다양한 타임 스탬핑 기법의 비교 분석

강재은, 이필중

포항공과대학교, 전자전기공학과

Comparison and Analysis of various Time-stamping schemes

Jae Eun Kang, Pil Joong Lee

Department of Electronic & Electrical Engineering, POSTECH

{florist, pjl}@postech.ac.kr

요 약

본 논문에서는 메시지 또는 메시지의 전자서명 값이 생성되거나 변경된 시점을 효과적으로 증명해 줄 수 있는 다양한 타임 스탬핑 기법들에 대해 설명한다. 이러한 타임 스탬핑 기법에서는 타임 스탬프를 발행하는 TSI(Time Stamp Issuer)가 중요한 역할을 하게 된다. 따라서 각 타임 스탬핑 기법에 대해, TSI가 타임 스탬프를 발행하는데 필요한 전자서명과 해쉬함수의 연산 수, TSI가 직접 저장하는 데이터와 그 데이터의 크기를 구하고, 각각을 비교하고 분석한다.

I. 서론

공개키 기반 구조에서는 전송하고자 하는 메시지를 자신의 비밀키로 전자서명을 해서 보냄으로서, 메시지에 대한 무결성을 제공하고 서명자가 자신의 서명 사실을 부인하는 것을 막아준다. 이러한 전자서명의 확인은 메시지를 서명한 비밀키와 관계있는 공개키가 포함되어 있는 인증서를 통해 가능하다. 따라서 전자서명이 올바른지를 검증하기 위해서는 인증서의 폐지 여부를 확인하는 것이 매우 중요하다. 그러나 전자서명을 검증하기 위한 공개키가 들어있는 인증서가 파기되거나 폐지된 경우, 검증자는 이 전자서명이 인증서가 유효한 동안에 생성된 것인지 아닌지를 확인하기가 어려워진다. 따라서 메시지 뿐만 아니라 메시지의 전자서명 값에 대해, 오랜 시간 동안 검증이 가능하도록 데이터가 생성되거나 변경되었던 시점에 대한 정보를 얻을 수 있도록 해야 한다. 이와 같이 시간 상에서, 어떤 특정 시점에 해당 데이터가 존재했음에 대한 증명을 제공하는 기술을 타임 스탬핑 기법이라고 한다[4, 5].

본 논문의 2장에서는 타임 스탬핑 기법들을 소개하기 위해, 타임 스탬핑 프로토콜에 관여하는 개체들과 기본적인 값들을 정의하고, 타임 스탬핑 기법들의 분류 기준을 간단하게 소개한다. 3장에서는 현재까지 제안된 다양한 타임 스탬핑 기법들의 간단히 설명하고, 4장에서는 각각의 기법에 대해서 TSI의 계산량과 데이터 저장량 구하고, 이를 비교 분석한다. 마지막으로 5장에서 본 논문의 결론을 맺는다.

II. 정의 및 분류 기준

1. 정의와 표기

본 절에서는 타임 스탬핑 프로토콜에 관여하는 개체들을 정의하고, 기본적인 데이터들의 표기를 살펴본다.

- TSI ($TimeStampIssuer$) : □타임 스탬프를 발행하고, 발행과 검증 과정에 관련된 모든 데이터를 저장하는 개체.
- TSR ($TimeStampRequester$) : □타임 스탬프 발행을 요청하는 개체.
- TS ($TimeStamp$) : □TSR의 요청을 받고 TSI가 발행해 주는 타임 스탬프.
- X: TS를 발행받고자 하는 메시지. (TSR는 주로 $H=h(X)$의 형태로, TS 발행을 요청한다.)
- H: 메시지 X의 해쉬값.
- T: TSR로부터 TS 발행을 요청 받은 특정 시점을 명시하는 시간값.
- S: TSI의 서명값.
- n: TS의 serial number.
- L: linking information.
- R_r: r번째 라운드(일정한 시간간격)의 축적

된 라운드 스탬프.

- ID : identifier.

2. 타임 스탬핑 기법의 분류

현재까지 제안된 타임 스탬핑 기법은 다음과 같이 크게 세 가지로 분류할 수 있다.

