스트림 인증에 적합한 일회용 서명 기법

박용수, 조유근

서울대학교, 전기.컴퓨터 공학부

An Efficient One-time Signature Scheme for Stream Authentication

Yong-su Park, Yoo-kun Cho

School of Computer Science and Engineering, Seoul National Univ.

요 약

본 논문에서는 스트림 데이터를 인증하는데 적합한 일회용 서명 기법을 제시한다. 스트림 인증에 일회용 서명 기법을 사용할 경우, 가장 큰 문제점은 큰 서명 크기로 인해 네트워크 오버헤드가 많다는 점이다. 제시한 기법은 기존 일회용 서명 기법 중 서명 크가 가장 작다. 또한, 제시된 기법의 검증 연산량은 매우 작다. 스트림 인증에 적합한 기존 기법과 온라인 서명 연산량을 비교하면, HORS보다는 다소 많지만, BiBa나 Powerball보다 적다.

I. 서론

1. 스트림 서명에 적합한 일회용 서명 기법[1]

1) 연구배경

인터넷 사용자가 늘어나고 망 대역폭이 커짐에 따라 인터넷을 통하여 오디오나 비디오 같은 스트림 데이터를 제공하는 서비스가 늘어나고 있다. 또한, 인터넷이 상업적인 용도로 이용되면서 이런 서비스들도 점차 유료화되고 있는 추세이다.

상업적인 스트림 서비스가 널리 사용되기 위해서는 보안이 무엇보다도 중요하다. 일례로, 방송국에서는 임의의 해커가 방송 데이터에 상업적인 내용을 삽입하는 행위를 방지하고 싶어할 것이며, 방송 청취자는 증권 정보나 뉴스가 위/변조되지 않았음을 확인하고 싶어할 것이다. 이렇듯, 여러 가지 보안 요소 중 데이터의 위/변조를 방지하고 부인 방지를 제공하는 인증 기법이 무엇보다 중요하다 [2].

실시간으로 생성/전달되는 스트림 데이터를 인증하는 문제는 일반 데이터를 인증하는 것과 다르다. 가장 큰 문제점은 데이터의 크기가 상당히 크다는 점이다. 일반적으로 전자 서명이 많은 계산량을 요구하는 작업임을 생각해볼 때, 모든 데이터를 서명하는 것은 송신자에게 상당한 부하를 준다.

일회용 서명 기법은 일반적인 전자 서명 기법보다 서명 연산량이 매우 작아 스트림 인증에 적합하지만, 서명 크기가 커서 높은 통신 오버헤드를 야기하는 단점이 있다 [3], [5]. 일례로, SHA-1을 사용하는 경우, Lamport 일회용 서명 기법의 서명 크기는 1600 byte이다. 서명될 스트림 청크의 크기가 통상 512 byte보다 작은 것을 생각해볼 때 [2], 통신 오버헤드가 상당히 큰 것을 알 수 있다. 기존 연구 중 [5]는 기존 일회용 서명 기법의 서명 크기를 줄이는 방법을 고안하였으며, [1], [3], [4]는 서명 크기가 작은 일회용 서명 기법을 제시하였다.

본 논문에서는 스트림 인증에 적합한 효율적인 일회용 서명 기법을 제시한다. 우리는 HORS (Hash to Obtain Random Subset) [4]를 개선하여, 서명 크기를 줄였다. 기존 일회용 서명 기법과 비교해 볼 때, 제시된 기법은 가장 작은 서명 크기를 가진다. 또한, 제안 기법은 검증 연산량이

1 이 논문은 2001년도 두뇌한국21사업에 의하여 지원되었음.

매우 작다. 온라인 서명 연산량은 HORS보다 다소 많지만, 기존 기법 중 스트림 인증에 적합한 기법인 BiBa [3], Powerball [1]보다는 적다.

2) 표기법

본 논문에서 사용하는 용어의 의미는 다음과 같다. $f(x)$는 l 비트 출력 크기를 가지는 일방향 함수를 의미하며, $h(x)$는 일방향 해쉬 함수를 뜻한다 (출력 크기는 적어도 $k\log_2 t$ 비트보다 커야하며 k, t는 II-1 절에서 설명한다). 또한, C||D는 데이터 C와 D를 연결시킨 (concatenate) 것을 의미한다. $f(x)$와 $h(x)$는 표준 해쉬 함수로 (SHA-1 혹은 RIPEMD-160) 구현될 수 있다 [4].

