Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.31, NO.3, Jun. 2021 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2021.31.3.517

양자 컴퓨팅 환경에서 안전한 다중 서명 기법*

고 찬 영,[1†] 이 영 경,[1] 이 광 수,[2] 박 종 환[3‡]
[1]고려대학교 (대학원생), [2]세종대학교 (교수), [3]상명대학교 (교수)

A Post-Quantum Multi-Signature Scheme*

Chanyoung Ko,[1†] Youngkyung Lee,[1] Kwangsu Lee,[2] Jong Hwan Park[3‡]
[1]Korea University (Graduate student), [2]Sejong University (Professor),
[3]Sangmyung University (Professor)

요 약

최근 양자 컴퓨터의 개발이 가속화되면서 기존 인터넷 환경에서 사용되고 있는 인수분해 및 이산대수 기반의 전자서명 기법들의 안전성에 대한 문제가 제기되고 있다. 이에 대응하기 위해 미국 국립표준기술연구원(NIST)의 표준화 작업을 비롯하여 양자 컴퓨팅 환경에서 안전한 여러 가지 전자서명 기법들이 제시되고 있다. 본 논문에서는 2018년 Behnia 등이 CCS 컨퍼런스에서 발표한 TACHYON 전자서명 기법을 이용한 다중 서명 기법을 설계하여 제시하고, 안전성을 증명하고자 한다. 다중 서명 기법은 최근 많은 관심을 받고 있는 암호화폐 분야에서 전자지갑의 개인키 의존도를 분산하여 보다 안전한 전자지갑 시스템을 구성할 수 있는 핵심 기술로, 최근 많은 연구자들과 개발자들이 관심을 갖고 있는 전자서명 응용 분야이다. 본 논문에서 제시하는 다중 서명 기법은 일반 공개키 모델(plain public key model)에서 공개키 결합(public key aggregation)이 가능한 기법으로, 부가적인 영지식 증명(zero-knowledge proof) 기법이 필요하지 않으며, 결합 된 단일 공개키만으로 효율적인 시스템을 구성할 수 있는 기법이다.

ABSTRACT

Recently, the acceleration of the development of quantum computers has raised the issue of the safety of factorization and discrete logarithm based digital signature schemes used in existing Internet environments. To solve the issue, several digital signature schemes are presented that are safe in post-quantum computing environments, including standardization work by the National Institute of Standards and Technology(NIST). In this paper, we design and present a multi-signature scheme based on the TACHYON announced by Behnia et al. in 2018 CCS conference, and prove the security. Multi-signature schemes are key techniques that can distribute the dependence of cryptocurrency-wallet on private keys in the cryptocurrency field, which has recently received much attention as an digital signature application, and many researchers and developers have recently been interested. The multi-signature scheme presented in this paper enables public key aggregation in a plain public key model, which does not require additional zero-knowledge proof, and can construct an effective scheme with only an aggregated public key.

Keywords: Post-quantum, multi-signature, key aggregation, generalized compact knapsack

Received(04. 27. 2021), Modified(05. 25. 2021), Accepted(05. 25. 2021)

* 본 연구는 고려대 암호기술 특화연구센터를 통한 방위사업청과 국방과학연구소의 연구비 지원으로 수행되었습니다.

† 주저자, 2020572059@korea.ac.kr
‡ 교신저자, jhpark@smu.ac.kr(Corresponding author)

I. 서 론

현재 인터넷 환경에서 사용되는 공개키 암호, 전자서명, 키 교환 등의 공개키 시스템은 인수분해 및 이산대수의 어려움에 기반하는 기법들을 사용한다. 1994년 Shor는 이러한 인수분해 문제나 이산대수 문제를 양자 컴퓨팅 환경에서 효율적으로 해결할 수 있는 알고리즘을 제안하였다[11]. 최근 양자 컴퓨터의 개발 가속화 및 실현 가능성이 높아짐에 따라 양자 컴퓨팅 환경에서 안전한 공개키 시스템 개발의 필요성이 대두되고 있으며, 미국 국립표준기술연구원(NIST)의 표준화 작업을 비롯한 노력이 계속되고 있다.

전자서명 기법은 사용자 및 메시지 인증 기능을 제공할 수 있는 암호학적 기법으로, 전자상거래 등 인터넷 환경에서 신뢰가 필요한 작업에 필수적인 기술이다. 전자서명 기법은 최근 가장 큰 이슈 중 하나인 블록체인을 활용한 암호화폐 기술에서도 전자지갑의 형태로 이용되고 있어 암호화폐 기술 관련 연구자 및 개발자들의 많은 관심을 받고 있다. 전자서명 기법은 공개키/개인키로 구성되어 개인키를 이용하여 서명을 하고 공개키로 검증하는 단순한 형태의 기술이며, 다중 서명 기법은 전자서명에서 확장된 기술로써 다수의 사용자가 자신의 공개키/개인키 쌍을 생성한 후 하나의 메시지에 대한 서명을 생성하는 것이다.

다중 서명 기법은 암호화폐의 전자지갑에 응용될 수 있으므로 최근 많은 관심을 받고 있다[1,4]. 기존의 전자서명으로 전자지갑을 구현할 경우 단일 개인키의 분실 및 노출은 전자지갑에 저장된 모든 가치의 분실을 초래할 수 있으므로 문제가 될 수 있다. 하나의 개인키로 서명하는 서명 기법과 달리 다수의 사용자의 개인키를 사용한 다중 서명 기법을 응용하여 사용자, 이상 거래탐지 서버, 디지털 금고 등 단일 주체가 아닌 다중 노드의 서명 생성으로 거래하는 시스템을 구축하여 일정 수 이상의 개인키를 필요로 함으로서 하나의 개인키에 전자지갑의 모든 가치가 의존되는 문제를 해결할 수 있다.

