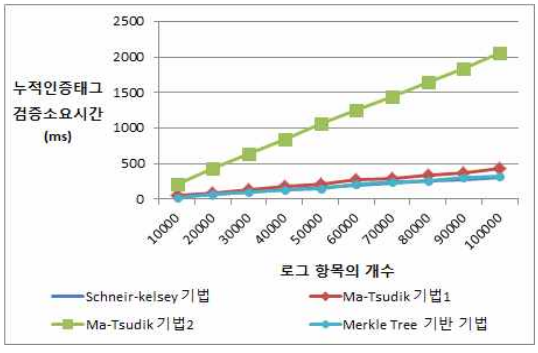
증에 걸리는 시간을 측정한다. 마지막으로 보안 요구사항 중 하나인 부분 검증의 경우 로그 항목들 중 특정 부분에 대한 개별인증태그의 검증이 필요할 경우 이에 걸리는 시간을 측정하였으며, 여기서는 1000개의 로그 중 500~600번째에 대한 검증을 측정한다.



(그림 8) 개별인증태그 검증시간 비교

(그림 8)은 로그 개수의 증가에 따른 개별인증태그 검증시간의 차이를 나타낸다. 그림에서도 보이듯이 이는 그리 많은 차이를 보이지 않는다. 여기서 고려할 점은 다른 선형리스트 형태의 인증태그 구조를 사용하는 타기법의 경우 검증시간을 줄일 방법이 없다.

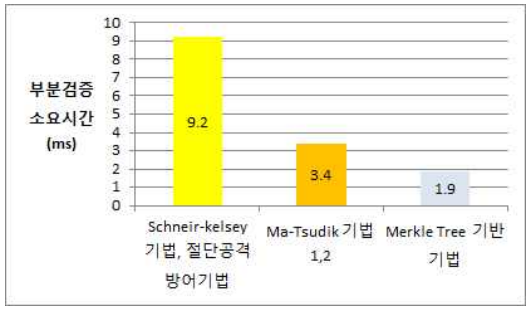
그러나 제안 기법의 경우 트리구조의 특성상 멀티스레딩으로 동시에 계산이 가능하기 때문에 다른 기법보다 개별인증 태그의 검증에 더 적은 시간을 소요할 수 있다. 물론 멀티스레딩 적용시 해당 장치의 자원에 따라 그 속도가 더 느려질 수도 있기 때문에 본 비교에서는 다른 기법들과 마찬가지로 멀티스레딩이 아닌 싱글스레드로 검증하였다.



(그림 9) 누적인증태그 검증시간

그러나 (그림 9)에서 보이듯이 누적인증태그의 검증의 경우, 많은 차이를 보인다. 유독 많은 시간이 걸리는 Ma-Tsudik 기법 2 의 경우, Ma-Tsudik 기법 1과 똑같이 생성된 누적인증태그 V_n의 경우 비슷한 시간이 걸리지만 다른 키로 생성된 누적인증태그 T_n의 경우는 다르다. T_n은 Trusted Server에 로그 레코드와 T_n을 전송하여 그 검증결과를 받는데, 키 K_B로 생성된 개별인증태그를 생성하며 검증해야하기 때문에 더 많은 시간이 걸린다.

Ma-Tsudik 기법 1의 경우 생성된 개별인증태그를 해시체인 형태로 해시함수를 적용하여 누적인증태그를 생성한다. Schneier-Kelsey 기법과 제안 기법의 경우 Trusted Server로 로그 저장장치로부터 받은 누적인증태그, 레코드의 개수, 로그 분석장치에서 검증한 누적인증태그 생성에 필요한 재료값을 전송하여 해당 값을 검증받는다. 여기에는 Trusted Server에서의 키 갱신 시간도 포함된다. 오프라인 검증인 Ma-Tsudik 기법 1과는 달리 전송에 약간의 시간이 소요되지만, 그 전송량이 적으므로 본 비교에서는 전송시간을 무시한다.



(그림 10) 기법별 부분검증시간 비교

(그림 10)은 기법별 부분검증시간을 나타낸 그림이다. Schneier-Kelsey 기법과 절단공격 방어기법의 경우 개별인증태그 I_n의 검증을 위해서는 해시체인이 연결돼 있는 첫 레코드부터 개별인증태그를 생성하여 해당 부분을 검증해야하기 때문에 사실상 부분검증이 힘들다. Ma-Tsudik기법들의 경우, 개별인증태그가 메시지 인증코드 함수로 생성되었으므로, 검증을 위해서는 Trusted Server로부터 키를 요청해야 하며, 이

를 사용하여 부분검증이 가능하다. 제안 기법의 경우, 개별인증 태그 역할을 하는 상위노드들 중 해당 부분에 해당하는 레코드들의 바로 위의 상위노드만 비교하여 부분검증이 가능하다. 위 그림은 1000개의 레코드 중 중간의 레코드들을 대상으로 부분검증을 하였기 때문에 Merkle Tree 기반 기법이 더 빨랐지만, 부분검증의 위치가 레코드의 앞쪽으로 갈수록 선형리스트 형태의 해시체인에 기반한 Schneier-Kelsey 기법은 계산량이 줄어들므로 부분검증시간이 짧아질 수 있다.