II. 본문

1. HORS [4]

HORS는 키 생성, 서명 생성, 그리고 서명 검증의 3 가지 모듈로 구성된다. 첫째, 키 생성 모듈은 다음과 같다. 입력 매개변수 l, k, 그리고 t에 대하여, 서명자는 t 개의 l 비트 스트링 $s_1, s_2, \cdots, s_t$를 생성한다. 그 후, 서명자는 $v_i=f(s_i)$를 계산한다 ($1\le i\le t$). 비밀키와 공개키는 각각 $SK=(s_1, s_2, \cdots, s_t)$, $PK=(v_1, v_2, \cdots, v_t)$이다. 둘째, 서명 생성 모듈은 다음과 같다. 서명될 메시지 m, 비밀키 SK가 있을 때, 서명자는 $h(m)$을 $h_1, h_2, \cdots, h_k$로 나눈다 (각 h_j ($1\le j\le k$)는 $\log_2 t$ 비트이다). [4]에서는 $h(m)$을 어떤 방식으로 나누어 h_j를 생성하는지 정확히 설명하지 않았지만, [2]에서 언급하였듯이, $h(m)= h_1||h_2||\cdots||h_k||\cdots$의 방법이 사용될 수 있다. h_j를 정수값 i_j로 해석하면 ($1\le j\le k$), 메시지 m에 대한 서명값은 $SIG=(s_{i1}, s_{i2}, \cdots, s_{ik})$이다. 셋째, 서명 검증은 다음과 같다. 메시지 m, 서명값 $SIG=(s_{i1}, s_{i2}, \cdots, s_{ik})$, 그리고 공개키 $PK=(v_1, v_2, \cdots, v_t)$에 대하여, 검증자는 $h(m)$을 $h_1, h_2, \cdots, h_k$로 나눈다. 그는 h_j를 정수 i_j ($1\le j\le k$)로 해석한다. 만일 모든 j에 대해 ($1\le j\le k$), $f(s_{ij})=(v_{ij})$를 만족할 때, 그는 서명을 용인 (accept)한다.

일례로, l=80, k=3, t=8일 경우, 첫째, 서명자는 비밀키 $SK=(s_1, s_2, \cdots, s_8)$을 생성한다. 이 때, 각 s_i ($1\le i\le 8$)는 80-비트 스트링이다. 그 후, 서명자는 공개키 $PK=(v_1, v_2, \cdots, v_8)$을 생성한다 ($v_i=f(s_i)$ ($1\le i\le 8$)). 메시지 m에 대하여, 서명자는 $h(m)= h_1||h_2||h_3||\cdots$을 계산한 후, 각 h_j를 정수 i_j로 해석한다 ($1\le j\le 3$). 일례로, i_1=2, i_2=3, i_3=5라고 가정하자. 그 후, 서명자는 메시지 m에 대한 서명값 $SIG=(s_2, s_3, s_5)$를 생성한다. 메시지 m, 서명값 $SIG=(s_2, s_3, s_5)$, 그리고 공개키 $PK=(v_1, v_2, \cdots, v_8)$에 대하여, 검증자는 $h(m)=h_1||h_2||h_3||\cdots$을 계산한 후, 각 h_j를 정수 i_j로 해석한다. 만일 m, PK, SIG 값이 변경되지 않았다면, i_1=2, i_2=3, i_3=5가 될 것이다. 만일 $f(s_2)=(v_2)$, $f(s_3)=(v_3)$, $f(s_5)=(v_5)$이면 검증자는 서명값을 용인 (accept)한다.