다중 서명에는 일반 공개키 모델(plain public key model)과 공개키 결합(public key aggregation)이라는 특성이 있다[8]. 공개키 결합이라는 특성은 다수의 공개키가 하나의 공개키로 결합할 수 있고, 여러 명의 개인키 소유자가 합의하여 서명한 다중 서명 검증 시 결합된 공개키 하나만으로 서명 검증이 가능한 특성을 의미한다. 일반 공개키 모델은 악의적인 공개키를 생성한 공격자로부터 시스템을 안전하게 하기 위해 공개키에 대응하는 개인키의 영지식 증명(zero-knowledge proof)을 부가적으로 수행하지 않아도 안전한 모델을 의미한다. 최근 연구에서는 일반 공개키 모델에서 공개키 결합이 가능한 Schnorr 기반의 다중 서명 기법[4]이 제시되었다. 본 논문에서는 공개키 모델에서 공개키 결합이 결합이 가능하면서 양자 컴퓨팅 환경에서도 안전한 TACHYON[12] 기반의 다중 서명 기법을 제시한다. 추가적으로 최근 연구[6]에서 제시된 TACHYON 서명에 대한 다중 사용자 공격방법의 대안책인 솔트를 사용하는 기법을 본 논문에서도 적용하여 같은 공격 방법에 안전할 수 있도록 한다.

II. 배경지식

이번 장에서는 공개키 서명에 대한 일반적인 기법과 본 논문에서 제안하는 다중 서명 기법의 기본 설계 토대가 되는 TACHYON 서명 기법과 해당 서명 기법이 기반하는 문제인 일반화된 콤팩트 냅색(generalized compact knapsack)문제를 설명한다.

2.1 공개키 서명

공개키 서명은 $KeyGen, Sign, Verify$ 세 개의 알고리즘으로 구성되며, 각 알고리즘은 다음과 같이 정의한다.

- $KeyGen(1^\lambda)$: 입력값으로 1^λ를 받아 개인키인 sk와 공개키인 pk를 출력한다.
- $Sign(M,sk,pk)$: 입력값으로 메시지 M, 개인키 sk와 공개키 pk를 받아 알고리즘을 실행하여 서명인 σ를 출력한다.
- $Verify(M,\sigma,pk)$: 입력값으로 메시지 M, 서명 σ, 공개키 pk를 받아 알고리즘을 실행하여 서명이 정당한 경우 1을, 그렇지 않은 경우 0을 출력한다.

공개키 서명은 모든 키 쌍과 모든 메시지에 대해서 다음 수식이 성립할 때 정확성(correctness)을 갖는다고 정의한다.

$$Verify(M,Sign(M,sk,pk),pk) \rightarrow 1$$

2.2 Generalized Compact Knapsack

모든 링 R과 부분 집합 $S \subset R$ 그리고 정수 $\mu \geq 1$에 대해 GCK 함수는 다음과 같이 정의된다[2].

$$GCK_A(x) = \sum_{i=1}^{\mu} x_i \cdot a_i$$

이때 x_i, a_i는 각각 $x \in S^{\mu}$, $A \in R^{\mu}$ 벡터의 i번째 링의 원소이다.

정리 2.1. 위에서 정의한 GCK 함수는 사이클릭 래티스(cyclic lattice) 문제의 어려움에 기반하여 일방향성(one-wayness)을 만족한다[2].

2.3 TACHYON 서명

TACHYON 서명[12]은 공개키 서명으로 세 개의 알고리즘 $TS.KeyGen$, $TS.Sign$, $TS.verify$으로 구성되며, 각 알고리즘은 다음과 같이 정의한다. 알고리즘에서 쓰이는 $Sample(\gamma)$은 각각 범위 $[-\gamma,\gamma]$에서 균등하게 뽑은 개체들로 이루어진 링 벡터 $r \in R^{\mu}$을 생성하는 함수이고, PRF는 의사난수함수로서 (0,1)로 이루어진 링 벡터 $x \in R^{\mu}$를 생성한다. ξ와 ρ는 각각 GCK 함수의 일방향성의 안전성에 관련된 파라미터이며, H_1과 H_2는 각각 랜덤 오라클 모델에서의 해시함수로 정의한다.

• $TS.KeyGen(1^{\lambda})$: PRF의 키 $z \in \{0,1\}^{\lambda}$를 선택하고, 상수 l개의 링 벡터 $x_i \in R^{\mu}$를 $PRF(z,i)$ 함수를 이용하여 생성한다. (즉, $PRF(z,i) \rightarrow x_i$, $i=1,...,l$) l개의 링 벡터 x_i를 각각 입력으로 다음과 같이 링 $y_i = GCK_A(x_i)$를 계산하고 개인키 $sk=z$와 공개키 $pk=(l,k,y=(y_{1,...,}y_l),H_1,H_2)$를 출력한다.

• $TS.Sign(M,sk,pk)$: 작은 계수 값을 갖는 링 벡터를 추출하는 알고리즘을 통해 링 벡터 $Sample(\xi-1) \rightarrow r=(r_1,...,r_{\mu}) \in R^{\mu}$을 생성한다. r을 이용하여 다음과 같이 $GCK_A(r) \rightarrow v \in R$를 계산하고, 해시함수 H_1을 통해 $H_1(v) \rightarrow h$를 계산한다. 다음으로 메시지 M과 h를 해시함수 H_2의 입력값으로 넣어 k개의 인덱스 셋 $\langle idx_1 \| \cdots \| idx_k \rangle$를 출력한다. (즉, $H_2(M \| h) \rightarrow \langle idx_1 \| \cdots \| idx_k \rangle$) 상수 k개의 인덱스가 주어지면 개인키 z를 이용하여 $PRF(z,idx_j) \rightarrow x_{idx_j}$ 값을 계산한 뒤 $s=(\sum_{j=1}^{k} x_{idx_j}) + r \in R$를 계산한다. 마지막으로 상숫값 ρ에 대해 $\| s \|_{\infty} < (\xi-\rho)$를 만족하는지 확인하고 서명 값 $\sigma=(s,h)$를 출력한다. 만약 $\| s \|_{\infty} \geq (\xi-\rho)$라면 링 벡터 r을 뽑는 과정부터 다시 시작한다.

