Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.27, NO.4, Aug. 2017 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2017.27.4.743

프락시 기반 애플리케이션 전자서명 검증 시스템*

권 상 완,[1†‡] 김 동 욱,[1] 이 경 우[2]
[1]삼성전자 소프트웨어센터, [2]연세대학교 컴퓨터과학과

Proxy Based Application Digital Signature Validation System*

Sangwan Kwon,[1†‡] Donguk Kim,[1] Kyoungwoo Lee[2]
[1]Software R&D Center, Samsung Electronics,
[2]Department of Computer Science, Yonsei University

요 약

스마트폰의 대중화와 더불어 웨어러블 디바이스, 가전제품, 스마트 TV에도 운영체제가 보급되고 있다. 사용자는 운영체제가 설치된 디바이스 상에서 다양한 애플리케이션을 사용할 수 있게 되었지만, 반면에 애플리케이션을 통한 해커의 위협도 증가하고 있다. 이에 따라, 운영체제 내 애플리케이션의 위변조를 탐지하기 위한 기술은 더욱 중요시되고 있으며, 애플리케이션의 위변조를 탐지하기 위한 기술 중 하나로, 전자서명 기술이 널리 사용돼 왔다. W3C의 규격에 따라 애플리케이션은 최소 2회 이상의 서명이 필요하며, 설치 시 각 구성요소에 대한 모든 서명 파일이 검증되어야 한다. 따라서 애플리케이션 전자서명 검증 시스템의 성능은 운영체제의 인스톨러 성능에 큰 비중을 차지한다. W3C 규격을 준수한 운영체제의 애플리케이션 전자서명 검증 시스템은 구성요소에 대한 무결성 검증 과정이 중복되기 때문에 성능 저하가 발생한다. 본 연구는 이러한 문제를 해결하기 위해 무결성 검증 과정 내 프락시 시스템을 도입하여 성능을 향상하는 방법을 제안한다. 제안된 연구는 기존 검증 시스템 대비 효율적으로 성능이 개선됨을 보여준다.

ABSTRACT

As smart phones are becoming popular, an operating system is being used at wearable devices, home appliances and smart TVs. A user is able to use various applications on devices with operating system, but there is an increased threat of hacker. Thus, the technology for detecting the forgery of applications is becoming more important on operating system. To detect the forgery of the application, a digital signature technology is used on the filed of application digital signature. According to W3C recommendation, the signing process of application digital signature must be performed at least twice, and the applications which are signed by the application digital signature have to be validated for all signature files when the application is installed in the operating system. Hence, the performance of the application digital signature validation system is closely related to the installer performance on the operating system. Existing validation system has performance degradation due to redundancy of integrity verification among application components. This research was conducted to improve the performance of the application digital signature validation system. The proposal of validation system which is applied proxy system shows a performance improvement compared to the existing verification system.

Keywords: application digital signature, application integrity, signature validation system

I. 서 론

전자서명이란 전자 문서의 무결성, 서명자 인증, 부인 방지의 기능을 보장받을 수 있는 기술로 전자상거래, 금융 시스템에 핵심 기술로 사용되고 있으며, 운영체제 내에서 애플리케이션의 무결성을 보장

Received(06. 27. 2017), Modified(07. 17. 2017), Accepted(07. 18. 2017)
* 본 연구는 삼성전자 소프트웨어센터의 지원 및 관리로 수행하였습니다.
† 주저자, sangwan.kwon@samsung.com
‡ 교신저자, sangwan.kwon@samsung.com(Corresponding author)

하고 서명자를 인증하려는 방법으로 널리 이용된다.

애플리케이션 전자서명 중 서명은 애플리케이션을 구성하는 각 파일들의 해시값들을 모아서 서명하고 이를 포함하여 해시값들과 함께 하나의 파일을 생성한다. 이를 서명 파일이라 하며, 애플리케이션과 함께 배포된다. W3C의 규격에 따라 서명은 애플리케이션의 개발자의 서명과 배포자의 수만큼 서명이 필요하다[1]. 애플리케이션 설치 시 운영체제 인스톨러는 애플리케이션 내 모든 서명 파일에 대하여 검증을 실시해야 한다. 이와 같은 이유로 애플리케이션의 무결성을 보장하기 위한 전자서명 검증 시스템의 성능은 인스톨러의 성능에 큰 비중을 차지한다.

