

한국정보보호학회 동계학술대회

CISC-W'18

Conference on Information Security and Cryptography-Winter 2018

일자 2018년 12월 8일(토)

장소 세종대학교(군자관, 총무관)

주최  **한국정보보호학회**
Korea Institute of Information Security & Cryptology

주관  **세종대학교**
SEJONG UNIVERSITY

후원  **국가정보원**
National Intelligence Service Korea

 **행정안전부**

 **한국인터넷진흥원**

 **ETRI** **한국전자통신연구원**
Electronics and Telecommunications Research Institute

 **NSR**
국가보안기술연구소

 **LIG 넥스원**

 **보안프로젝트**
WWW.BSAPROJECT.COM

 **kakao**

Proceedings

포스터발표 트랙 F

POSTER P2 - 총무관 211호

세션	번호	발표논문
09:00 - 10:00	1	사이버 기만을 위한 사이버 기만 트리 모델 박창욱, 김영갑 (세종대학교)
	2	전자적 문서의 법적 효력 향상을 위한 검증방안 연구 -보험회사를 중심으로- 허원석 (고려대학교), 임상현 (동국대학교), 허재영 (한국교육학술정보원)
	3	중복된 난수 사용을 통한 EC-KCDSA 알고리즘 개인키 추출 방안 고주성, 광진 (아주대학교)
	4	Performance Analysis of IPsec in Kernel-land and User-land with multi-core implementation Sami Ullah, Joontae Choi, Heekuck Oh (한양대학교)
	5	비콘 기반 서비스의 접근 취약점에 대한 보안 방안 박준영 (대륜고등학교)
	6	가상화폐 거래소 탈중앙화 동향에 관한 연구 조형택, 엄홍열 (순천향대학교)
	7	공개물을 이용한 패턴 기반 보이스피싱 탐지 시스템 탁현주, 임수민, 김민서, 엄홍열 (순천향대학교)
	8	카드해킹의 문제점과 대비방안 노무승 (창원중앙고등학교), 이진우 (울산대송고등학교), 이정훈 (울산범서고등학교), 이창엽 (창원경일고등학교), 오현수 (부산진고등학교)
	9	스마트 헬스케어 서비스를 위한 IoT 센서기반 플랫폼 서비스 이원영, 김지영, 강희진, 한수민 (서울여자대학교)
	10	체크리스트 기반 소프트웨어 설계보안 연구 : 소프트웨어 개발보안제도를 중심으로 성선화, 최진영 (고려대학교)
	11	DexGuard 난독화 도구 특성 분석 김규호, 신용구, 이선준, 이정현 (숭실대학교)
	12	의료정보 공유 플랫폼을 위한 하이퍼레저 패브릭 응용 방안 이민기, 민서현, 지청민, 홍만표 (아주대학교)
	13	클라우드 환경에서 CP-ABE 방식을 활용한 안전한 데이터 접근 기법에 관한 연구 황용운, 이임영 (순천향대학교)
	14	블록체인 키 관리서버를 이요한 PGP+ 노창현, 이임영 (순천향대학교)
	15	컨소시엄 블록체인환경에서의 Privacy를 보장하는 안전한 통신시스템에 관한 연구 라경진, 이임영 (순천향대학교)
	16	크로스 아키텍처 바이너리 간 함수 유사도 분석 기법 심규원, 김지민, 지청민, 홍만표 (아주대학교)

중복된 난수 사용을 통한 EC-KCDSA 알고리즘 개인키 추출 방안

고주성*, 꺾진**

*아주대학교 컴퓨터공학과 정보보호응용및보증연구실

**아주대학교 사이버보안학과

Extraction a Private Key on EC-KCDSA Algorithm with Duplicated Random Nonce

Ju-Seong Ko*, Jin Kwak**

*ISAA Lab., Department of Computer Engineering, Ajou University.

**Department of Cyber Security, Ajou University.

요 약

전자 서명은 온라인 상에서 신원 인증 및 무결성 확인을 위해 사용될 수 있다. 대표적으로는 공개키 기반 구조의 공인인증서로 사용되어 왔으며, 최근 블록체인의 발달로 거래 정보의 무결성을 보장하기 위해 사용되고 있다. 암호 알고리즘들은 이론상으로는 안전하나 프로그램 구현 시 취약해질 수 있다고 평가되며, 이는 전자 서명 알고리즘에도 적용된다. 전자 서명 알고리즘의 경우 구현을 위해 여러 암호학적 요소가 수반되어야 한다. 난수 생성, 해시 알고리즘은 전자 서명의 안전성에 큰 영향을 미치며, 안전하지 않다고 평가되는 알고리즘 사용 시 전자 서명 알고리즘의 안전성을 저하시킬 수 있다. 따라서 본 논문에서는 국산 타원 곡선 기반 전자 서명 알고리즘인 EC-KCDSA에 대해 중복된 난수를 사용한 경우 개인키를 추출하는 과정을 분석하며 안전하다고 평가되는 난수 생성, 해시 알고리즘 사용의 중요성을 강조한다.