2. 제안 기법

HORS에서 공격자가 메시지 m에 대한 서명값 $SIG=(s_{i1}, s_{i2}, \cdots, s_{ik})$를 가지고 있다고 가정해보자. 만일 그가 $h(m')=h_2||h_3||h_1||\cdots$ 혹은 $h(m'')=h_3||h_2||h_1||\cdots$을 만족하는 m'이나 m''을 발견하면 m'에 대한 서명값 $(s_{i2}, s_{i3}, s_{i1}, \cdots, s_{ik})$ 혹은 m''에 대한 서명값 $(s_{i3}, s_{i2}, s_{i1}, \cdots, s_{ik})$을 위조할 수 있다. 본 논문의 주 아이디어는 제한 조건을 추가함으로써 HORS의 이 약점을 보완하는 것이다. 그 결과, II-3 절에서 언급하겠지만, 제안된 기법은 기존 기법보다 작은 서명 크기를 가짐에도 불구하고 기존 기법들과 거의 같은 수준의 안전성을 가진다.

HORS와 마찬가지로 제안된 기법도 키 생성, 서명 생성, 그리고 서명 검증의 3 가지 모듈로 구성된다. 첫째, 키 생성은 다음과 같다. 서명자는 t 개의 l 비트 스트링 $s_1, s_2, \cdots, s_t$를 생성한다. 그 후, 서명자는 $s_i'=f(s_i)$, $v_i=f(s_i')$를 계산한다 ($1\le i\le t$). 비밀키와 공개키는 각각 $SK=((s_1, s_2, \cdots, s_t), (s_1', s_2', \cdots, s_t'))$, $PK=(v_1, v_2, \cdots, v_t)$이다. 둘째, 서명 생성은 다음과 같다. 메시지 m과 비밀키 SK가 있을 때, 서명자는 임의의 값 c를 선택한 후, $h(m||c)= h_1||h_2||\cdots||h_k||\cdots$를 계산한다. 각 h_j는 정수값 i_j로 해석된다 ($1\le j\le k$). 만일 i_j ($1\le j\le k$)가 조건 $i_1<i_2<\cdots<i_{k/2}$, $i_{k/2+1}<i_{k/2+2}<\cdots<i_k$, $\{i_j|(1\le j\le k/2)\}\cap\{i_j|(k/2+1\le j\le k)\}=\varnothing$를 만족시키지 못하면, 서명자는 c를 다른 값으로 다시 선택한 후, 위 작업을 반복한다. 그 결과, 서명자는 앞에서 언급한 조건을 만족하는 c, $i_1, i_2, \cdots, i_k$를 얻는다. 서명값은 $SIG=(c, (s_{i1}, s_{i2}, \cdots, s_{ik/2}), (s'_{ik/2+1}, s'_{ik/2+2}, \cdots, s'_{ik}))$이다. 셋째, 서명 검증은 다음과 같다. 메시지 m, 서명값 $SIG=(c, (s_{i1}, s_{i2}, \cdots, s_{ik/2}), (s'_{ik/2+1}, s'_{ik/2+2}, \cdots, s'_{ik}))$, 그리고 공개키 $PK=(v_1, v_2, \cdots, v_t)$에 대하여, 검증자는 $h(m||c)=h_1||h_2||\cdots||h_k||\cdots$를 계산한다. 각 h_j는 정수값 i_j로 해석된다 ($1\le j\le k$). 만일 i_j ($1\le j\le k$)가 조건 $i_1<i_2<\cdots<i_{k/2}$, $i_{k/2+1}<i_{k/2+2}<\cdots<i_k$, $\{i_j|(1\le j\le k/2)\}\cap\{i_j|(k/2+1\le j\le k)\}=\varnothing$를 만족시키지 못하면, 서명 검증은 실패한다. ($1\le j\le k/2$)에 대하여 $f(f(s_{ij}))=(v_{ij})$가 만족되고, ($k/2+1\le j\le k$)에 대하여 $f(s'_{ij})=(v_{ij})$가 만족되면, 검증자는 서명을 용인 (accept)한다.