기존 전자서명의 검증 성능을 강화하는 연구들은 서명 시 신뢰할 수 있는 프락시에 위임하여[2][3], 여러 구성요소들에 대해 동일한 개인키를 서명을 하고, 단일 공개키로 일괄적으로 검증하는 연구가 주로 이루어졌다[4]. 하지만 애플리케이션 내 전자서명 서명 과정의 경우, 여러 개의 애플리케이션 스토어가 존재할 수 있기 때문에 서명자들 즉, 개발자와 배포자 간에 동일한 개인키를 가질 수 없는 구조를 가지고 있다. 따라서 기존 전자서명에 관한 연구를 애플리케이션 내 전자서명에 적용할 수 없었다.

본 논문에서는 W3C 규격에 따른 애플리케이션 전자서명 검증 시스템의 성능을 향상시키는 방법을 제안한다. 현재 애플리케이션 전자서명 검증 시스템은 레퍼런스(구성요소)들의 해시값을 검증할 때에 서명 검증 횟수에 따라 이를 중복적으로 검사하여 성능 저하가 발생하였다. 제안한 방법에서는 검증 시스템 내 프락시 시스템을 도입하여 W3C 규격을 유지하면서 불필요한 무결성 검사를 제거한다. 제안한 방법을 평가하기 위하여 타이젠 운영체제를 이용하여 시스템을 구현하고, 이에 대한 성능을 측정하였다. 타이젠 운영체제는 W3C 규격을 준수하고 있는 유일한 모바일 플랫폼이다[5]. W3C의 규격을 준수하고 있는 타이젠 운영체제의 경우 서명 검증 과정은 애플리케이션 설치 중 이루어지는 9단계의 과정 중 하나이지만 소요시간은 큰 비중을 차지한다[6]. 기존 타이젠 운영체제의 애플리케이션 전자서명 검증 시스템과 본 연구에서 제안한 애플리케이션 전자서명 검증 시스템을 비교 실험함으로써 서명 검증의 성능 개선을 확인하였다.

II. 배경지식

2.1 전자서명

전자서명이란 서명자를 확인하고 서명자가 당해 전자 문서에 서명하였음을 인증하기 위하여 당해 전자 문서에 첨부되거나 논리적으로 결합한 전자적 형태의 정보를 말한다. 공개키 암호기술에 기반을 두는 방식으로 서명자는 한 쌍의 공개키와 개인키를 소유하며 자신의 개인키와 암호학 알고리즘을 통하여 메시지에 서명한다. 검증자는 서명자의 공개키와 암호학 알고리즘으로 서명의 유효성을 검증한다[7].

전자서명은 서명할 전자 문서에 대해 무결성, 서명자 인증, 부인 방지와 같은 보안 요구 사항을 만족시킨다.

2.2 애플리케이션 전자서명

운영체제 내 애플리케이션의 무결성을 보장하기 위해 기존 전자서명을 응용한 기술을 애플리케이션 전자서명이라 한다. 전자서명 관점에서 애플리케이션은 전자 문서를 뜻하며 개발자 및 배포자는 서명자를 뜻한다. 애플리케이션 전자서명은 W3C의 규격에 명시되어 있다[8].

2.2.1 서명

애플리케이션 전자서명에서 서명 과정은 레퍼런스 생성 단계와 서명 생성 단계로 이루어진다. 레퍼런스 생성 단계는 무결성을 보장하기 위해 애플리케이션의 구성 파일들, 즉 모든 레퍼런스들의 해시값들을 구하는 단계이다. 서명 생성 단계는 서명자를 인증하기 위해 공개키 기반 구조를 이용하여 서명값을 생성하는 단계이다. 레퍼런스 생성 단계와 서명 생성 단계의 결과물인 해시값들과 서명값은 메타 정보와 함께 서명 파일에 기록되며 서명 파일은 당해 애플리케이션에 포함된다[9].

서명자의 주체에 따라 서명은 두 가지로 구분된다. 애플리케이션 개발자의 개인키와 인증서로 서명하는 것을 개발자 서명이라 하고, 배포자의 개인키와 인증서로 서명하는 것을 배포자 서명이라 한다. 배포자 서명 전 개발자 서명이 진행되어야 하며 배포자 서명은 개발자 서명 파일을 포함하여야 한다[1]. 서명을 통해 개발자는 자신의 애플리케이션이 변조되지

않음을 확인하며 배포자는 해당 애플리케이션이 자신의 스토어를 통해 배포되었음을 인증한다.

2.2.2 검증

애플리케이션 전자서명의 검증 과정은 무결성 검증과 서명 검증으로 이루어진다. 무결성 검증 단계에선 서명 파일에 명시된 메타 정보를 이용하여 애플리케이션의 구성 정보들에 대해 해시값을 재생성한다. 재생성된 해시값과 서명 파일에 명시되어 있는 해시값을 비교함으로써 무결성을 보장한다. 서명 검증은 서명 파일에 명시된 서명자의 공개키를 메타 정보를 이용하여 서명 파일에 명시되어 있는 서명값을 확인함으로써 서명자 인증과 부인방지를 보장한다. 검증 과정은 애플리케이션에 포함되어 있는 모든 서명 파일에 대해 진행되어야 한다.