I. 서론

전자 서명은 데이터의 출처와 무결성을 확인할 수 있도록 데이터에 첨부되는 값으로 수신자가 전자 서명 검증을 통해 데이터 위조 확인 및 부인 방지 등의 서비스를 제공받을 수 있다[1]. 대표적인 국산 전자 서명 알고리즘으로는 KDSA, EC-KCDSA가 있으며 EC-KCDSA는 타원 곡선 상에서 KCDSA를 구현한 것이다[2, 3].

KCDSA와 EC-KCDSA는 국산 암호 활성화를 위해 국내의 신뢰받는 기관을 통해 개발 및 검토되었으며, 현재까지 알려진 최선의 공격 방법에 기반하여 공격자로 데이터를 보호할 수 있는 수준으로 평가된다[2, 3].

하지만 암호 알고리즘은 이론상 안전하더라도

실제 적용 시 취약해질 수 있다. 보안에 취약한 난수 사용은 암호 알고리즘의 안전성 저하를 초래할 수 있다는 것이 밝혀졌으며[4], 이는 전자 서명 알고리즘에도 적용될 수 있다.

EC-KCDSA의 서명 생성 준비 과정을 살펴보면 키와 난수 값을 생성하기 위해 난수 생성 및 해시 알고리즘을 사용하고 있다[2, 3]. 이는 안전하지 않은 난수 생성 및 해시 알고리즘 사용으로 EC-KCDSA의 안전성이 저하될 수 있음을 보여준다.

따라서 본 논문의 2장에서는 국산 전자 서명, 난수 생성 및 해시 알고리즘에 대해 설명한다. 3장에서는 중복된 난수 사용 시 발생할 수 있는 문제에 대해 분석하며, 4장에서는 분석한 문제를 기반으로 서명 값으로부터 개인키를 추출하는 과정을 보여준다. 5장에서는 개인키 추출 방식을 위해 권고되는 암호 알고리즘을 사용해야 함을 강조하며 결론을 맺는다.

본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 대학ICT연구센터육성 지원사업의 연구결과로 수행되었음 (IITP-2018-2016-0-00304)

II. 관련 연구

2.1 KCDSA

KCDSA(Korean Certificate-based Digital Signature Algorithm)는 임의의 길이를 갖는 메시지에 대해 전자 서명을 생성 및 검증할 수 있게 해주는 한국형 인증서 기반 전자 서명 알고리즘으로 인증, 무결성 및 부인 방지 등의 정보보호 서비스를 제공한다[2].

2.2 EC-KCDSA

EC-KCDSA(Korean Certificate-based Digital Signature Algorithm using Elliptic Curves)는 한국형 인증서 기반 타원 곡선 전자 서명 알고리즘으로, KCDSA와 동일한 정보보호 서비스를 제공한다[3]. EC-KCDSA의 서명 생성 과정은 (표 1)과 같으며 개인키는 d 로 표기한다.

(표 1) EC-KCDSA 서명 생성 과정

순서	과정
1	난수 k 를 임의로 선택 ($k < n-1$)
2	$kG = (x_1, y_1)$
3	$r = Hash(x_1)$
4	$c_Q = MSB(x_Q \ y_Q, L)$
5	$v = Hash(C_Q \ M)$
6	$e = (r \oplus v) \bmod n$
7	$s = d(k - e) \bmod n, (s \neq 0)$
8	(r, s) 반환

2.3 난수 생성 및 해시 알고리즘

불안전한 난수 생성 알고리즘 사용 시 개인 정보 유출 등의 해킹사고가 발생할 수 있다. 대표적으로는 RC4가 있으며, 해당 알고리즘은 사용하지 않는 것이 권고되고 있다[5].

안전한 난수 생성 알고리즘으로 난수를 생성하더라도 이를 사용 전 재가공하는 해시 알고리즘이 취약할 경우 문제가 발생할 수 있다. SHA-1의 경우 다른 입력에 대해 동일한 출력이 나오는 문제가 발생할 수 있으며[6], 이는 중복된 난수 생성을 사용하는 것과 동일한 결과를 초래할 수 있다.

III. 문제점 분석

EC-KCDSA는 개인키를 이용해 공개키를 생성하며, 이 공개키는 누구나 검증할 수 있도록 공개되어 있다. 따라서 여러 전자 서명들의 공개키가 같은 경우, 서명을 생성하기 위해 이용된 개인키가 동일함을 알 수 있다. 3.1절에서는 EC-KCDSA 알고리즘의 서명 생성의 마지막 단계를 나타내며, 3.2절은 중복된 난수 사용으로 발생할 수 있는 문제를 분석한다.