3. 분석

1) 연산량 분석

[1], [4]와 마찬가지로, 제안된 기법이 필요로 하는 일방 (해쉬) 함수의 호출 횟수를 계산하여 연산량을 분석한다. 먼저, 키 생성은 $f()$를 $2t$ 번 호출한다. 그리고, 서명 검증은 $h()$를 1 번 호출하고, $f()$를 $3k/2$ 번 호출한다. 서명 생성에서는 $h(m||c)$를 조건 $i_1<i_2<\cdots<i_{k/2}$, $i_{k/2+1}<i_{k/2+2}<\cdots<i_k$, $\{i_j|(1\le j\le k/2)\}\cap\{i_j|(k/2+1\le j\le k)\}=\varnothing$를 만족시킬 때까지 호출한다. 평균 호출 횟수를 구하기 위해, 우리는 [4]의 가정처럼 $h()$가 랜덤 오라클 기반 일방 해쉬 함수라고 하고, $i_1, i_2,\cdots, i_k$가 고른 분포 (uniform distribution)을 가진다고 가정할 때, $i_1<i_2<\cdots<i_{k/2}$, $i_{k/2+1}<i_{k/2+2}<\cdots<i_k$, $\{i_j|(1\le j\le k/2)\}\cap\{i_j|(k/2+1\le j\le k)\}=\varnothing$를 만족시키는 확률을 계산한다.

정리 1. $i_1, i_2,\cdots,i_k$가 $[0 \sim t-1]$ 구간에서 고른 분포 (uniform distribution)을 가지는 독립 확률 변수일 때, $i_1<i_2<\cdots<i_{k/2}$, $i_{k/2+1}<i_{k/2+2}<\cdots<i_k$, $\{i_j|(1\le j\le k/2)\}\cap\{i_j|(k/2+1\le j\le k)\}=\varnothing$일 확률은

$$P(k,t)=\frac{\binom{t}{k/2}\binom{t-k/2}{k/2}}{t^k}$$ 이다.

증명. $i_1<i_2<\cdots<i_{k/2}$ 조건을 만족시킬 확률은 $\frac{\binom{t}{k/2}}{t^{k/2}}$ 이며, $i_{k/2+1}<i_{k/2+2}<\cdots<i_k$ 조건을 만족시킬 확률 또한 이와 동일하다. $i_1<i_2<\cdots<i_{k/2}$, $i_{k/2+1}<i_{k/2+2}<\cdots<i_k$ 조건을 만족시킬 때, $\{i_j|(1\le j\le k/2)\}\cap\{i_j|(k/2+1\le j\le k)\}=\varnothing$ 조건을 만족시킬 확률은 $\frac{\binom{t-k/2}{k/2}}{\binom{t}{k/2}}$ 이다. 따라서, 세 조건을 모두 만족시킬 확률은 $\frac{\binom{t}{k/2}\binom{t-k/2}{k/2}}{t^k}$ 이다.

정리 1에 의하여, 서명 생성 시 $h()$의 호출 횟수는 평균 $\frac{1}{P(k,t)}=\frac{t^k}{\binom{t}{k/2}\binom{t-k/2}{k/2}}$ 이다.

2) 서명 크기

서명 크기는 $kl+|c|$이다. l의 크기는 표준 해쉬 함수 SHA-1을 사용할 경우 20 byte이다. c의 크기는 II-3-1) 절에서 분석한 $P(k,t)$와 관련이 있다. 일례로, t=1024, k=8일 경우, 약 $P(k,t)$=0.00169이며, 이는 c 값을 바꾸어 $h(m||c)$를 계산해 볼 때, 평균 592 번에 한 번씩 올바른 서명값을 찾을 수 있다는 뜻이다. 이 경우, $|c|$는 10 bit면 충분하다.

3) 안전성 분석

우리는 [4]처럼, $h()$가 랜덤 오라클 기반 일방 해쉬 함수라고 가정하며, 공격자가 $h()$, $f()$의 충돌을 찾을 확률이 무시할만하다고 (negligible) 가정할 때, 제안된 기법의 안전성을 분석한다.

공격자가 메시지 m에 대하여 서명값 $SIG=(c, (s_{i1}, s_{i2}, \cdots, s_{ik/2}), (s'_{ik/2+1}, s'_{ik/2+2}, \cdots, s'_{ik}))$를 가지고 있다고 가정하자. 공격자가 다른 메시지 m'에 대해 위조 서명을 구할 수 있는 확률을 생각해보면 다음과 같다.