2.2.3 전자서명과의 차이점

기존 전자서명은 주로 단일 문서의 무결성을 보장하지만 애플리케이션 전자서명은 애플리케이션 내 모든 구성요소에 대해 무결성을 보장하여야 한다. 이에 따라 기존에 해시 모듈이 한번 수행되었던 것에 비해 애플리케이션 내 구성 파일 개수만큼 해시 모듈이 수행된다. 또한 서명, 검증의 단위가 단일 서명 파일에 이루어졌던 거에 비해 애플리케이션 내 모든 서명 파일에 대해 검증이 이루어져야 한다. W3C의 규격에 따라 애플리케이션은 1개의 개발자 서명 파일과 최소 1개의 배포자 서명 파일 가져야 하며, 애플리케이션 스토어, 즉 배포자의 수에 따라 서명 파일은 N개까지 증가할 수 있다. 애플리케이션 설치 시 모든 서명 파일을 검증하는 과정에서 애플리케이션의 레퍼런스를 중복적으로 검사하는 부하가 존재하였다.

기존의 단일 공개키로 일괄적으로 서명을 검증하는 전자서명 시스템의 성능 개선 연구와 달리, W3C 규격을 따르는 시스템에서 성능을 개선하는 연구는 국내외 학계에서 발표된 바 없기 때문에 본 연구가 최초라고 볼 수 있다.

2.3 애플리케이션 전자서명 서명 관계도

W3C 규격에 명시되어 있는 애플리케이션 전자서명(ADS, Application Digital Signature)의 서명 관계도는 Fig. 1.과 같다[1]. 실선의 화살표는

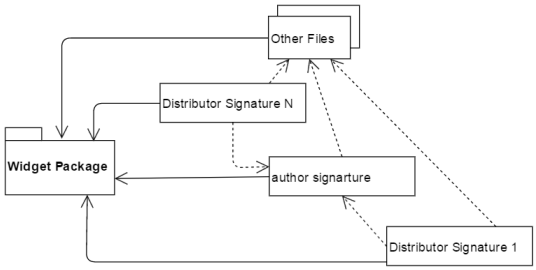

Fig. 1. ADS signing model

애플리케이션에 포함되는 레퍼런스의 관계를 나타내며 점선의 화살표는 서명의 대상이 되는 관계를 나타낸다.

개발자 서명은 애플리케이션 내 모든 파일을 대상으로 전자서명을 실시하며, 배포자 서명은 추가로 개발자 서명 파일을 포함하여 전자서명을 실시한다[1]. 배포자 서명 시에는 다른 배포자들의 서명 파일은 제외함으로써 애플리케이션이 다른 배포자들에 의해서도 재배포 될 수 있게 한다.

III. 타이젠 운영체제 애플리케이션 전자서명 검증 시스템

W3C의 규격을 준수하고 있는 타이젠 3.0 운영체제의 전자서명 검증 시스템을 분석하여 기존 시스템의 문제점을 확인한다.

3.1 시스템 구성도

애플리케이션은 운영체제 인스톨러에 의하여 설치되며, 설치 과정의 일환으로 애플리케이션 전자서명 검증이 진행된다.

타이젠 애플리케이션 전자서명(TADS, Tizen Application Digital Signature)의 검증 시스템의 시스템 구성도는 Fig. 2.와 같다. 실선 화살표는 서명 파일의 입력 흐름을 나타내며 점선 화살표는 서명 파일에 대한 무결성 검증 흐름을 나타낸다. 무결성 검증 흐름은 서명 파일의 개수만큼 존재한다.

TADS의 검증 시스템(cert-svc)은 운영체제 인스톨러에 의해 호출된다. 검증 시스템은 서명 파일 탐색, 파서(parser), 애플리케이션 권한 검증의 기능을 제공한다. 서명 파일의 검증 기능은 검증 모듈(xmlsec1)에서 수행한다.

