2024 DOI: 10.1109/TIFS.2024.3409070

PkT-SIN: A Secure Communication Protocol for Space Information Networks with Periodic k-Time Anonymous Authentication

Yang Yang, Wenyi Xue, Jianfei Sun*, Guomin Yang, Yingjiu Li, HweeHua Pang, Robert H. Deng, *IEEE Fellow*

***Abstract*— Space Information Network (SIN) enables universal Internet connectivity for any object, even in remote and extreme environments where deploying a cellular network is difficult. Access authentication is crucial for ensuring user access control in SIN and preventing unauthorized entities from gaining access to network services. However, due to the complex communication environment in SIN, including exposed links and higher signal delay, designing a secure and efficient authentication scheme presents a significant challenge. In this paper, we propose a secure communication protocol for SIN with periodic k-time anonymous authentication (named PkT-SIN) that allows satellite users to anonymously authenticate to ground stations at most k times in each single time period. An efficient handover mechanism is designed to ensure seamless communication for satellite users to communicate with different satellites and ground stations, taking into account the dynamic topology of SIN. As a core component of PkT-SIN, we propose a novel primitive, periodic k-time keyed-verification anonymous credential (PkT-KVAC), that enables users to derive k tokens from a credential for anonymous and unlinkable authentication. On the other hand, a verifier can always recognize a reused token from a dishonest user. PkT-KVAC is of independent contribution to anonymous authentication in pay-per-use business scenarios. Formal security proofs confirm that PkT-SIN and PkT-KVAC have desired security features. The supremacy of their computing features is demonstrated through comprehensive comparison and rigorous performance analysis.**

***Index Terms*—space information networks, periodic k-time authentication, keyed-verification anonymous credential**

I. Introduction

SPACE information network (SIN) refers to a network architecture that amalgamates terrestrial communications with satellite systems [1]. In this architecture, a Low Earth Orbit satellite (LEO) takes on the task of relaying and enhancing ground signals, facilitating the connection of users to distant ground stations. The global market size for satellite communication reached a valuation of USD 71.6 billion in 2021, as reported in [2]. SIN systems offer several advantages, including an extensive coverage, a minimal fading margin, and the ability to surmount geographic constraints, enabling services in diverse environments such as oceans, airspace, forests, and deserts. SIN finds widespread deployment in fields like ship and aircraft navigation, long-distance telephone transmission, and real-time weather forecasting.

To prevent unauthorized service access, dependable authentication is crucial for SIN. Because sensitive personal information and service data are transmitted over satellite-ground links, anonymous authentication is desired in SIN to address the concern about potential misuse of personal data. The concept of anonymous authentication in SIN is explored in-depth in [3–7].

The foremost challenge to address in anonymous authentication in SIN is the propagation delay of authentication. Low Earth Orbit (LEO) satellites orbit the Earth at altitudes ranging from 500 to 2000 km, resulting in propagation delays of approximately 10 to 40 ms, as reported in [3]. To mitigate authentication time delays, several existing schemes, as outlined in [3, 5–7], empower LEO satellites with independent authentication capabilities. This enables user authentication without the need for ground entity involvement. However, many of these solutions, detailed in [3, 5–7], necessitate LEO satellites to perform intricate computations, such as resource-intensive pairing operations. Given the challenges associated with satellite launch and maintenance in space, LEO satellites are significantly constrained in terms of computing resources and are unsuitable for heavy computational tasks.

The approach of Keyed-Verification Anonymous Credential (KVAC) can be leveraged to mitigate the authentication propagation delay within SIN. KVAC was initially proposed by Chase et al. [8, 9], with the primary aim of enhancing the efficiency of authentication schemes. In KVAC, the issuer of anonymous credentials and the verifier of anonymous credentials share the same secret key. This shared secret key allows for a more efficient design of anonymous credentials, alleviating the need for resource-intensive bilinear pairing computations. Consequently, the computation efficiency of KVAC surpasses that of traditional anonymous credentials, as exemplified in [10–12].