• $TS.Verify(M,\sigma,pk)$: 주어진 서명 값 $\sigma=(s,h)$와 공개키 $pk=(l,k,y=(y_{1,...,}y_l),H_1,H_2)$를 이용하여 메시지 M에 대한 서명 검증을 다음과 같이 수행한다. 먼저 $\| s \|_{\infty} < (\xi-\rho)$를 만족하는지 확인하고 아니라면 0을 출력한다. 그 다음으로 해시함수 값 $H_2(M \| h) \rightarrow \langle idx_1 \| \cdots \| idx_k \rangle$를 계산하고 해당 인덱스 셋을 이용하여 $v' = GCK_A(s) - \sum_{j=1}^{k} y_{idx_j}$ 값을 계산하고 $H_1(v') = h$를 성립하는지 확인한다. 성립하는 경우 서명이 검증됐음을 의미하는 1을 출력하고, 그렇지 않은 경우 0을 출력한다.

정리 2.2. TACHYON 서명은 GCK 함수의 일방향성에 기반하여 랜덤 오라클 모델에서 선택 메시지 공격에 대한 위조 불가능성(EUF-CMA)을 만족한다[12].

III. 다중 서명 기법

3.1 다중 서명의 정의

다중 서명 기법은 $MS.Setup$, $MS.KeyGen$, $MS.Sign$, $MS.AggKey$, $MS.Verify$의 다섯 개 알고리즘으로 구성되며, 각 알고리즘은 다음과 같이 정의한다.

• $MS.Setup(1^{\lambda})$: 시큐리티 파라미터인 1^{λ}를 입력값으로 받아 시스템 파라미터인 pp를 출력한다.

• $MS.KeyGen(pp)$: 시스템 파라미터인 pp를 입력값으로 받아 개인키인 sk와 공개키인 pk를 출력한다.

• $MS.Sign(M,sk,pkl,pp)$: 공개키 리스트 pkl에 있는 속한 사용자들 간의 다중 서명을 생성하는 프로토콜로 각각의 사용자들은 자신의 비밀키 sk를 이용하여 다중 서명인 σ를 생성한다.

• $MS.AggKey(pkl)$: 공개키 리스트 pkl를 입력으로 받아 리스트에 속한 공개키들을 결합한 공개키 apk를 생성하는 알고리즘이다.

• $MS.Verify(M,\sigma,apk,pp)$: 메시지 M, 서명 σ, 결합된 공개키 apk를 입력으로 서명 검증을 수행하는 알고리즘으로 검증이 되면 1을, 아니면 0을 출력한다.

다중 서명은 모든 키 쌍과 모든 메시지에 대해서 다음 수식이 성립할 때 정확성(correctness)을 갖는다고 정의한다.

$$MS.Verify(M, MS.Sign(M,sk,pkl), MS.AggKey(pkl)) \rightarrow 1$$

정의 3.1. 선택 메시지 공격에 대한 존재적 위조 불가능성은 다음의 도전자 C와 다항 시간 공격자 A사이의 행동으로 정의할 수 있다.

• $Setup$: C는 $MS.Setup(1^{\lambda})$ 알고리즘을 통해 시스템 파라미터 pp를 출력하고, $MS.KeyGen(pp)$를 통해 도전 키 쌍인 (sk^*,pk^*)를 출력하여 A에게 (pp,pk^*)를 전달한다.

• $Signature\ Query$: A는 pk^*를 포함한 공개키 집합인 $pkl = pk_{i=1}^{n}, pk^* \in pkl$에 대해 어떤 메시지에 대한 서명도 요청할 수 있다. 요청을 받은 C는 $MS.Sign(M,sk,pkl,pp)$를 통해 다중 서명인 σ를 A에게 전달한다.

• $Challenge$: A는 공개키 집합인 pkl^*에 대한 메시지 M^*의 서명인 σ^*를 출력한다. 이에 대해 다음 세 조건을 만족한 경우 1을 출력한다.

1) $MS.Verify(M^*,\sigma^*,MS.AggKey(pkl^*),pp) = 1$
2) M^*은 $Signature\ Query$에서 요청되지 않은 메시지이다.
3) $pk^* \in pkl$

공격자 A의 공격 이점은 다음과 같이 정의한다.

$$Adv_A^{MS}(\lambda) = \Pr[C = 1]$$

위의 공격 이점이 무시해도 될 정도로 작다면, 다중 서명은 선택 메시지 공격에 대해 존재적 위조가 어렵다고 할 수 있다.

3.2 TACHYON 기반 다중 서명 기법[12]

이번 절에서는 TACHYON 기반의 다중 서명 기법을 구성하는 다섯 개의 알고리즘을 아래와 같이 설명한다. TACHYON 기법에서 사용된 해시함수 H_1, H_2 이외에도 추가적인 랜덤오라클 모델에서의 H_c, H_a 해시함수를 이용한다. H_c는 GCK 함수의 출력값을 입력으로 받아 $\{0,1\}^{\lambda}$에 속하는 커밋먼트를 출력하는 해시함수고, H_a는 공개키 목록과 특정 공개키를 입력으로 받아 상수 l크기의 랜덤한 순열(permutation) π을 출력하는 해시함수다.

• $TM.Setup(1^{\lambda})$: 1^{λ}를 입력값으로 받아 시스템 파라미터인 $pp = (l,k,H_1,H_2,H_c,H_a)$를 출력한다.