또한 Ma-Tsudik 기법들은 키 갱신 시간이 줄어들기 때문에 더 빨라질 것이다. 그러나 반대로 레코드의 뒤쪽으로 갈수록 그 계산량이 늘어나므로 부분검증에 소요되는 시간도 더 길어진다. Merkle Tree 기반 기법의 경우 부분검증을 해야 할 부분이 레코드의 앞이던 뒤이던 부분검증 부분의 범위에 의해서만 검증시간이 달라질 뿐이다.

6. 결 론

빅데이터에 대한 관심이 높아짐에 따라, 로그 통합관리 시스템이 여러 방면에서 다시 재조명 되고 있다. 빅데이터를 효율적으로 분석하는 것도 중요하지만 분석한 데이터의 무결성이 보장되지 않는다면, 그 분석은 틀린 분석이 될 것이다. 본 논문에서는 현재 로그관리 인프라를 통해 로그를 수집하는 로그 저장장치의 Forward-Secure 무결성을 보장하기 위한 기법들을 분석했다. 또한, Merkle Tree를 이용하여 기존 연구들의 특징은 그대로 유지하고, 부분검증의 용이성과 신속한 인증태그 생성시간 등의 추가적인 장점이 있다. 또한, 트리 구조의 특성상 멀티스레딩이 가능한 점을 통해 검증 및 생성시간을 추가적으로 줄일 수 있다.

로그의 경우 다양한 종류의 장치에서 생성되는 만큼 다양한 요구사항과 응용환경이 존재한다. 이 때문에 모든 장치에 적합한 기법은 존재할 수 없으므로, 다양한 요구사항과 응용환경에 대한 꾸준한 연구가 필요하다. 또한, Forward-Secure 무결성의 경우 공격자의 침입 이전의 무결성만을 보장하므로, 암호화와 동시에 분석이 가능하게 해주는 동형암호에 대한 연구를 통해 공격자의 침입과 관계없이 로그 데이터의 무결성을 보장할 수 있는 방법에 대한 연구가 더 필요할 것이라 판단된다.

참고문헌

[1] Chuvakin, A., Schmidt, K., & Phillips, C. (2012). Logging and log management: the authoritative guide to understanding the concepts surrounding logging and log management. Newnes.

[2] Malasri, K., & Wang, L. (2009). Securing wireless implantable devices for healthcare: Ideas and challenges. IEEE Communications Magazine, 47(7), 74-80.

[3] Gerhards, R. (2009). The syslog protocol. RFC.5424

[4] Bellare, M., & Yee, B. (1997). Forward integrity for secure audit logs (Vol. 184). Technical report, Computer Science and Engineering Department, University of California at San Diego.

[5] Hsu, W. W., & Ong, S. (2007). WORM storage is not enough [Technical Forum]. IBM Systems Journal, 46(2), 363-369.

[6] 강석규, & 박창섭. (2015). 전방 안전성이 보장되는 로그 시스템 보안기법 비교분석. 융합보안논문지, 15(7), 85-96.

[7] 강석규, & 박창섭. (2015). 키 지연 노출에 기반을 둔 로그 전송을 고려한 로그 저장 기법. 융합보안논문지, 15(5), 37-45.

[8] Schneier, B., & Kelsey, J. (1999). Secure audit logs to support computer forensics. ACM Transactions on Information and System Security (TISSEC), 2(2), 159-176.

[9] Ma, D. & Tsudik, G. (2007, May). Forward-secure sequential aggregate authentication. In 2007 IEEE Symposium on Security and Privacy (SP'07) (pp. 86-91). IEEE.

[10] Ma, D. & Tsudik, G. (2009). A new approach to secure logging. ACM Transactions on Storage (TOS), 5(1), 2.

[11] Merkle, R. C. (1989, August). A certified digital signature. In Conference on the Theory and Application of Cryptology (pp. 218-238). Springer New York.

[저 자 소 개]

이 정 엽 (Jung-yeob Lee)
2015년 2월 단국대학교
컴퓨터과학과 학사
2017년 2월 단국대학교
컴퓨터학과 석사

email : youp89@naver.com

박 창 섭 (Chang-seop Park)
1983년 2월 연세대학교
경제학과 학사
1987년 2월 Leigh University
컴퓨터과학과 석사
1990년 2월 Leigh University
컴퓨터과학과 박사
1990년 3월 ~ 현재 단국대학교
컴퓨터학과 교수

email : csp0@dankook.ac.kr