Given the considerable distance of LEO satellites (LEOs for short) from earth and sufficient protection of SIN infrastructure, it is reasonable to entrust LEOs with the issuer's secret key for performing KVAC verification without significant concern on LEOs' compromise. We aim to significantly reduce not only authentication propagation delay in SIN but also the computation overhead on LEOs for achieving anonymous

Y. Yang is with the College of Computer Science and Big Data, Fuzhou University, Fuzhou, China, 350116; and School of Computing and Information Systems, Singapore Management University, Singapore 188065. (email: yang.yang.research@gmail.com), W. Xue is with the College of Computer Science and Big Data, Fuzhou University, Fuzhou, China, 350116. (email: wenyixue.research@gmail.com), J. Sun is with the School of Computer Science and Engineering, Nanyang Technological University, Singapore; (email: sjf215.uestc@gmail.com). G. Yang, H. Pang and R. H. Deng are with the School of Computing and Information Systems, Singapore Management University, Singapore 188065. (email: {gmyang, hhpang, robertdeng}@smu.edu.sg) Y. Li is with the Department of Computer Science, University of Oregon. (email: yingjiul@uoregon.edu)

*Corresponding author: Jianfei Sun.

authentication in this setting.

We face several significant challenges in achieving our goal. First, since each LEO satellite travels at a speed of approximately 7.5 km/s, it can only provide services to each user with an effective service duration ranging from 5 to 15 minutes [13]. To ensure seamless real-time service continuity, anonymous authentication may be performed with different LEOs over time in each service session. Our challenge is to make the handover of authentication between multiple LEOs highly efficient and privacy-preserving.

Second, in scenarios where the number of authentication attempts is not limited, dishonest users may freely distribute their anonymous credentials to many unauthorized users. This would not only damage the profitability of SIN services but also open a door to more potential attacks. Our challenge for mitigating this problem is to restrict the number of times each satellite user can authenticate to LEOs within any defined period. Third, malicious users may launch Distributed Denial of Service (DDoS) attacks. A notable example is the large-scale DDoS attack targeted at a prominent American satellite broadband services provider, Viasat, in February 2022. This attack resulted in a disruption of satellite broadband service with tens of thousands of users affected in Europe [14]. Users performing anonymous authentication in SIN may leverage their anonymity in performing such DDoS attacks. Our challenge for mitigating this threat is to enable LEOs to identify suspicious users if their authentication attempts exceed a specified limit within any defined period.

A. Contributions

Addressing these identified challenges, we develop a novel secure communication protocol for SINs. The cornerstone of our protocol is a novel anonymous credential primitive, termed Periodic k-Time Keyed-Verification Anonymous Credential (PkT-KVAC). PkT-KVAC is a new type of KVAC that empowers users to generate a maximum of k valid authentication tokens within any defined time period. Users who exceed this authentication limit within any time period can be promptly identified. The number of authentication times in PkT-KVAC is automatically reset to k in each time period, eliminating the need for users to interact with their credential issuer. In line with the efficiency advantage of KVAC, our PkT-KVAC exhibits low computation costs.

Subsequently, we propose PkT-SIN, a secure communication protocol for SIN featuring periodic k-time keyed verification authentication. PkT-SIN empowers LEO to play a pivotal role in authentication token verification and utilizes PkT-KVAC for anonymous authentication. The former design feature effectively reduces interactions and communication delays, and the later considerably alleviates the computation costs on LEOs. With the property of periodic k-time authentication, PkT-KVAC can effectively deter users from sharing their anonymous credentials and launching DDoS attacks in SINs. In addition, PkT-SIN incorporates a secure session establishment protocol and achieves an array of security properties.

Our contributions are summarized as follows:

• **Novel Anonymous Credential Primitive**. We introduce PkT-KVAC, a novel keyed-verification anonymous credential primitive that enables periodic k-time authentication. While preserving the efficiency of KVAC, PkT-KVAC restricts the number of authentication instances within each time period. We formally define the syntax and security model of PkT-KVAC and provide an efficient construction without the need for pairing operations.