1) Simple 기법

타임 스탬프 TS를 발행받기 원하는 TSR이 메시지 또는 메시지의 해쉬값을 TSI에게 보내면, TSI는 받은 데이터에 타임 스탬프 요청을 받은 시점의 시간값 T를 포함한 후, 서명해서 TSR에게 보낸다. 이 기법은 매우 간단하지만, TSI가 반드시 TTP ($Trusted\ Third\ Party$)이어야 하는 단점이 있다.

2) Linking 기법

일정한 시간 간격, 즉 한 라운드 동안에 발생하는 TS 발행 요청에 대해, TRI는 TS 체인을 형성한다[1]. 단지 TRI는 TS 체인 형성을 위해 계산만 수행하고, 라운드 정보만을 주기적으로 발행하는 개체이기만 하면 될 뿐, 반드시 TTP일 필요는 없다. 하지만, 라운드 내에 발생하는 요청들의 TS 값들을 연결하기 위해 추가적인 연산들이 필요하므로, 시스템이 복잡하다.

3) Distributed 기법

비밀 값을 공유한 다중 TRI들이 협력해서 비교적 신뢰할 수 있는 TS 값을 발행하는 기법이므로 다중 TRI들이 반드시 TTP일 필요는 없지만, 많은 수의 TRI가 서명 값을 생성해야 하므로, 다른 시스템에 비해 연산량이 많고 복잡하다.

3. Temporal Authentication

메시지에 대한 temporal authentication은 크게 다음과 같은 두 가지 개념으로 나눌 수 있다[6].

1) Absolute

Absolute temporal authentication은 타임 스탬프 TS가 실제 시간 값을 포함하므로 가능하다. 따라서 임의의 두 메시지의 생성 순서를 알아보기 위해서는 실제로 메시지의 TS에 포함되어 있는 시간값 T를 비교하면 된다. 이 T는 Simple 기법에서 TTP에 의해 얻어지는 값과 distributed 기법에서 다중 TSI에 의해 얻어지는 값을 말한다.

2) Relative

Relative temporal authentication는 단지 어떤 메시지가 다른 메시지보다 이전에 생성되었다는 것만 확인할 수 있게 해준다. 즉 linking 기법에서처럼 한 라운드 내에서 발생한 TS 발행 요청들에 대해서, 특정 연산을 통해 TS 값들의 체인을 형성함으로서 상대적인 메시지가 생성되거나 변경된 시간 정보를 제공하는 것이다. 일반적으로 TRI가 발행하는 시간값 T를 신뢰할 수 없으므로, 이 방식을 선호한다.

III. 다양한 타임 스탬핑 기법 소개

1. The Electronic Notarization system

이 기법은 가장 간단한 타임 스탬핑 기법으로, TRI가 TTP의 역할을 하며, TRI가 발행해 주는 TS는 신뢰할 수 있는 시간값 T를 갖게 된다[9]. 따라서 나중에 메시지에 대한 검증은 이 T값만 확인하면 된다. TS 발행 과정은 다음과 같다.

(1) $TSR \rightarrow TSI$: X.□

(2) TSI : X□에 대한 T를 가지고, 서명값 $S = Sig_{TSI}(X, T)$를 생성한다.

(3) TSI : $H() = h(X, T, S)$□를 계산하고 저장한다.

(4) $TSI \rightarrow TSR$: $TS = [X, T, S]$.□

2. Benaloh와 de Mare에 의해 제안된 linking 타임 스탬핑 기법

Benaloh와 de Mare는 "One-way accumulator" 기술을 이용한 linking 타임 스탬핑 기법을 제안하였다[2]. 일방향 accumulator는 일방향 해쉬 함수 $f : X \times Y \rightarrow X$인데, $f(f(x, y_1), y_2) = f(f(x, y_2), y_1)$의 특징을 갖는다.

(1) (Prepartion) TSI는 p, q, x를 선택하고, $n = pq, x_0 = x^2 mod\, n$을 미리 계산해서 TSR들에게 알려준다.

(2) TSR_j : 해쉬 값 y_j를 생성한다.

(3) $TSR_j \rightarrow TSI$: y_j.

(4) TSI : $(m-1)$명의 다른 TSR들로부터 $[y_1, \square, y_{j-1}, y_{j+1}, \square, y_m]$를 받아서 다음을 계산한다. $Y = y_1 \times \square \times y_m$, $z = x_0^{\,Y} mod\, n$.