• $TM.KeyGen(pp)$: $z \in (0,1)^{\lambda}$를 랜덤으로 선택한 후 $i = 1...l$에 대해 $PRF(z,i) = x_i \in R^{\mu}$를 출력한다. 다음으로 $y_i = GCK_A(x_i)$를 계산하여 개인키 $sk = z$와 공개키 $pk = (y = \langle y_1,...,y_l \rangle)$를 출력한다.

• $TM.AggKey(pkl)$: $pk_i = (y_i = (y_{i,1},...,y_{i,l}))$에 대한 공개키 리스트를 $pkl = \{pk_1,...,pk_n\}$로 설정하고, $i = 1,...,n$에 대해 H_a를 통해 랜덤 순열인 $\pi_j \leftarrow H_a(pkl \parallel pk_j)$를 계산하고, $idx = 1,...,l$에 대해 $ay_{idx} = \sum_{j=1}^{n} y_{j,\pi_j(idx)}$와 같이 각각의 공개키 벡터의 값을 순열화하여 더해준다. 이 후 결합된 공개키인 $apk = (ay = (ay_1,...,ay_l))$를 출력한다.

• $TM.Sign(M,sk,pkl,pp)$: $sk = sk_{i^*} = z_{i^*}$으로 가정하고 pk_{i^*}를 sk_{i^*}와 대응되는 공개키라고 가정한다. 공개키 리스트 $pkl = \{pk_1,...,pk_n\}$에 대해 $i^* \in [1,n]$이다. 알고리즘은 아래의 과정에 의해 서명 σ를 생성하게 된다.

(1) $r_{i^*} \leftarrow Sample(\xi - 1)$을 다음과 같이 생성하고 $v_{i^*} \leftarrow GCK_A(r_{i^*})$를 계산한다. 계산한 v_i를 해시함수의 입력값으로 하여 $t_{i^*} = H_c(v_{i^*})$값을 계산하고 t_{i^*}를

모든 서명자들에게 전송하여 $j \neq i^*$인 모든 서명자로부터 t_j를 받는다.

(2) 모든 참여 서명자들로부터 t_j값을 전송받았음을 확인한 후 모든 서명자에게 v_{i^*}값을 전송하고 $j \neq i^*$인 모든 서명자로부터 v_j를 받는다. 이 후 $j \neq i^*$인 모든 v_j값에 대해 $t_i = H_c(v_i)$가 성립하는지 확인하고, 아니라면 프로토콜을 중단한다.

(3) $TM.AggKey$ 알고리즘을 사용하여 결합된 공개키인 $apk \leftarrow TM.AggKey(pkl)$를 생성하고, (2)에서 전달받은 값으로 $av = \sum_{j=1}^{n} v_j$를 계산하고, 이를 해시 함수의 입력값으로 하여 $h = H_1(av)$를 계산한다. 이후 H_2를 이용하여 $\langle idx_1 \| \cdots \| idx_k \rangle \leftarrow H_2(apk \| h \| M)$를 계산하고, 해시함수인 H_a를 이용하여 순열 $\pi_{i^*} \leftarrow H_a(pkl \| pk_{i^*})$를 생성한다. $(idx_j, j = 1,...,k)$에 대해 $x_{i^*,\pi_{i^*}(idx_j)} = PRF(z_{i^*}, \pi_{i^*}(idx_j))$를 계산하고, 마지막으로 $s_{i^*} = (\sum_{j=1}^{k} x_{i^*,\pi_{i^*}(idx_j)}) + r_{i^*}$를 계산하여 모든 서명자들에게 전송하고 $j \neq i^*$인 모든 서명자로부터 s_j를 받는다. 모든 서명자가 전송한 s_j값으로 $s = \sum_{j=1}^{n} s_j$을 계산한다. 이때 $\| s \|_\infty < n(\xi - \rho)$인지 확인하고, $\| s \|_\infty \geq n(\xi - \rho)$이라면 과정(1)부터 재시작하고 아니라면 결합 서명인 $\sigma = (s, h)$를 최종 서명 값으로 결정한다.

• $TM.Verify(M, \sigma, apk, pp)$: 먼저 주어진 서명값 s에 대해 $\| s \|_\infty < n(\xi - \rho)$이 성립하는지 확인하고 아닌 경우 서명이 정당하지 않다는 뜻의 0을 출력한다. 이후 $(idx_1 \| ... \| idx_k) \leftarrow H_2(apk \| h \| M)$를 계산한 후 해당 인덱스 셋을 이용하여 검증에 필요한 $av' = GCK_A(s) - \sum_{j=1}^{k} ay_{idx_j}$ 값을 계산하고 이를 해시함수의 입력값으로 하여 $H_1(av') = h$이 성립하는지 확인한다. 성립한다면 서명이 검증됐음을 의미하는 1을 출력하고, 그렇지 않은 경우 0을 출력한다.

해당 기법의 정확성은 다음의 수식으로 확인할 수 있다.

$$GCK_A(s) - \sum_{j=1}^{k} ay_{idx_j} = GCK_A(\sum_{i=1}^{n} s_i) - \sum_{j=1}^{k} ay_{idx_j}$$

$$= \sum_{i=1}^{n} GCK_A(\sum_{j=1}^{k} x_{i,\pi_i(idx_j)} + r_i) - \sum_{j=1}^{k}\sum_{i=1}^{n} y_{i,\pi_i(idx_j)}$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{k} GCK_A(x_{i,\pi_i(idx_j)}) + \sum_{i=1}^{n} GCK_A(r_i) - \sum_{j=1}^{k}\sum_{i=1}^{n} y_{i,\pi_i(idx_j)}$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{k} (GCK_A(x_{i,\pi_i(idx_j)}) - y_{i,\pi_i(idx_j)}) + \sum_{i=1}^{n} GCK_A(r_i)$$

$$= \sum_{i=1}^{n} v_i = av$$

위 과정을 통해, 서명 생성 과정에서 생성한 $h = H_1(av)$값과 서명 검증 시 생성한 $h' = H_1(av')$값이 같다는 것을 알 수 있다.