기존 검증 시스템은 애플리케이션 내 서명 파일

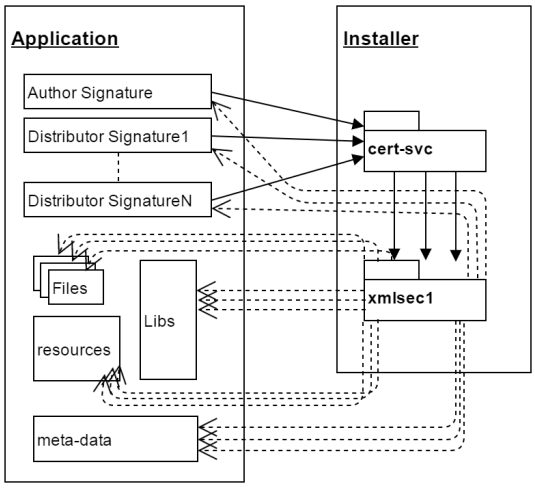

Fig. 2. TADS validation system architecture

검증 시, 검증 모듈을 통해 파일에 명시되어 있는 모든 레퍼런스들의 해시값들을 비교하여 무결성을 검증한다[8]. 이때, 각 레퍼런스 파일에 대한 입출력 비용과 해시 연산 비용이 발생한다.

3.2 검증 플로차트

타이젠 운영체제의 애플리케이션 전자서명 검증 시스템의 플로차트는 아래 Fig. 3.과 같다. 애플리케이션 내 모든 서명 파일에 대하여 검증하므로 W3C 규격에 따라 최소 2회 검증이 진행된다[1].

먼저 검증 시스템은 애플리케이션 내 존재하는 서명 파일들을 탐색 후 파서를 통해 서명 파일 내 기록되어있는 서명자의 공개키, 레퍼런스 해시값, 서명값, 메타 정보를 파싱 한다.

서명 검증은 서명 파일 파싱 후 Fig. 3.의 왼쪽 분기에 따라 진행된다. 파싱 된 공개키, 서명 파일, 공개키 알고리즘으로 기존의 서명을 검증하고, 이로써 적합한 서명자에 의해 서명이 되었음을 인증하며 부인방지를 보장한다.

무결성 검증은 Fig. 3의 오른쪽 분기에 따라 진행 된다. 전자서명 대상 파일들의 무결성을 검증하기 위하여 레퍼런스들의 해시값을 구하여 서명 파일에 적혀 있는 해시값과 비교한다. 애플리케이션은 리소스, 소스 코드, 라이브러리, 메타 데이터 등 다양한 파일로 구성되어 있으므로 오른쪽 분기의 로직은 이 모든 레퍼런스에 대해 반복적으로 진행된다.

레퍼런스의 해시값을 구하는 부분은 해시 함수의

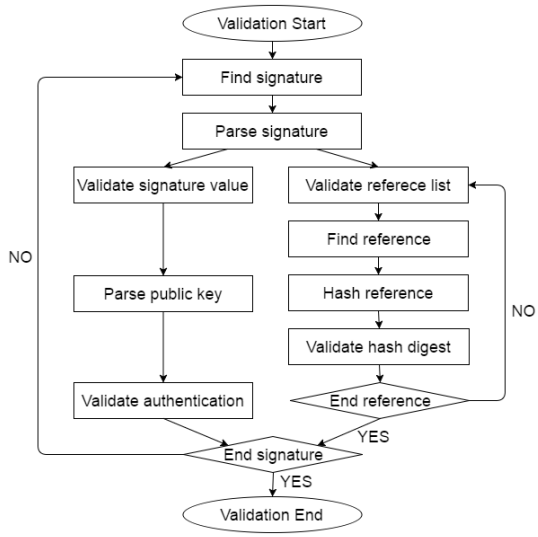

Fig. 3. TADS validation flowchart

비용과 해당 파일을 파일 시스템에서 메모리로 읽어드리는 입출력 비용으로 구성되며 파일 크기가 커질수록 더욱 크게 영향을 받는다.

3.3 검증 비용

애플리케이션 내 서명 파일의 개수를 n, 레퍼런스의 개수를 r이라고 할 때 이에 따른 검증 비용의 식은 수식(1)과 같다.

$$Cost(n) = \sum_{i=1}^{n}(SC(i) + \sum_{j=1}^{r} IC(i,j)) \quad (1)$$

SC : Signature validation Cost
IC : Integrity validation Cost

전체 검증 비용은 서명 검증 비용과 무결성 검증 비용의 합이다. 서명 검증 비용은 서명값을 구하기 위해 공개키 알고리즘을 통한 비용이 발생하며 무결성 검증 비용은 해시값을 구하기 위해 해시 함수의 비용과 파일 입출력 비용이 발생한다.

서명 검증 비용은 n번 발생하며 무결성 검증 비용은 n번의 비용마다 각 r번의 비용이 발생한다.

