TSP(Time Stamp Protocol) SDK 구현

원형석[0], 노종혁, 최대선, 진승헌
한국전자통신연구원
(moho[0], jhroh, sunchoi, jinsh)@etri.re.kr

TSP SDK Implementation

Hyung Suk Won[0], Jonghyuk Roh, Daesun Choi, Seunghun Jin
Electronics and Telecommunications Research Institute

요 약

시점확인 서비스(Time-Stamping Service)를 위한 프로토콜인 Time-Stamp protocol(TSP)를 구현하는데 필요한 기능들을 SDK로 구현하였다. 어떤 데이터가 특정 시점 이전에 존재했음을 증명하기 위해 Time-Stamp token을 발행하는 시점확인 서비스는 전자공증, 전자상거래, DVCS 등 여러 보안과 관련된 응용서비스에서 필요한 모듈이다. 본 논문에서는 시점확인 서비스를 위한 time-stamp protocol을 설명하고, TSP SDK 구현 내용, SDK를 시험하기 위한 테스트베드 및 SDK 구현시 참고사항을 서술한다.

1. 서 론

Time-Stamp protocol[1]은 TSA(Time-Stamp Authority)와 요청자 사이에 시점확인을 위한 토큰을 주고 받는 Time-stamping service를 위한 규격으로써, Time-stamping service란 어떤 데이터가 특정 시간 이전에 존재했음을 뒷받침해 주기 위한 서비스이다. 일반적으로 시점확인 서비스로 번역되어 사용된다. TSP는 [RFC-3161]에 근거하고 있다.

2. Time-Stamp protocol (TSP)

2.1 TSA의 요구사항

TSA는 다음과 같은 요구사항을 만족해야 한다.

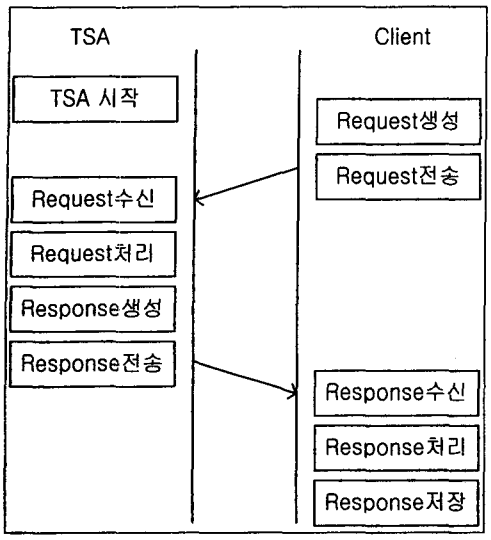
(1) 타임소스가 믿을만 해야 한다.
(2) 타임스탬프 토큰에 신뢰있는 시간값을 포함해야 한다.
(3) 각각의 타임스탬프 토큰마다 고유한 정수값을 포함해야 한다.
(4) 요청자로부터 요청이 있을 때 타임스탬프 토큰을 만들어야 한다.
(5) 타임스탬프 토큰 안에 토큰이 만들어진 보안정책을 고유하게 식별할 수 있는 지정자가 포함되어야 한다.
(6) OID에 의해 고유하게 식별되는 일방향 충돌회피 해쉬 함수(one-way collision resistant hash-function)와 관련된 *imprint* 된 데이터의 해쉬된 것만을 타임스탬프 해야 한다.
(7) 일방향 충돌회피 해쉬 함수의 OID를 검사하고 해쉬된 값의 길이가 해쉬 알고리즘과 일치하는 가를 검사한다.
(8) 길이 이외에 타임스탬핑 하는 imprint를 검사해서는 안된다.
(9) 타임스탬프 토큰에는 요구하는 측의 어떠한 식별자도 들어가지 않는다.
(10) 서명을 위해 배타적으로 생성한 키를 사용하여 각각의 타임스탬프 토큰에 서명을 하고 해당하는 인증서에 명시된 키의 속성을 명시한다.
(11) 만약 요구하는 측에서 확장필드를 이용해 추가적인 정보를 요구하면, 타임스탬프 토큰 안에 추가적인 정보를 포함한다. 만약 이것이 가능하기 않으면, TSA는 에러메시지를 보내야한다.