IV. 안전성 증명

이번 장에서는 forking lemma[3]를 응용한 multiple forking lemma를 적용하여 앞서 설계한 TACHYON 기반 다중 서명 기법의 안전성을 GCK 함수의 일방향성에 기반하여 증명하고자 한다.

정리 4.1. 3.2절에서 제시된 TACHYON 기반 다중 서명 기법은 GCK 함수의 일방향성에 기반하여 랜덤 오라클 모델에서 선택 메시지 공격에 대한 위조 불가능성(EUF-CMA)을 만족한다.

증명. 3.2절에서 제시된 다중 서명 기법의 선택 메시지 공격에 대한 위조가 가능한 공격자 F가 있다고 가정한다면, GCK 함수의 일방향성에 반하는 알고리즘 B가 존재한다는 것을 보여 정리 4.1을 증명하고자 한다. 이를 위해 먼저 위조 공격자 F의 도전자 역할을 수행하는 래핑 알고리즘 A를 설계한다. A는 EUF-CMA 게임에서 도전자 역할을 수행하여 위조 서명을 출력하는 알고리즘이다. 이 증명에서는 multiple forking lemma를 적용하여 알고리즘 A를 같은 랜덤 테이프에 대해 하나의 분기점 이후에 각각 하나의 분기점을 잡아 총 네 번 실행하여 네 개의 위조 서명을 얻는다. 이 후 네 개의 위조 서명을 이용하여 GCK 함수의 일방향성을 부정할 수 있는 B를 설계한다.

알고리즘 A는 $pk^* = (y_1, y_2, ..., y_n)$, $\pi_1, ..., \pi_q$, $I_1, ..., I_q$와 랜덤 코인 ρ_F을 입력값으로 받아 출력값으로 $(i_0, i_1, pkl, av, s, \pi, I_{i_1})$을 출력한다. 이 출력값에서 $i_0, i_1 \in 1, ..., q$ 이고, $pkl = \{pk_1, ..., pk_n\}$ $(pk^* \in pkl)$ 이다. 또한 $pk_i = pk^*$인 i에 대해 $\pi_i = \pi_{i_0}$인 순열 $\pi = (\pi_1, ..., \pi_n)$를 가진다. 각각 H_1, H_2, H_c, H_a의 값을 저장하는 테이블인 T_1, T_2, T_c, T_a를 가지며 알고리즘 시작 시에 빈 테이블로 초기화한다. 또한 두 카운터인 $ctr_0 := 0$와 $ctr_1 := 0$를 포함한다. 이 후 F에게 pk^*를 입력값으로 주어 실행하며, 각 $query$에 대한 응답은 다음과 같이 진행된다.

• $H_1(av)\, query$: $T_1(av)$를 출력하며, $T_1(av)$가 정의되어 있지 않다면 $T_1(av) \xleftarrow{\$} (0,1)^\lambda$값을 저장 후 출력한다.

• $H_2(ay \parallel h \parallel M)\, query$: $T_2(ay \| h \| M)$를 출력하며, $T_2(ay \parallel h \parallel M)$가 정의되어 있지 않다면 카운터값을 $ctr_1 := ctr_1 + 1$로 변경하고 $T_2(ay \parallel h \parallel M) := I_{ctr_1}$ 값을 저장 후 출력한다.

• $H_c(v)\, query$: $T_c(v)$를 출력하며, $T_c(v)$가 정의되어 있지 않다면 $T_c(v) \xleftarrow{\$} (0,1)^\lambda$값을 저장 후 출력한다.

• $H_a(pkl \parallel pk)\, query$: $T_a(pkl \parallel pk)$를 출력하며, $T_a(pkl \parallel pk)$가 정의되어 있지 않다면 카운터값을 $ctr_0 := ctr_0 + 1$로 변경 후 pk^*에 대해서는 $T_a(pkl \parallel pk^*) := \pi_{ctr_0}$값을 저장하고, $pk' \in pkl / pk^*$인 모든 pk'에 대해서는 $T_a(pkl \| pk') := \pi \xleftarrow{\$} \Pi$를 저장한 후 $T_a(pkl \parallel pk)$를 출력한다.

• $Signature\, query\, (pkl, M)$:

1) $pk^* \notin pkl$인 경우, $\perp$를 출력한다. 그렇지 않은 경우 $pk_i = (y_i = (y_{i,1}, ..., y_{i,l}))$에 대해 공개키 리스트를 $pkl = (pk_1 = pk^*, pk_2, ..., pk_n)$로 정의한다.

2) $T_a(pkl \parallel pk^*)$가 정의되어 있지 않으면, 내부 $H_a(pkl \parallel pk^*)\, query$를 통해 정의한다. 이 내부 $query$는 각각의 $i \in (1, ..., n)$에 대해 $T_a(pkl \parallel pk_i)$가 정의되며, $\pi_i = T_a(pkl \parallel pk_i)$값을 설정하고 $ay_j = \sum_{i=1}^{n} y_{i,\pi_i(j)}$식을 통해 $ay = (ay_1, ..., ay_l)$값을 계산한다.

3) $ctr_1 := ctr_1 + 1$로 카운터 값을 변경하고, $h \xleftarrow{\$} (0,1)^l$와 $s_1 \xleftarrow{\$} sample(\xi - \rho - 1)$값을 랜덤으로 선택한다. 이후 $I_{ctr_1} = (idx_1 \| \cdots \| idx_k)$에 대해 $I := I_{ctr_1}$를 설정하고, $v_1 := GCK_A(s_1) - \sum_{j=1}^{k} y_{1,\pi_1(idx_j)}$ 값을 계산한다.

4) $T_c(v_1)$가 이미 정의된 경우, $\perp$를 출력한다. 그렇지 않은 경우 내부 $H_c(v_1)\, query$를 통해 값을 정하고 $t_1 := T_c(v_1)$을 위조 공격자에게 보낸다.