3.1 EC-KCDSA의 서명 생성 과정

EC-KCDSA를 이용한 전자 서명은 서명 값 r , s 를 생성하며, 생성 과정은 다음과 같다.

- $r = Hash(x_1), kG = (x_1, y_1)$
- $s = d(k - e) \bmod n$

3.2 발생할 수 있는 문제 분석

① 서명 값 r 이 같으면 동일한 난수를 사용함

전자 서명의 서명 생성 과정으로부터 난수 k 가 서명 값 r 생성에 관여하고 있음을 알 수 있다. 난수 k 외의 값들은 정의한 파라미터에 따라 고정된 값이기 때문에 난수 k 의 변화가 서명 값 r 의 변화에 영향을 미치고 있다. 즉, 난수 k 가 변경되지 않으면 서명 값 r 도 동일하다.

② 난수 중복 사용 시 개인키 추출 가능

서명 값 s 생성 과정은 3.1절에서 볼 수 있다. 이 과정을 동일한 개인키에 대해 중복된 난수 k 를 사용하여 두 개의 EC-KCDSA 전자 서명을 생성한 경우 다음과 같이 나타낼 수 있다.

- $s_1 = d(k - e_1) \bmod n, s_2 = d(k - e_2) \bmod n$

두 개의 식을 빼 후 정리하면 다음과 같이 나타낼 수 있다.

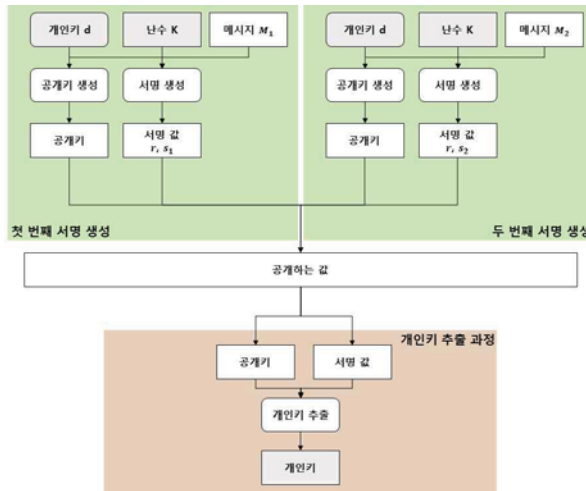
- $s_1 - s_2 = d(e_2 - e_1) \bmod n$
- $d = (s_1 - s_2)(e_2 - e_1)^{-1} \bmod n$

위의 정보들로부터 EC-KCDSA 전자 서명 알고리즘에 대해 중복된 난수 k 가 사용되는 경우, 개인키 추출이 가능함을 확인할 수 있다.

IV. 전자 서명 개인키 추출

4.1절에서는 [3]에서 제시한 EC-KCDSA(secp 224r)/SHA-256 환경의 전자 서명 알고리즘을 기반으로, 서명 값으로부터 개인키를 추출하는 과정을 보여준다. 4.2절에서는 실제 값을 통해 개인키가 추출될 수 있음을 보인다.

4.1 개인키 추출 과정



(그림 1) 전자 서명 생성 및 키 추출 과정

(그림 1)은 EC-KCDSA 서명 생성과 키 추출 과정을 요약한 것이다.

① 서명 생성 과정

(그림 1)의 두 서명 생성 과정에서 메시지 M, 서명 값 s만 다르다. 이는 동일한 개인키를 이용하기 때문에 동일한 공개키를 가지며, 중복된 난수를 사용하여 동일한 서명 값 r을 가지기 때문이다.

② 개인키 추출 과정

개인키 추출 과정에서 얻을 수 있는 정보는 메시지, 공개키 쌍, 두 개의 서명 값 쌍이며, 필요한 정보는 중간 값 e이다. 따라서 얻은 정보를 통해 e를 구하기 위해 (표 1)의 4, 5, 6 과정을 거친다.

$$d' = (s_1 - s_2)(e_2 - e_1)^{-1} \bmod n$$

필요한 정보가 계산되면 위의 식을 통해 개인키 d를 계산한다.

4.2 EC-KCDSA 개인키 추출

(표 2)는 EC-KCDSA 서명 생성에 사용된 값들을, (표 3)은 개인키 추출에 사용된 값들을 정리한 것이다.