2.2 Transaction

TSA와 client간의 transaction은 아래의 [그림1]과 같이 진행된다. 이들 transaction들 중, request 생성, request 처리 response 생성, response 처리 부분이 구현된 TSP SDK 함수들로 처리가 가능하다.

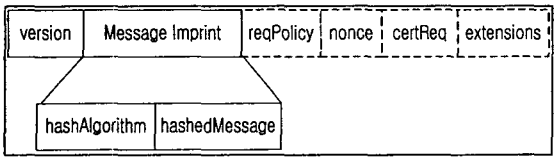
2.3 요청 메시지 구조

요청 메시지 구조는 [그림2]와 같다. 그림의 점선 부분으로 표시된 필드들은 optional을 의미한다. 각각의 필드들에 대한 간략한 설명은 아래와 같다.



[그림 1] TSP transaction

version	time-stamp request의 버전 (현재 v1)
messageImprint	사용자가 시점확인을 요구하는 데이터의 해쉬값으로써, 사용되는 해쉬 알고리즘은 일방향 충돌회피 해쉬 함수여야 하고 TSA는 사용된 해쉬 알고리즘과 해쉬 길이를 검사한다.
reqPolicy	TSA policy를 요구하여 사용자의 요구에 맞는 지 검사하게 된다. 명시되지 않을 경우, TSA의 모든 정책을 수용한다는 의미가 된다.
nonce	요청자가 응답의 적절성을 증명하기 위해 사용된다. 요청에 포함된 nonce 값이 반드시 응답에 포함되어야 한다.
certReq	만약 true로 세팅되면, TSA의 응답시 TSA의 public key certificate가 SignedData structure의 certificate 필드에 signingCertificate attribute에 포함되어야 한다. 필드가 없거나 false이면, 응답시 certificate field가 없어야 된다
extensions	추가 요구 정보



[그림 2] 요청 메시지 구조

2.4 응답 메시지 구조

응답 메시지 전체 구조는 [그림3]과 같으며, 응답 메시지를 구성하는 주요 구조체의 필드들에 대한 간략한 설명은 아래와 같다.

Response 구조

status	만약 0 또는 1이면, TimeStampToken이 반드시 있어야 하고, 2~5이면 오류가 있으므로, TimeStampToken을 발행하지 않는다. 0~5사이의 수이어야만 한다. 즉, 5를 넘으면 안된다. 만약 5를 넘으면 요청자가 에러를 발생시켜야 한다.
timeStampToken	타임스탬프 토큰

SignedData 구조

version	CMS version, TST의 경우는 version 3
digestAlgorithms	message digest algorithm identifiers
encapContentInfo	content type identifier와 content 로 구성되어 있으며, TSP에서는 TSTInfo가 content에 들어가게 된다.
certificates	TSA의 certificate
crls	collection of certificate revocation lists(CRLs)
signerInfos	signer의 정보를 담는 곳

TSTInfo 구조

version	Time-stamp token의 버전
policy	Response를 만드는 TSA의 policy
messageImprint	요청받은 messageImprint
serialNumber	TSA가 TimeStampToken에 부여하는 고유한 정수값
genTime	타임스탬프 토큰이 만들어진 시간, UTC time
accuracy	시간 편차
Ordering	필드가 없거나 false로 세팅되어 있으면, 동일한 또는 다른 TSA가 발행한 time-stamp token들은 두 토큰의 genTime 차이가 각 토큰들의 accuracy 들을 합친 것 보다 클 때에만 순서나열이 가능하다. 필드가 존재하고 true로 세팅되어 있으면, 같은 TSA에서 만든 모든 time-stamp token은 genTime의 accuracy에 상관없이 순서대로 나열이 가능하다.
nonce	만약 TimeStampRequest에 nonce가 있으면 반드시 같은 값이 응답에 포함되어야 한다.
tsa	TSA의 이름을 구분하는데 힌트를 준다. 만약 필드가 존재하면, 토큰을 확인하기 위해 사용되는 certificate 안에 포함된 subject name들 중의

	하나에 해당한다.
extension	추가정보를 담기 위한 확장필드

3. TSP SDK

TSP SDK는 Time-Stamp token을 요구하는 클라이언트 측과 time-stamp token을 생성하여 응답하는 서버 측, 그리고 SDK 함수들을 테스트 하기 위한 테스트베드 함수들로 구성되어 있다.