3.4 기존 검증 시스템의 문제점

애플리케이션 전자서명 검증 시스템은 W3C의 규격에 따라 애플리케이션 내 모든 서명 파일에 대하여

검증을 진행하여야 하고, 각 서명 파일의 검증은 서명 파일에 포함된 모든 레퍼런스에 대하여 무결성 검증을 진행하여야 한다[1]. 이를 위해 TADS의 검증 시스템은 각 서명 파일 단위로 독립적으로 검증을 실시하는데, 이는 각 서명 파일 검증 도중에 파일이 변조 되어도 마지막 서명 파일 검증 시 이를 탐지할 수 있게 한다. 또한 이전 서명 파일 검증이 실패하는 경우, 나머지 서명 파일의 검증을 실시 않아도 되는 장점이 있다.

하지만, 이러한 장점에도 불구하고 기존 검증 시스템은 중복적인 무결성 검증 비용이 발생하게 되므로 이에 대한 성능개선이 필요하다.

IV. 프락시 기반 애플리케이션 전자서명 검증 시스템

4.1 시스템 구성도

본 연구에서 제안한 프락시 기반 애플리케이션 전자서명(PADS, Proxy based Application Digital Signature)의 검증 시스템 구성도는 Fig. 4.와 같다. 실선 화살표는 서명 파일의 입력 흐름을 나타내며 점선 화살표는 서명 파일에 대한 무결성 검증 흐름을 나타낸다. 프락시를 도입함으로 TADS 시스템 구성도(Fig. 3.)의 무결성 검증 흐름이 서명 파일 개수인 것에서 한 번으로 줄어든 것을 확인할 수 있다.

프락시 시스템은 실제 검증 시스템(cert-svc)과 검증 모듈(xmlsec1) 사이에 위치한다. 프락시 시스템은 검증 시스템으로부터 서명 파일을 전달받고 검증 모듈과 상호 통신한다. 이때, 이미 검증된 파일에 대한 해시값을 개발자 서명검증과 배포자 서명검증에 재활용하기 때문에 무결성 검증 내 중복 비용을 제거한다. 또한, 타임스탬프를 활용하여 검증 기간 내 무결성을 보증함으로써, 기존과 동일한 보안 수준을 제공한다.

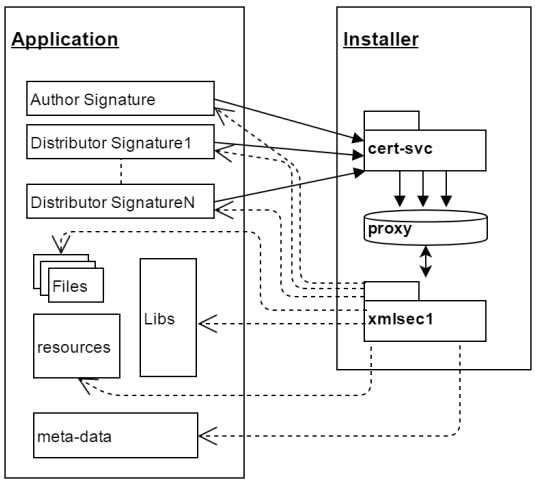

Fig. 4. PADS validation system architecture

4.2 검증 플로차트

제안한 검증 시스템의 검증 플로차트는 Fig. 5.와 같다. 전처리 부분인 서명 파일 탐색, 파싱 과정과 서명 검증 부분은 기존과 같다. 프락시가 적용된 부분은 Fig. 5.의 플로차트 오른쪽 분기의 무결성 검증 부분이다.

애플리케이션 내 서명 파일 중 검증 시스템에 첫 번째 입력으로 들어오는 서명 파일에 대한 무결성 검증 시 프락시의 캐시에 저장된 값이 없으므로 기존의 방법과 같은 방식으로 1회 진행된다. 이후 진행되는 서명 파일의 무결성 검증 시 프락시가 적용되어 중복 비용을 제거한다.

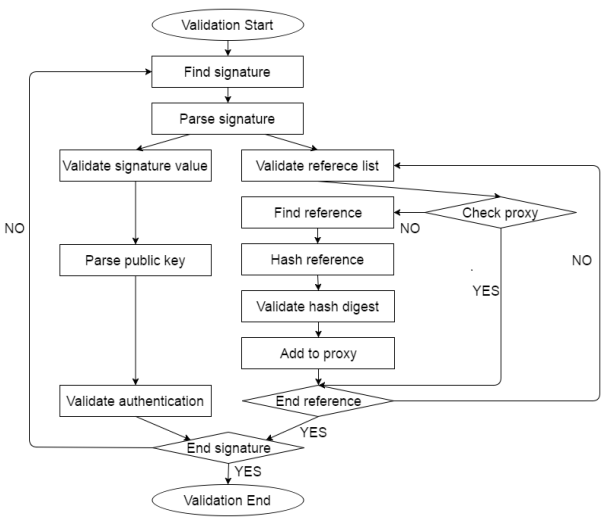

Fig. 5. PADS validation flowchart

4.3 프락시 플로차트

프락시 플로차트는 Fig. 6.와 같다. 프락시는 인코더, 캐시로 구성되어 있다.