(표 2) EC-KCDSA 서명 생성 과정

과정	정보	실제 값(Hex)
첫 번째 서명 생성	메시지	4E 65 73 73 61 67 65 5F 66 6F 72 5F 57 68 6F 49 73 5F 32 30 31 38 5F 6D 69 6E 69 5F 43 54 46
	난수 k	61 6E 64 20 43 72 79 70 74 6F 67 72 61 70 68 79 2D 57 69 6E 74 65 72
	개인키 d	43 30 6D 70 49 65 74 65 20 33 43 4B 43 44 53 41 20 50 72 6F 62 6C 65 6D 6D 6D 6D 21
	공개키 X_q	AC 3D D2 3F DC 20 1D 54 2B 7D 22 B8 C7 28 D9 14 F1 1E 84 27 50 48 C1 69 E7 A9 2C 8B
	공개키 Y_q	C5 9F AE BB 19 35 62 93 94 C2 10 1B FF 0F FA A7 0E 4E 1F 5A 5A 8C 25 31 0A B6 68 DD
	서명 값 r	D1 CD E0 4F 8B CA 2A 9E 11 59 EE 51 BE 22 20 B9 E0 11 AF 09 51 2A 33 4E B4 A3 44 A2
두 번째 서명 생성	서명 값 s_1	51 CA 2E 9F 17 72 C8 BE 65 80 BE 95 7B 9A 3B 5F 92 6B 24 68 E7 9A 53 F3
	메시지	65 63 2D 6B 63 64 73 61 20 64 69 67 69 74 61 6C 20 73 69 67 6E 61 74 75 72 65
	난수 k	첫 번째와 동일
	개인키 d	
	공개키 X_q	
	공개키 Y_q	
서명 생성	서명 값 r	9C 8B E5 28 CB A4 33 98 E8 CC A0 A3 7F 78 46 B3 A8 12 C7 59 D3 9E C3 59 CC 86 FE C7
	서명 값 s_2	

(표 3) EC-KCDSA 개인키 추출 과정

과정	정보	실제 값(Hex)
개 인 키 추 출	모듈러 n	FF FF FF FF FF FF FF FF FF FF FF FF FF 16 A2 E0 B8 F0 3E 13 DD 29 45 5C 5C 2A 3D
	$s_1' - s_2'$	70 BF AE D2 86 25 FB 06 2E A6 28 1A B5 07 8E 84 B4 40 64 43 D2 A9 8A 54 77 6F 7F 69
	$e_2' - e_1'$	A5 B6 B0 42 58 96 4D 2A 1C A0 DE 3A 19 6A A2 9B 5C E9 FB 5F 2B BE E7 BD AE 33 57 19
	$\text{inv}(e_1' - e_2')$	3D 08 07 74 8D 8A A9 CF 5A EE 91 1D 66 37 5B 3E FC BA 6B 75 61 9E B7 43 B3 FB F0 09
	d'	43 30 6D 70 49 65 74 65 20 33 43 4B 43 44 53 41 20 50 72 6F 62 6C 65 6D 6D 6D 6D 21

(표 2)의 개인키 d 와 (표 3)의 d' 을 비교한 결과 동일한 것을 확인할 수 있으며, 이는 개인키를 성공적으로 추출한 것이다.

V. 결론

본 논문에서는 전자 서명을 사용하기 위해 부가적으로 필요한 암호학적 요소로부터 취약점이 발생할 수 있음을 언급하였다. 이는 전자 서명 알고리즘이 안전하더라도 프로그램 구현 시에 취약점이 발생할 수 있음을 의미한다.

동일한 개인키를 통해 중복된 난수 값을 사용하는 경우 개인키를 추출할 수 있는 방안에 대해 분석하였다. 실생활에서 공인인증서의 재발급 기간이 짧지 않기 때문에, 난수 재사용으로 인한 위협이 발생할 수 있다.

따라서 난수 생성 및 해시 알고리즘 등 난수 사용에 영향을 미치는 알고리즘들은 취약하지 않은 버전을 사용 중인지 확인해야 한다. 사용 중인 알고리즘의 취약점이 알려진 경우, 빠른 시일 내에 취약하지 않은 알고리즘으로 대체해야 한다.

[참고문헌]

- [1] 한국정보통신기술협회, “전자 서명, digital signature”, 한국정보통신기술협회 정보통신 용어사전
- [2] 한국정보통신기술협회, “부가형 전자 서명 방식 표준 - 제2부: 한국형 인증서 기반 전자 서명 알고리즘(KCDSA)”, 정보통신단체 표준, 2016.12.
- [3] 한국정보통신기술협회, “부가형 전자 서명 방식 표준 - 제3부: 한국형 인증서 기반 타원 곡선 전자 서명 알고리즘(EC-KCDSA)”, 정보통신단체표준, 2016.12.
- [4] KISA 침해사고대응단 종합상황대응팀, “구글, 안드로이드 취약점으로 인한 비트코인 탈취 인정”, KISA 인터넷 보호나라&Krcert 자료실 게시판, 2013.08.
- [5] 서울대학교 정보화본부, “유무선 공유기 보안가이드”, 서울대학교, 2018.05.
- [6] CWI Amsterdam, Google Research, “The first collision for full SHA-1”, <https://shattered.io/>, 2017.01.