3.1 Client 측 SDK

request 생성	*TS_RequestInit()*	request 초기화
	TS_RequestSetNonce()	Nonce setting
	TS_RequestMessageImprint()	Message imprint
	TS_RequestSetPolicyID()	Policy setting
	TS_RequestSetCertReq()	Certificate 요구 유무
	TS_RequestSetExtension()	Extension setting
response 처리	*TS_ResponseStatusCheck( )*	response 상태검사
	TS_ResponseDecodeTimeStampToken()	타임스탬프토큰 디코딩
	TS_ResponseVerify()	TST info 검사

3.2 Server 측 SDK

request처리	*TS_RequestCheck()*	request 검사
response 생성	*TS_ResponseInit()*	response 초기화
	TS_ResponseTSTInfoGenerate()	TSTInfo 생성
	TS_ResponseSignedDataGenerate()	SignedData 생성
	TS_ResponseGenerate()	response 생성

3.3 테스트베드

작성된 TSP SDK의 함수들을 사용하여 요청 및 응답 메시지를 생성하고 encoding/decoding을 통해 생성된 토큰을 확인을 해 볼 수 있는 샘플코드로 구성되어 있다.

3.4 구현시 고려사항

TSP 구현시 고려해야 할 사항들은 다음과 같다. time-stamp token의 경우, content type이 id-ct-TSTInfo가 되고, TSTInfo가 [CMS][2]의 SignedData의 encapsulated content부분에 들어가야 된다. 그렇게 되면 [CMS]의 규정에 따라, SignedData의 version이 3이 되어야 한다. 또한, 원래는 SignedData를 구성하는 signerInfo의 signedAttributes 필드가 optioanl이지만, 타임스탬프 토큰이 들어가게 되면, 이 필드가 반드시 들어가야 된다.

3.5 개발 환경 및 개발 툴

- 프로그래밍 언어 : C
- OSS ASN Compiler, OSS library
- 개발/적용환경 : Windows, Unix

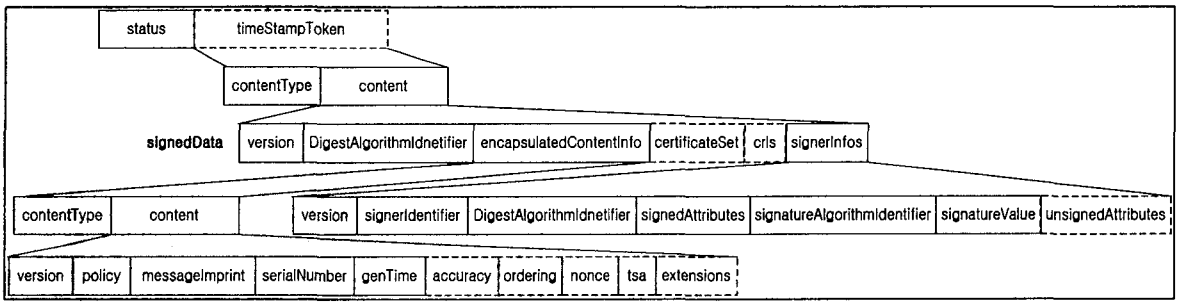
4. 결론

본 논문에서는 시점확인 서비스를 위한 TSP을 소개하고 TSP SDK 구현내용과 고려사항들을 기술하였다. TSP SDK를 이용하여 TSA를 구성하여 다른 여러 보안응용 서비스와 연계하여 사용될 수 있다. 하지만 실제로 TSP를 이용한 TSA가 기존의 서비스들과 연동되어 효율적으로 운영되기 위해서는 해결되어야 할 요소들이 있다. 먼저 TSP에서 가장 중요한 타임소스를 어떻게 할 지에 대한 가이드 라인이 필요하다. 또한 외부 서비스와 연동을 고려한 타임스탬프 토큰 생성 방식을 고려되어야 한다. 이런 요소들이 해결된다면 TSP SDK로 구성할 TSA가 보다 쉽게 여러 서비스와 연동되어 많이 이용될 수 있을 것이다.

[참고문헌]

[1] C. Adams, P. Cain, D. Pinkas, and R. Zuccherato, "Time Stamp Protocol", RFC 3161, August 2001.

[2] R. Housley, "Cryptographic Message Syntax", RFC-2630, June 1999.



[그림 3] 응답 메시지 구조