• **Anonymous Authentication and Secure Session Establishment**. By incorporating PkT-KVAC as a cornerstone, We propose PkT-SIN, a new secure communication protocol for SINs. While enabling anonymous and unlinkable user authentication, PkT-SIN maintains the benefit of low computation costs inherited from PkT-KVAC, rendering it well-suited for implementation in LEOs. In response to the frequent handover of SINs, we have developed distinct secure session establishment protocols for three scenarios, ensuring forward and backward secrecy. PkT-SIN achieves an array of desired security properties, including mutual authentication, anonymity, unlinkability, accountability, and resistance to replay/impersonation/man-in-the-middle/DDoS attacks.

• **Periodic k-Time Anonymous Authentication**. PkT-SIN supports periodic k-time anonymous authentication, granting valid users to be authenticated anonymously at most k times within each designated time period. This function serves to defer unauthorized credential sharing and resist DDoS attacks. The number of authentication times is automatically reset to k in each time period, which avoids redundant interactions and delays caused by users reapplying access permissions. Without involving any trusted third party, PkT-SIN facilitates rapid detection and identification of users who perform anonymous authentication more than k times.

• **Seamless Handover**. Accommodating the movements of satellites and mobile users, PkT-SIN incorporates secure handover protocols including: (1) LEO handover for mobile users at low speeds, (2) ground station handover for mobile users at high speeds, and (3) handover of both LEOs and ground stations for mobile users at high speeds. We streamlined handover interactions based on the movements of entities to eliminate redundant communication delays without compromising security. The performances of the proposed handover protocols are rigorously evaluated in comparative studies.

B. Related Works

Secure Satellite Communication. A list of protocols had been proposed for secure satellite communications [15–27]. In particular, Cruickshank et al. [15] introduced a secure commercial satellites communication system based on public key cryptography and digital signatures. However, this scheme had high computational costs and limited efficiency. Hwang et al. [16] proposed a hash-based authentication protocol for LEOs, which improves communication delay and data capacity. Many recent works [17–20] enhanced Hwang's research to be secure against various attacks. In another work, Dharminder et al. [21] proposed a post-quantum secure key exchange and authentication protocol for satellite communication, aiming to withstand quantum computing-based attacks. Several other works [22–25] focused on optimizing power allocation for multi-beam satellite communication while ensuring physical layer security. In addition, software-defined networks [23, 26] were introduced to improve flexibility in resource and service utilization for secure satellite communication. A moving target defense strategy was introduced to mitigate Distributed Denial of Service (DDoS) attacks to satellite communications [27]. However, no protocol mentioned above [15–27] supports anonymity or unlinkability, leaving sensitive personal information vulnerable to potential exposure to LEO satellites and ground stations.

In the context of secure satellite communication, Xue et al. [4] presented an efficient handover protocol for anonymous batch verification, but it failed to achieve handover unlinkability. Yang et al. [3] proposed an anonymous authentication protocol for SIN using group signatures, achieving unlinkable LEO handovers but with increased satellite computation costs. Liu et al. [5] utilized blockchain technology for decentralized authentication and fair billing in cross-company satellite services. Their subsequent work [6, 7] introduced distributed systems, collaborative credential issuing algorithms, and incentive mechanisms. Wazid et al. [38] and Wang et al. [39] leveraged Hash and MAC operations to enable lightweight and secure authentication in the context of Internet of Space Things (IoST) and space-ground integrated railway networks, respectively. By reducing the computational burden on resource-constrained LEO devices, these schemes demonstrated practicality in the targeted application domains. However, authentication schemes [4, 38, 39] require verifiers to be aware of users' identities, thus failing to provide anonymity and unlinkability for NCC and LEO. Ma et al. [40] proposed a lattice-based authentication scheme for SIN and designed an aggregated signature mechanism for large-scale IoT device authentication. Another post-quantum authentication protocol proposed by Guo et al. [41] was constructed based on the ring learning with errors (RLWE) problem, exhibiting more compact ciphertexts and lower computation overheads. Despite these advancements, similar challenges of unauthorized credential sharing, DDoS attacks and heavy computation overheads remain unaddressed in the related works.