(6) TSI : $Y_j = Y/y_j$, $z_j = x_0^{\,Y'} mod\, n$을 계산한다.

(7) $TSI \rightarrow TSR_j$: $TS = [z_j, z, y_j]$.

이 기법은 어떤 특정한 시간 간격 동안에 TS 발행 요청을 한 다른 메시지들과 함께, 타임 스탬핑 서비스를 받았음을 증명한다. 또한 TS내의 z값은 각 라운드마다 구별되므로, 이는 시간값으로 간주된다.

3. Harber와 Stornetta에 의해 제안된 linking 타임 스탬핑 기법

Haber와 Stornetta는 이전에 발행한 TS_{n-1}내의 "linking information"을 다음 타임 스탬프 TS_n의 발행 요청이 있을 때 함께 묶어서 일방향 해쉬 연산을 함으로써, 발행하는 TS들 간에 체인을 형성하는 기법을 제안하였다[5]. TS 발행 프로토콜은 다음과 같다.

(1) $TSR \rightarrow TSI$: X.

(2) TSI : $H_n = h(n, ID_{TSR}, X)$.

(3) TSI : $L_n = h(H_n, L_{n-1})$.

(4) $TSI \rightarrow TSR$:

$TS = [\, n, ID_{TSR}, L_{n-1}, Sig_{TSI}(L_n)\,]$.

이 기법에서는 어떤 두 메시지의 TS를 비교하기 위해서, 두 메시지 사이에서 생성된 모든 "linking information"을 TSI로부터 전송 받아야 하므로, 최악의 경우 검증자는 TSI가 계산 했던 만큼의 해쉬함수 연산을 해야 하고, TSI는 모든 "linking information"를 모두 저장하고 있어야 하는 단점이 있다.

4. Buldas et al.에 의해 제안된 linking 타임 스탬핑 기법

Buldas et al.은 "threaded authentication tree" 구조를 갖는 타임 스탬핑 기법을 제안하였다[3]. 이 기법은 linking 기법을 이용하여 relative temporal authentication를 제공하고, TRI가 한 라운드 동안에 발생하는 TS 발행 요청에 대해 "Time Certifiate"를 발행하고, 각 라운드의 마지막에 생성된 "cumulative round stamp"를 PA ($Publicationauthority$)에게 보낸다. 이 기법은 어떤 두 메시지에 대해 TS 발행 순서를 검증하기 위해 TSI와의 통신 없이 TS 발행 과정에서 전송받은 $Cert(n)$ 정보를 통해서 빠르게 검증이 가능하다는 장점이 있다. TS를 발행하고 PA가 "cumulative round stamp"를 공개하는 프로토콜은 다음과 같다.

(1) $TSR \rightarrow TSI$: H_n.

(2) TSI : L_n과 $[\, n, L_n, Sig_{TSI}(n, L_n)\,]$.

(3) $TSI \rightarrow TSR$: $TS = [\, n, L_n, Sig_{TSI}(n, L_n)\,]$.

(4) TSI : $R_r, r, Sig_{TSI}(R_r, r)$.

(5) $TSI \rightarrow PA$: $R_r, r, Sig_{TSI}(R_r, r)$.

(6) PA : $R_r, r, Sig_{PA}(R_r, r, Sig_{TSI}(R_r, r))$.

(7) $PA \rightarrow TSI$:

$R_r, r, Sig_{PA}(R_r, r, Sig_{TSI}(R_r, r))$.

(8) PA : $R_r, r, Sig_{TSI}(R_r, r)$,

$Sig_{PA}(R_r, r, Sig_{TSI}(R_r, r))$을 공개한다.

(9) TSR : R_r을 확인한다.

(10) $TSR \rightarrow TRI$: n을 보내고, $Cert(n)$을 요청한다.

(11) TSI : $Cert(n), Sig_{TSI}(Cert(n))$.

(12) $TSI \rightarrow TSR$: $Cert(n), Sig_{TSI}(Cert(n))$.