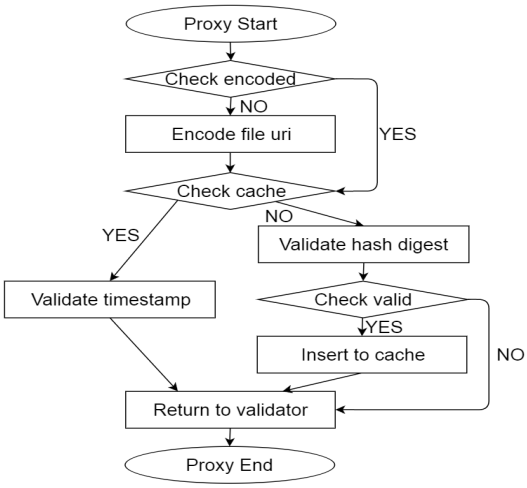

Fig. 6. PADS proxy system flowchart

인코더는 입력으로 들어온 파일의 위치를 나타내는 URI 값을 정규화 한다. 정규화는 파일 시스템의 종류에 상관없이 URI 값을 같은 형태로 나타내기 위함이며, 정규화 알고리즘은 W3C 규격에 따라 C1N14로 설정한다[9].

캐시는 무결성이 검증된 파일에 대한 정보를 저장한다. 저장하는 파일의 정보는 URI 값, 해시값, 타임스탬프이다. 타임스탬프는 PADS의 검증 과정 중에 일어날 수 있는 파일의 위변조를 검증하는 데에 사용된다. 캐시에 최초 들어갈 때 파일에 대한 타임스탬프를 저장하고, 이후 캐시에 접근 할 때의 파일 스탬프와 비교하여 검증 과정 중 위변조가 있었는지 판단한다.

이처럼 캐시에 저장된 URI 값은 파일스탬프 검증을 거쳐 무결성 검증을 완료한다. 캐시에 존재하지 않은 URI 값은 기존과 같은 방식으로 무결성 검증 로직에 따라 진행한다.

4.4 검증 비용

애플리케이션 내 서명 파일의 개수를 n, 레퍼런스의 개수를 r이라고 할 때 이에 따른 검증 비용의 식은 수식(2)와 같다.

$$Cost(n) = \sum_{i=1}^{n} SC(i) + IC(1) + \sum_{i=1}^{n-1}(\sum_{j=1}^{r} PC(i,j)) \quad (2)$$

SC : Signature validation Cost
IC : Integrity validation Cost
PC : Proxy system Cost

전체 검증 비용은 서명 검증 비용과 무결성 검증 비용 그리고 추가로 발생한 프락시의 비용의 합이다. 기존 시스템의 검증 비용(1)과 비교하였을 때 무결성 검증비용은 n번에서 1번으로 감소하였고, 추가로 n-1번의 프락시 시스템 비용이 증가하였다.

프락시 비용은 입력으로 들어온 레퍼런스 URI 값이 기 검증되었는지 확인하는 비용으로 해시 비용과 파일 입출력 비용을 포함하는 무결성 검증 비용에 비해 상당 부분 감소한다.

Table 1. Performance according to application size
(NR: Number of References, IVT: Integrity Validation Time, IVP: Integrity Validation Portion)

No	Application Name	Size (KB)	NR	TADS			PADS		
				Total (ms)	IVT (ms)	IVP (%)	Total (ms)	IVT (ms)	IVP (%)
1	MessagePort	148	10	70	12	16	61	7	10
2	ContextHistory	164	12	76	15	18	65	8	12
3	SampleAccount	172	14	77	16	20	65	10	14
4	OAuth2	212	17	88	20	23	71	11	15
5	Appcommon	284	22	104	29	28	82	16	20
6	FileManager	333	25	110	44	36	84	25	26
7	ResourceManagement	2800	176	494	306	62	310	159	51
8	MetadataExtractor	6700	11	719	653	91	386	322	83
9	Player	6800	14	736	672	91	382	332	84
10	Media	26000	48	2660	2545	96	1363	1268	93

V. 평 가

PADS의 검증 시스템의 성능을 평가하기 위해서 W3C의 규격을 준수하고 있는 TADS의 검증 시스템과 비교 실험을 진행하였다. 본 실험은 타이젠 3.0 운영체제와 쿼드코어 ARMv7 CPU, 1G RAM이 내장되어 있는 실제 단말인 Samsung Z3 상에서 수행되었다. 연산에 필요한 알고리즘은 TADS와 같은 알고리즘으로 진행 되었다. 해시값 연산 시에는 sha256 알고리즘이 사용되었으며[10], 서명 연산 시에는 rsa-sha256 알고리즘이 사용되었다[11].