공격자가 m'에 대해 위조된 서명 $SIG'=(c', (s_{i1}, s_{i2}, \cdots, s_{ik/2}), (s'_{ik/2+1}, s'_{ik/2+2}, \cdots, s'_{ik}))$를 계산하기 위해서는 $h(m'||c')=h_1'||h_2'||\cdots||h_k'||\cdots$일 때, $i_1'=i_1<i_2'=i_2<\cdots<i_{k/2}'=i_{k/2}$이고 $i_{k/2+1}'=i_{k/2+1}<i_{k/2+2}'=i_{k/2+2}<\cdots<i_k'=i_k$를 만족시켜야 한다. 따라서, 공격자가 서명을 위조할 확률은$(1/t)^k$이다.

4) 기존 기법과 비교 결과

본 절에서는 기존의 일회용 서명 기법과 제안된 기법에 대하여 서명 크기, 검증 연산량, 서명 연산량 그리고, 공개키 크기에 대해 비교한 결과를 제시한다. 각 일회용 서명 기법의 위조 확률은 $1/2^{80}$ 이하로 설정했으며, 제안된 기법, Powerball [1], HORS에서 t 값은 [1]의 설정에 따라 1024로 정하였다. 비교 결과는 다음과 같다.

기법	서명 크기	검증 연산량
Lamport	80	80
Merkle-Winternitz	23	169
Bleichenbacher-Maurer	45	72
BiBa	11	23
Powerball	10	20
HORS	13	14
Our scheme	8	13

기법	서명 연산량(off-line)	서명 연산량(on-line)	공개키 크기
Lamport	160	1	160
Merkle-Winternitz	355	1	1
Bleichenbacher-Maurer	382	1	1
BiBa	1024	2048	1024
Powerball	2048	2048	1024
HORS	1024	1	1024
Our scheme	2048	592	1024

위 표에서 연산량은 일방향 (해쉬) 함수의 수를 나타낸다. 서명 크기, 공개키 크기는 일방향 (해쉬) 함수 단위이며, 만일 SHA-1 (20바이트)를 사용하는 경우, 각 값에 20을 곱하면 바이트 크기가 된다 (단, 위 표에서 제안된 기법의 서명 크기는 c의 크기를 제외한 값이며, 이를 추가하면 실제 약 10 bit 정도 더 커진다).

표에 나와있듯이, 제안된 기법은 기존 일회용 서명 기법보다 아주 작은 서명 크기를 가진다.

스트림 인증을 위한 기존 기법 (BiBa, Powerball, HORS)와 제안된 기법의 온라인 서명 연산량을 비교해보면, HORS의 서명 연산량보다는 다소 많지만, BiBa나 Powerball보다는 작다. 해쉬 함수 연산이 매우 빠른 점을 생각해보면, 제안된 기법은 서명 서버에 크게 부하를 주지 않음을 알 수 있다. 또한, [1], [3]에서 언급하듯이, 병렬 서버나 프로세서를 사용하면 쉽게 서명 속도를 향상시킬 수 있다.

III. 결론

본 논문에서는 스트림 데이터를 인증하는데 적합한 일회용 서명 기법을 제시하였다. 스트림 인증에 일회용 서명 기법을 사용할 경우, 가장 큰 문제점은 큰 서명 크기로 인해 네트워크 오버헤드가 크다는 점이다. 제시된 기법은 기존 기법에 비해 가장 작은 서명 크기를 가진다. 또한, 제시된 기법의 검증 연산량은 매우 작다. 온라인 서명 연산량은 HORS보다 다소 많지만, 기존 기법 중 스트림 인증에 적합한 기법인 BiBa, Powerball보다는 적다.

참고문헌

[1] M. Mitzenmacher, A. Perrig, Bounds and Improvements for BiBa Signature Schemes, Technical Report, 2002.

[2] A. Perrig, R. Canetti, D. Song, and D. Tygar. Efficient authentication and signature of multicast streams over lossy channels. In proceedings of the IEEE Symposium on Research in Security and Privacy, May 2000.

[3] A. Perrig, The BiBa One-Time Signature and Broadcast Authentication Protocol, ACM CCS'01, 2001.

[4] L. Reyzin, N. Reyzin, Better than BiBa: Short One-time Signatures with Fast Signing and Verifying, ACISP'02, 2002.

[5] P. Rohatgi. A compact and fast hybrid signature scheme for multicast packet. In 6th ACM Conference on Computer and Communications Security, p93-100, 1999.