5. The Time Signature Distributed System

Takura et al.은 TSI가 TS의 발행 요청을 받는 RS ($ReceptionServer$)와 부분 서명을 수행하는 여러개의 $MSSs$ ($MultipleSignServers$)로 이루어진 time signature distributed system을 제안하였다[8].

(1) $TSR \rightarrow RS$: $H, t, Sig_{TSR}(H, t)$.

(t: valid period data)

(2) RS : t와 $Sig_{TSR}(H, t)$를 검증한다.

(3) $RS \rightarrow MSSs$: H, T.

(4) $MSSs$: 각각은 현재 시간이 T에 가깝다면, $[H,T]$의 부분 서명값을 생성한다.

(5) $MSSs \rightarrow RS$: 생성한 부분 서명값들을 전송한다.

(6) RS : 전송 받은 부분 서명값의 개수가 일정한 개수 이상이면, 서명값 S와 TS를 생성한다.

(7) $RS \rightarrow TSR$: $TS = [\, H, T, S\,]$.

6. Digital Notary

Digital Notary는 *Surety.com*에서 제공되는 타임 스탬핑 기법이다[9]. 이 기법은 4절의 Buldas et al.이 제안한 linking 타임 스탬핑 기법과 매우 유사하게, "linking information"을 생성할 때, 이진 트리를 사용하고 있다. 하지만 차이점이 있다면, 각 라운드마다 생성되는 "cumulative round stamp"를 별도의 PA에게 보내지 않고, 이전 라운드에서 생성된 "cumulative round stamp"와 일방향 해쉬함수를 취해서 그 값을 저장하고 있다.

IV. 타임 스탬핑 기법의 비교 분석

다음 표 1은 앞 장에서 설명한 각 타임 스탬핑 기법에서 생성되는 타임 스탬프 TS에 포함되어 있는 값들과 각 기법에서 연산을 수행하는 개체들을 나타낸 것이다.

여기에서는 앞에서 소개한 다양한 타임 스탬프 기법들을 비교 분석하기 위해, 몇 개의 매개변수들을 정의한다.

- R : 한 라운드 내에서 발생하는 TS 발행 요청 횟수
- T_S : 각 개체가 서명하는데 걸리는 시간
- T_h : 해쉬함수 연산에 걸리는 시간
- k : 해쉬값의 크기

	TS	개체
1	$X, T, Sig_{TRI}(X, T)$ □	TSI, TSR
2	z_j, z, y_j	TSI, TSR
3	$n, ID_{TRS}, L_{n-1}, Sig_{TSI}(L_n)$	TSI, TSR
4	$n, H_n, Sig_{TSI}(n, L_n)$	TSI, TSR, PA
5	H, T, S	$TSI(RA + MSSs), TSR$
6	T_n, ID_n, H_n, L_n	TSI, TSR

표 1: TS내의 값들과 개체들

다음 표 2는 각각의 타임 스탬핑 기법에 대해 TS를 발행하기 위해 TSI가 수행하는 전자서명의 횟수와 필요한 시간을 구한 것이다. 각 TSI은 발생하는 각각의 R번의 TS 발행 요청에 차례로 응답하며, 이것을 한 라운드로 간주한다[1].

전자서명 횟수와 그에 필요한 시간이 다른 기법들에 비해 기법 4와 기법 5가 더 큰 값을 갖는다. 그 이유는 기법 4의 경우, 발행되는 각각의 $Cert(n)$에 대한 추가로 전자 서명을 하기 때문인데, 이는 후에 TSI와의 통신없이 효과적으로 TS를 검증하는 것이 가능하도록 해준다. 또한 기법 5의 경우, m개의 MSS가 각각의 TS 발행 요청에 대해 부분 서명값을 생성하므로, 전체적인 서명의 횟수는 많으나, 동시에 부분 서명값을 생성할 수 있으므로, 전체적인 전자서명에 소요되는 시간은 다른 타임 스탬핑 기법과 차이가 나지 않는다.

	전자서명 횟수	소요 시간
1	R	R□ T_S
2*	-	-
3	R	R□ T_S
4	$2R + 2$	$2(R+1)$□ T_S
5	R□ $(m + 1)$ $(\#ofMSSs = m)$	$2R$ □ T_S
6	0	0

표 2: TSI의 TS 생성시 전자서명

기법 2의 경우, "One-way accumulator" 기술로 연산을 하는 방법의 종류가 다양하므로, 전자서명과 해쉬함수 연산의 횟수와 그에 요구되는 시간이 달라지므로 고려하지 않기로 한다.