5) 위조 공격자로부터 커밋먼트 $t_2, ..., t_n$를 받은 다음 각 $i \in 2, ..., n$에 대해 $t_i = T_c(v_i)$를 만족하는 v_i를 찾고 각각 하나의 값을 갖는지 확인한다.

6) 위의 조건이 충족되지 않을 경우 실행을 중단한다. 그렇지 않은 경우 위조 공격자에게 v_1을 보내고, $av := \sum_{i=1}^{n} v_i$를 계산한다.

7) $T_1(av)$ 혹은 $T_2(ay \parallel h \parallel M)$가 이미 정의되어 있는 경우 실행을 중단한다. 그렇지 않은 경우 각 값을 $T_1(av) := h$, $T_2(ay \| h \| M) := I$으로 정의한 후 위조 공격자에게 v_1을 보낸다.

8) 위조 공격자 F로부터 $v_2, ..., v_n$를 받은 후 모든 $i \in (2, ..., n)$에 대해 $H_c(v_i) = t_i$가 성립하는지 확인한다.

9) 성립하지 않는다면 서명 단계를 중단하고, 그렇지 않다면 위조 공격자에게 s_1을 보내고 pk^*의 소유자인 정당한 사용자와 같은 서명 단계를 진행한다.

위조 공격자 F가 $\perp$를 출력한 경우, A도 $\perp$를 출력한다. 그렇지 않은 경우 위조 공격자가 $pk^* \in pkl$, $\sigma = (s, h)$인 위조 서명 (σ, M, pkl)을 출력한다. 다음으로 A는 $pk_i = (y_i = (y_{i,1}, ..., y_{i,l}))$에 대하여 $pkl = (pk_1 = pk^*, pk_2, ..., pk_n)$로 정의하고 위조 서명의 유효성을 다음과 같이 확인한다. 먼저 A는 각각의 $i \in (1, ..., n)$에 대해 $T_a(pkl \| pk_i)$가 정의된 내부 $H_a(pkl \| pk^*)\, query$를 실행하고, $\pi_i = T_a(pkl \| pk_i)$를 설정한다. 이후 모든 $j \in (1, ..., l)$에 대해

$ay_j = \sum_{i=1}^{n} y_{i,\pi_i(j)}$식을 통해 $ay = (ay_1,...,ay_l)$값을 계산한다. 이 후 내부 $H_2(ay\|h\|M)$ $query$를 통해 $I = (idx_1\|...\|idx_k)$에 대해 $I = T_2(ay\|h\|M)$를 설정하고 $av := GCK_A(s) - \sum_{j=1}^{k} ay_{idx_j}$를 계산한다. $\hat{h} = H_1(av)$를 계산한 후 만약 $h \neq \hat{h}$라면, A는 $\perp$를 출력한다. 그렇지 않은 경우 A는 다음과 같은 추가정보와 함께 위조 서명을 출력한다.

A는 $T_a(pkl\|pk_i) = \pi_i$에 대해 $i_0 := i$를 정하고, $I = I_i$에 대해 $i_1 := i$를 정하여 $\pi = (\pi_1,...,\pi_n)$에 대해 $(i_0,i_1,pkl,ay,av,s,\pi,I)$를 출력한다.

이 다음 과정을 통해 알고리즘 A를 활용하여 알고리즘 B를 설계한다.

1) 알고리즘 B는 y^*를 입력값으로 받아 $l-1$개의 $x_i \xleftarrow{\$} R^{\mu}$을 랜덤하게 뽑는다. 이후 $b \in \{1,...,l\}$인 b에 대해 $y^* = y_b$ 포함한 $pk^* = (y_1,...,y_l)$를 설정하고 모든 $i \in \{1,...,l\}\backslash\{b\}$에 대해 $y_i = GCK_A(x_i)$를 계산한다.

2-1) 이 후 B는 $Fork^A$를 실행한다. 이는 B가 랜덤 코인인 ρ_A를 고른 후 공개키 pk^*와 $\pi_1,...,\pi_q$, $I_1,...,I_q$를 랜덤하게 골라 알고리즘 A를 실행하는 것이다. 만약 A가 $\perp$를 출력하면, B도 $\perp$를 출력하고, 그렇지 않다면 A는 $pkl = \{pk_1,...,pk_n\}$, $ay = (ay_1,...,ay_l)$, $\pi = (\pi_1,...,\pi_q)$에 대해 출력값으로 $(i_0,i_1,pkl,ay,av,s,\pi,I)$를 출력한다.

2-2) B는 같은 랜덤 코인 ρ_A에 대해 다시 알고리즘 A를 실행한다. 이 때 입력값으로는 공개키 pk^*와 $\pi_1,...,\pi_q$, $I_1,...,I_{i_1-1}$를 그대로 사용하고, $I'_{i_1},...,I'_q$는 랜덤하게 고른다. 만약 A가 $\perp$를 출력하면, B도 $\perp$를 출력하고, 그렇지 않다면 출력값으로 $(i'_0,i'_1,pkl',ay',av',s',\pi',I')$를 출력한다.

2-3) $i_0 = i'_0, i_1 = i'_1, I_{i_1} \neq I'_{i_1}$이 성립하는지 확인 후, 성립하지 않으면 B는 $\perp$를 출력한다. 성립할 경우 B는 같은 랜덤 코인 ρ_A에 대해 다시 알고리즘 A를 실행한다. 이 때 입력값으로는 공개키 pk^*와 $\pi_1,...,\pi_{i_0-1}$, $I_1,...,I_q$를 그대로 사용하고, $\pi'_{i_0},...,\pi'_q$는 랜덤하게 고른다. 만약 A가 $\perp$를 출력하면, B도 $\perp$를 출력하고, 그렇지 않다면 출력값으로 $(i''_0,i''_1,pkl'',ay'',av'',s'',\pi'',I'')$를 출력한다.