5.1 애플리케이션 크기에 따른 성능 평가

애플리케이션의 크기에 검증 시스템 성능 비교 결과는 Table 1.과 같다. 실험 대상 애플리케이션은 타이젠 스튜디오의 공식 샘플 애플리케이션 중에서 크기 별로 10건을 선정하였다. 기본적인 애플리케이션의 서명 파일은 개발자 서명 파일과 배포자 서명 파일 2개로 구성되어 있다.

애플리케이션의 크기가 증가할수록 구성하는 단일 파일의 크기가 크거나 구성 파일이 많은 것을 의미한다. 즉, 애플리케이션의 크기가 커질수록 무결성 검증 내 해시 비용 및 파일 입출력 비용이 증가함으로 전체적인 검증 성능과 직접적인 연관이 있다.

TADS 검증 시스템의 경우 크기가 작은 MessagePort 애플리케이션의 무결성 검증 성능의 비중이 16%로 나타났다. 이는 애플리케이션의 크기가 작기 때문에 무결성 검증의 비중보다 서명 검증의 비중이 높은 것으로 판단된다. 크기가 MB 단위로 넘어가면서 무결성 검증의 비중이 더 높아지는 것을 볼 수 있으며 크기가 가장 큰 Media 애플리케이션의 무결성 검증 비중은 96%를 보였다. 이에 애플리케이션 전자서명 검증 중 무결성 검증 비중이 많은 부분을 차지하는 것을 확인하였다.

PADS의 검증 시스템 성능은 TADS의 검증 시스템 성능에 비해 실험 대상 애플리케이션 모두 감소하였으며 무결성 검증의 비중 또한 다소 감소하였다. 전체적인 검증 성능이 감소됨에 따라 무결성 검증의 비중이 작은 폭으로 감소 된 것으로 판단된다. 가장 작은 무결성 검증 비중은 10%로 측정되었으며 가장 큰 비중은 93%로 측정되었다.

Fig. 7.은 TADS의 검증 시스템 대비 PADS의

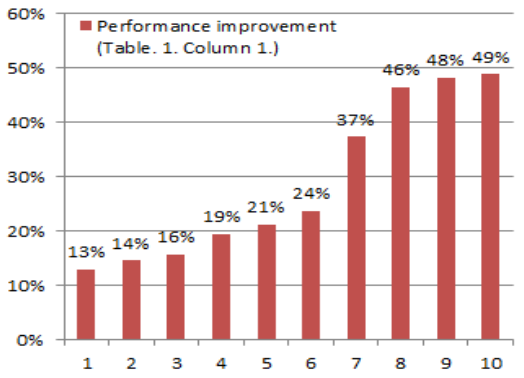

Fig. 7. Performance improvement of PADS

검증 시스템의 성능 개선 비율을 나타낸 것이다. 애플리케이션 크기가 1MB 미만인 경우 평균 약 20%의 성능 개선을 보였으며, 2.8M의 크기인 경우의 약 37% 개선을 보였다. 비교적 파일 크기가 큰 6.8MB의 경우 49%의 성능 개선을 나타내었다. 결론적으로 애플리케이션의 크기가 클수록 전체 검증 중 무결성 검증 비중이 많이 차지하였으며 PADS의 검증 시스템은 이를 효율적으로 개선하였음을 확인하였다.

5.2 서명 파일 개수에 따른 성능 평가

서명 파일 개수는 W3C 규약 내 명시되어 있는 대로 배포자 수의 증가에 따라 N개까지 늘어 날 수 있다. 이를 이용하여, 5.1절의 실험 결과를 바탕으로 서명 파일 개수를 임의로 변경하여 이에 따른 비교 실험을 진행하였다. 모수를 선정하기 위해 성능 개선 비율에 따라 20% 미만, 20% 이상 40% 미만, 40% 이상으로 군집화하여 서명 파일 개수를 증가시키면서 추가 실험을 진행하였다.

성능 개선 비율이 20%미만이었던 OAuth2의 경우, 성능 개선 비율이 최소 19%에서 최대 25%까지 성능이 개선됨을 확인하였다. 두 번째 성능 개선 비율이 20% 이상 40% 미만이었던 FileManager의 경우, 최소 24%에서 최대 37%의 성능 개선을 보였다. 마지막 성능 개선 비율이 40% 이상인 Media의 경우, 최소 49%에서 최대 78%의 성능 개선을 보였다.