표 3은 TSI의 해쉬함수 연산수와 그에 필요한 시간을 보여준다. 기법 4와 6은 모두 이진 트리를 기준으로 계산한 값인데, 기법 4의 경우 차례대로 들어오는 TS 발행 요청 메시지에 대해서 차례로 linking 해쉬값을 생성하는 반면, 기법 6은 전체 요청 메시지가 다 들어온 후에 두개씩 묶어서 해쉬함수 연산을 하므로, 소요시간이 훨씬 짧다. 하지만 두 경우 모두, 라운드 내에서 같은 수의 TS 발행 요청 메시지가 발생하므로, 실제로 연산 시간에는 차이가 없다.

	해쉬함수연산	소요 시간
1	R	R□ T_h
2*	-	-
3	$2R$	$2R$□ T_h
4	$2R - 1$	$(2R-1)$□ T_h
5	0	0
6	R	$(logR + 1)$□ T_h

표 3: TSI의 TS 생성시 해쉬함수연산

다음 표 4은 각 기법에서 TSI가 저장하고 있는 데이터와 데이터의 양을 보여준다. 역시 마찬가지로 기법 4가 저장해야 하는 데이터의 양이 가장 많아 보이는데, 각각의 요청에 대해 $Cert(n)$를 발행한 후에는, 메시지의 TS를 검증하기 위해 TSI와의 통신없이 가능하므로, 더 이상 L_n값을 보관할 할 필요가 없다. 기법 2에서는 데이터의 크기를 알 수 없다.

	저장 데이터(갯수)	데이터의 크기
1	$H0(R)$	$R\square k$
2*	n, x_0, Y, z	-
3	$L_n(R)$	$R\square k$
4	$TSI: L_n(2R-1)$ $PA: R_r(1)$	$2R\square k$
5	0	0
6	$L_n(R-1), R_r(1)$	$R\square k$

표 4: TSI의 저장 데이터와 저장량

V. 결론

본 논문에서는 메시지와 메시지의 전자서명 값이 생성되거나 변경된 시점에 대한 정보를 효과적으로 얻을 수 있는 다양한 타임 스탬핑 기법에 대해 알아보고, 각각에 대해 TSI의 계산량, 그에 요구되는 시간과 데이터 저장량 구하고, 이를 비교 분석하였다.

참고문헌

[1] J. Benaloh, and M. de Mare, " Efficient Broadcast Time-Stamping," *Technical Report 1*, Clarkson University Department of Mathematics and Computer Science, Aug. 1991.

[2] J. Benaloh, and M. de Mare, "One-Way Accumulators: A Decentralized Alternative to Digital Signatures," *Proceedings of EUROCRYPT93*, LNCS 765, pp. 247-285, 1994.

[3] A. Buldas, H. Lipmaa and B.Schoenmakers, "Optimally Efficient Accountable Time-Stamping," *Proceedings of PKC2000*, LNCS 1751, pp. 293-305, Aug. 1999.

[4] S. Harber, B. Kaliski and W. S. Stornetta, "How Do Digital Time-Stamps Support Digital Signatures?," *Cryptobyte*, vol. 1(3), pp. 14-15, 1995.

[5] S. Harber, and W. S. Stornetta, "How to Time-Stamps a Digital Document," *Journal of Cryptology*, vol. 3(2), pp. 99-111, 1991.

[6] M. Just, "Some Timestamping Protocol Failures," *Proceedings of the Symposium on Network and Distributed Security(NDSS 1998)*, Mar. 1998.

[7] M. Roos, "Integrating Time-Stamping and Notarization," *Master's Thesis*, Tartu University, May. 1999.

[8] A. Takura, S. Ono and S. Naito, "Secure and Trusted Time Stamping Authority," *Proceedings of IWS99*, pp. 123-128, 1999.

[9] M. Une, "The Security Evaluation of Time Stamping Schemes: The Present Situation and Studies," *IMES Discussion Paper Series*, Institute for Monetary and Economic Studies, 2001.