2-4) $i_0 = i''_0, i_1 = i''_1$이 성립하는지 확인 후, 성립하지 않으면 B는 $\perp$를 출력한다. 성립할 경우 B는 같은 랜덤 코인 ρ_A에 대해 다시 알고리즘 A를 실행한다. 이 때 입력값으로는 공개키 pk^*와 $\pi_1,...,\pi_{i_0-1}$, $\pi'_{i_0},...,\pi'_q$, $I_1,...,I_{i_1-1}$, $I'_{i_1},...,I'_q$를 사용한다. 만약 A가 $\perp$를 출력하면, B도 $\perp$를 출력한다. 그렇지 않은 경우에는 출력값으로 다음과 같이 $(i'''_0,i'''_1,pkl''',ay''',av''',s''',\pi''',I''')$를 출력한다.

위의 식을 이용해 다음의 식들을 정의할 수 있다.

$ay_i = y_{1,\pi_1(i)} + \sum_{j=2}^{n} y_{j,\pi_j(i)}$에 대해 다음식을 만족한다.

$$ay = ay' = (ay_1,...,ay_l)$$

$ay''_i = y_{1,\pi''_1(i)} + \sum_{j=2}^{n} y_{j,\pi_j(i)}$에 대해 다음식을 만족한다.

$$ay'' = ay''' = (ay''_1,...,ay''_l)$$

이후 B는 다음을 만족하는 $s^* := \tilde{s} - \tilde{s}'$를 계산한다.

$$GCK_A(s^*) = GCK_A(\tilde{s} - \tilde{s}')$$

$$= \sum_{i=1}^{k}(ay_{idx_i} - ay'_{idx'_i}) - \sum_{i=1}^{k}(ay''_{idx_i} - ay'''_{idx'_i})$$

$$= \sum_{i=1}^{k}(y_{1,\pi_1(idx_i)} + \sum_{j=2}^{n} y_{j,\pi_j(idx_i)} - (y_{1,\pi_1(idx'_i)} + \sum_{j=2}^{n} y_{j,\pi_j(idx'_i)}))$$

$$- \sum_{i=1}^{k}(y_{1,\pi''_1(idx_i)} + \sum_{j=2}^{n} y_{j,\pi_j(idx_i)} - (y_{1,\pi''_1(idx'_i)} + \sum_{j=2}^{n} y_{j,\pi_j(idx'_i)}))$$

$$= \sum_{i=1}^{k}(y_{1,\pi_1(idx_i)} - y_{1,\pi_1(idx'_i)} - (y_{1,\pi''_1(idx_i)} - y_{1,\pi''_1(idx'_i)}))$$

다음으로 $i \in \{1,...,k\}$에 대해 다음과 같이 정의한다. $IS_1 := \{\pi_1(idx_i)\}$, $IS_2 := \{\pi_1(idx'_i)\}$, 그리고 $IS_3 := \{\pi''_1(idx_i)\}$, $IS_4 := \{\pi''_1(idx'_i)\}$. 이후 공개키 $pk^* = pk_1 = (y_{1,1},...,y_{1,l})$에 대해 $y_{1,i} = y^*$이므로 B는 $i \in \{1,...,l\}\backslash\{b\}$에 대해 $y_{1,i}$의 모든 역함수 값을 알고 있다. 따라서 만약 b에 대해서 $b \in (IS_1 \cup IS_4)\backslash(IS_2 \cup IS_3)$라고 한다면, B는 GCK_A함수의 동형성질(homomorphic property)에 의해 다음식에 의해 $y_{1,b} = y^*$의 역함수도 알아낼 수 있다.

$$y^* = GCK_A(s^*)$$

$$- \sum_{\substack{i=1 \\ \pi_1(idx_i) \neq b \\ \pi''_1(idx'_i) \neq b}}^{k} (y_{1,\pi_1(idx_i)} - y_{1,\pi_1(idx'_i)} - y_{1,\pi''_1(idx_i)} + y_{1,\pi''_1(idx'_i)})$$

$$= GCK_A$$

$$(s^* - \sum_{\substack{i=1 \\ \pi_1(idx_i) \neq b \\ \pi''_1(idx'_i) \neq b}}^{k} (x_{1,\pi_1(idx_i)} - x_{1,\pi_1(idx'_i)} - x_{1,\pi''_1(idx_i)} + x_{1,\pi''_1(idx'_i)})$$

□

V. 결 론

본 논문에서는 양자 컴퓨팅 환경에서 안전한 서명 기법인 TACHYON 기법을 이용하여 다중 서명 기법을 설계하고 TACHYON 기법에서 안전성 증명에 쓰인 문제와 같은 문제인 GCK 함수의 일방향성에 근거하여 안전성을 증명하였다. 따라서 논문에서 제시한 TACHYON 기반의 다중 서명 기법 역시 TACHYON과 같이 양자 컴퓨팅 환경에서 안전한 기법으로, 최근 연구[4]에서 제시된 일반 공개키 모델에서 공개키 결합이 가능한 Schnorr 기반 기법의 설계 원리를 참고하여 설계한 일반 공개키 모델에서 공개키 결합이 가능한 기법이다. 최근 연구된 양자컴퓨팅 환경에서 안전한 다중 서명 기법은 다음 표와 같이 정리할 수 있다.