Table 2. Performance according to SF
(SF: Signature File, IT: Improved time)

Name	SF	TADS (ms)	PADS (ms)	IT (%)
OAuth2	2	88	71	19
	3	124	98	21
	4	167	124	26
	5	207	156	25
File Manager	2	110	84	24
	3	167	115	31
	4	226	146	35
	5	280	177	37
Media	2	2660	1363	49
	3	3995	1399	65
	4	5329	1439	73
	5	6660	1477	78

VI. 결론 및 향후 계획

본 연구는 애플리케이션 전자서명 검증 시스템의 성능 강화를 위하여 검증 시스템 내 프락시를 적용하였다. PADS의 검증 시스템은 기존 TADS의 무결성 중복 검사로 인한 검증 시스템의 성능 저하를 해결하기 위해, 프락시를 시스템을 적용하여 레퍼런스의 검사 유무를 기록하고 중복된 레퍼런스는 무결성 검사에서 제외하는 시스템을 제안하였다. 시스템의 변경에 따라 PADS의 서명 파일 검증 사이에 애플리케이션의 위변조가 일어날 수 있으나, 최초 검증 시 구성 요소의 파일 시스템 내 타임스탬프를 캐시에 저장하고 이를 검증에 활용함으로써 보안성을 유지하였다.

타이젠 스튜디오 내의 공식 샘플 애플리케이션을 대상으로 제안된 시스템을 검증한 결과, 무결성 검증의 중복이 효율적으로 제거되었고 애플리케이션의 크기 및 레퍼런스가 증가할수록 제안 시스템이 효율적인 것을 확인할 수 있었다.

향후에는 프락시 시스템에 병렬 프로그래밍을 적용하여 무결성 검증 과정 중 레퍼런스 해시 비용을 개선하는 방안을 연구할 예정이다.

References

[1] "XML Digital Signatures for Widgets," https://www.w3.org/TR/widgets-digsig

[2] Lijang Yi, Guoqiang Bai and Guozhen Xiao, "Proxy multi-signature scheme: a new type of proxy signature scheme," IEEE Electronics Letters, vol. 36, no. 6, pp. 527-528, Mar. 2000.

[3] Ying Sun, Chunxiang Xu, Qi Xia and Yong Yu, "Analysis and Improvement of a Proxy Blind Multi-signature Scheme without a Secure Channel," Information Assurance and Security 2009. Fifth International Conference on, vol. 2, pp. 661-664, Oct. 2009.

[4] Shiangfeng Tzeng, Chengchi Lee and Minshiang Hwang, "A batch verification for multiple proxy signature," Parallel Processing Letters, vol. 21, no. 01, pp. 77-84, Mar. 2011.

[5] "TIZEN W3C API," https://www.tizen.org/tv/w3c_api/

[6] "TIZEN Applications Installer," https://wiki.tizen.org/Applications_Installer/

[7] Merkle R.C, "A Certified Digital Signature," Advances in Cryptology – CRYPTO' 89 Proceedings, vol. 435, pp. 218-238, Jul. 2001.

[8] "XML Signature Syntax and Processing (Second Edition)," https://www.w3.org/TR/xmldsig-core/

[9] "XML Schema Part 1: Structures Second Edition," https://www.w3.org/TR/xmlschema-1/

[10] "XML Encryption Syntax and Processing," http://www.w3.org/2001/04/xmlenc#sha256

[11] "Additional XML Security Uniform Resource Identifiers (URIs)," http://www.ietf.org/rfc/rfc4051.txt

〈저자소개〉

권 상 완 (Sangwan Kwon) 정회원
2014년 2월: 국민대학교 컴퓨터공학과 졸업
2017년 2월: 연세대학교 컴퓨터공학과 석사 졸업
2014년 2월~현재: 삼성전자 소프트웨어센터 재직
〈관심분야〉 정보보호, 시스템 소프트웨어

김 동 욱 (Donguk Kim) 정회원
2007년 8월: 고려대학교 산업시스템정보공학과 졸업
2007년 7월~2008년 8월: 한국생산성본부 연구원 재직
2014년 2월: KAIST 산업및시스템공학과 박사 졸업
2014년 3월~현재: 삼성전자 소프트웨어센터 재직
〈관심분야〉 정보보호, 시스템 보안

이 경 우 (Kyoungwoo Lee) 정회원
1995년 2월: 연세대학교 컴퓨터과학과 졸업
1997년 8월: 연세대학교 컴퓨터과학과 대학원 석사 졸업
1997년 7월~2003년 7월: 엘지전자 DTV연구소 선임연구원 재직
2008년 12월: 캘리포니아대학교 얼바인캠퍼스 컴퓨터과학과 박사 졸업
2011년 3월~현재: 연세대학교 컴퓨터과학과 부교수 재직
〈관심분야〉 내장형시스템, 디펜더블 컴퓨터시스템