표에서 볼 수 있듯이 최근 연구된 양자컴퓨팅 환경에서 안전한 다중서명 기법들[13,9,14,5] 중 일반 공개키 모델과 키 결합 특성을 동시에 만족하는 기법으로, 검증 시 결합된 공개키 하나만으로 서명 검증이 가능할 뿐 아니라 공개키에 대응하는 개인키의 영지식 증명을 부가적으로 수행하지 않아도 안전한 장점을 모두 갖고 있다. 또한 최근 연구된 다른 기법들은 격자기반 난제의 어려움에 기반하고 있는데, TACHYON 논문[12]의 분석에 따르면 GCK 함수 기반의 TACHYON 서명이 격자기반 난제에 기반한 서명인 Dilithium, qTESLA[7,10] 등에 비해 서명 생성 및 검증 시간이 짧고 개인키의 크기가 작다는 장점이 있다는 것을 알 수 있다. 따라서 논문에서 제시한 기법 또한 같은 문제인 GCK 함수의 일방향성에 기반한 다중서명으로, 특정 환경에서 최근 연구된 양자컴퓨팅 환경에서 안전한 다중 서명 기법에 비해 효율적으로 사용될 수 있다. 향후 연구로는 현재 다른 기법에 비해 크기가 큰 공개키 및 서명의 크기를 줄인 GCK 함수 기반의 다중 서명에 대한 연구가 진행될 수 있을 것이다.

Table 1. Properties of post-quantum multi-signature schemes

	Hard problems	Plain public key model	Key aggregation
BS16	Ring-SIS	X	O
FH19	Ring-LWE	O	X
LTT20	SIS	X	O
DOTT21	LWE, SIS	O	X
Ours	GCK	O	O

References

[1] D. Boneh, M. Drijvers, and G. Neven, "Compact multi-signatures for smaller blockchains," International Conference on the Theory and Application of Cryptology and Information Security, pp. 435 - 464, Springer, Cham, Dec. 2018.

[2] D. Micciancio, "Generalized compact knapsacks, cyclic lattices, and efficient one-way functions," computational complexity, vol. 16, no. 4, pp. 365 - 411, Dec. 2007.

[3] D. Pointcheval and J. Stern, "Security arguments for digital signatures and blind signatures," Journal of Cryptology, vol. 13, no. 3, pp. 361-396, Dec. 2000

[4] G. Maxwell, A. Poelstra, Y. Seurin, and P. Wuille., "Simple schnorr multi-signatures with applications to bitcoin," Designs, Codes and Cryptography, vol. 87, no. 4, pp. 2139-2164, Feb. 2019.

[5] I. Damg˚ard, C. Orlandi, A. Takahashi and M. Tibouchi. "Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices," Cryptology ePrint Archive, Report 2020/1110, Sep. 2020.

[6] I. Dinur and N. Nadler, "Multi-Target Attacks on the Picnic Signature Scheme

and Related Protocols," Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 699-727, Springer, Cham, May. 2019.

[7] L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé, "CRYSTALS - Dilithium: A Lattice-Based Digital Signatures Scheme," IACR Transactions on Cryptographic Hardware and Embedded Systems, vol. 2018, no. 1, pp. 238-268, Feb. 2018.

[8] M. Bellare and G. Neven., "Multi-signatures in the plain public-key model and a general forking lemma," A Proceedings of the 13th ACM conference on Computer and communications security, pp. 390-399, Oct. 2006.

[9] M. Fukumitsu and S. Hasegawa. "A tightly-secure lattice-based multisignature," Proceedings of the 6th on ASIA Public-Key Cryptography Workshop, pp. 3-11, Jul. 2019.

[10] N. Bindel, S. Akleylek, E. Alkim, P. S. L. M. Barreto, J. Buchmann, E. Eaton, G. Gutoski, J. Kramer, P. Longa, H. Polat, J. E. Ricardini, and G. Zanon, "Submission to NIST's post-quantum project: Lattice-based digital signature scheme qTESLA," 2018.

[11] P.W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," Proceedings 35th annual symposium on foundations of computer science, pp. 124-134, Ieee, Nov. 1994.

[12] R. Behnia, M.O. Ozmen, A.A. Yavuz, and M. Rosulek, "TACHYON: fast signatures from compact knapsack," Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pp. 1855-1867, Oct. 2018.

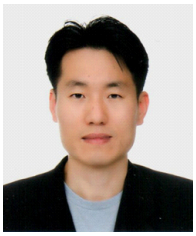
[13] R. E. Bansarkhani and J. Sturm. "An efficient lattice-based multisignature scheme with applications to bitcoins." International Conference on Cryptology and Network Security, pp. 140 - 155, Springer, Cham, Nov. 2016.

[14] Z.Y. Liu, Y.F. Tseng, and R. Tso. "Cryptanalysis of a round optimal lattice-based multisignature scheme." Cryptology ePrint Archive, Report 2020/1172, Sep. 2020.

〈저 자 소 개〉

고 찬 영 (Chanyoung Ko) 학생회원
2017년 2월: 아주대학교 수학과 졸업
2020년 9월~현재: 고려대학교 정보보호학과 석사과정
〈관심분야〉 암호 프로토콜, 공개키 암호

이 영 경 (Youngkyung Lee) 학생회원
2014년 2월: 고려대학교 수학과 졸업
2016년 2월: 고려대학교 정보보호학과 석사
2016년 3월~현재: 고려대학교 정보보호학과 박사과정
〈관심분야〉 암호 프로토콜, 암호이론, 인증 및 키 교환



이 광 수 (Kwangsu Lee) 종신회원
1998년 2월: 연세대학교 컴퓨터과학과 졸업
2000년 2월: 한국과학기술원 전산학과 석사
2011년 2월: 고려대학교 정보보호학과 박사
2016년 9월~2019년 8월: 세종대학교 정보보호학과 조교수
2019년 9월~현재: 세종대학교 정보보호학과 부교수
〈관심분야〉 암호 프로토콜, 공개키 암호

박 종 환 (Jong Hwan Park) 정회원
1999년 2월: 고려대학교 수학과 졸업
2005년 2월: 고려대학교 정보보호학과 석사
2008년 8월: 고려대학교 정보보호학과 박사
2013년 9월~2019년 8월: 상명대학교 컴퓨터과학과 조교수
2019년 9월~현재: 상명대학교 컴퓨터과학과 부교수
〈관심분야〉 함수 암호, 브로드캐스트 암호, 암호 프로토콜