***k*-Time Anonymous Authentication**. k-Time Anonymous Authentication (k-TAA) permits users to be authenticated at most k times while maintaining anonymity. Teranishi et al. [28] introduced the first k-TAA scheme, allowing identification of users exceeding the authentication limit. Nguyen et al. [29] extended this concept to dynamic k-TAA, enabling each authenticator to independently grant or revoke access permissions. Chaterjee et al. [30] proposed a k-TAA scheme based on physically unclonable functions, suitable for trusted platform modules (TPM). In addition, Huang et al. [31] devised an efficient k-TAA system tailored for pay-as-you-go pricing, facilitating multiple service accesses and associated payments within each single authentication period. However, all these schemes [28–31], require users to reapply for access permissions when their authentication tokens are depleted. Therefore, they are not ideal for scenarios in which frequent communication connections and authentications are required as in SIN.

Many existing k-TAA schemes [28–31] lack the capability of offering periodic anonymous authentication. Although the concept of periodic k-TAA has been realized in [32, 33], they fall short in supporting selective disclosure of credential attributes for achieving fine-grained authentication. Furthermore, they entail a substantial number of pairing operations, resulting in significant verification latency. In comparison, our solution supports periodic fine-grained anonymous authentication while alleviating the need for cumbersome pairing operations.

II. Preliminaries

A. Group Description and Hardness Assumptions

A group generator $\mathsf{GGen}(1^\lambda) \to (\mathbb{G}, p)$ inputs a security parameter 1^λ and outputs a cyclic group $\mathbb{G}$ of prime order p. We assume that the collision-resistant hash functions $H_1 : \{0,1\}^* \to \mathbb{Z}_p^*$ and $H_2 : \{0,1\}^* \to \mathbb{G}$ can be efficiently computed in $\mathbb{G}$. For security purposes, our scheme requires the following assumptions to hold in a group $\mathbb{G}$.

Definition II.1. *(Discrete Logarithm (DL) Assumption). Let g be a generator of $\mathbb{G}$. Given a tuple $(g, g^a) \in \mathbb{G}^2$, output $a \in \mathbb{Z}_p^*$. DL assumption holds if for all PPT adversary $\mathcal{A}$, the advantage function $\mathsf{Adv}_{\mathcal{A}}^{DDH}(\lambda) := Pr[\mathcal{A}(g, g^a) = a]$ is negligible.*

Definition II.2. *(Decisional Diffie-Hellman (DDH) Assumption). Let g be a generator of $\mathbb{G}$. The input is a tuple $\mathcal{T} = (g, g^a, g^b) \in \mathbb{G}^3$, where $a, b \xleftarrow{\$} \mathbb{Z}_p^*$. DDH assumption holds if for all PPT adversary $\mathcal{A}$, the advantage $\mathsf{Adv}_{\mathcal{A}}^{DDH}(\lambda) := |Pr[\mathcal{A}(\mathcal{T}, g^{ab}) = 1] - Pr[\mathcal{A}(\mathcal{T}, g^c) = 1]|$ is negligible, where $c \xleftarrow{\$} \mathbb{Z}_p^*$.*

Definition II.3. *(LRSW Assumption). Let g be a generator of $\mathbb{G}$. For $\widetilde{A} = g^a$ and $\widetilde{B} = g^b$ with random scalar $a, b \xleftarrow{\$} \mathbb{Z}_p^*$, a LRSW oracle $\mathcal{O}(m)$ is defined as follows: take $m \in \mathbb{Z}_p^*$ as input, $\mathcal{O}(m)$ chooses a random $h \in \mathbb{G}$ and outputs a tuple $\mathcal{T} = (h, h^b, h^{a+mab})$. LRSW assumption holds if for all PPT adversary $\mathcal{A}$, the advantage $\mathsf{Adv}_{\mathcal{A}}^{LRSW}(\lambda) := Pr[\mathcal{A}(\widetilde{A}, \widetilde{B}, \mathcal{O}(\cdot)) = (\widetilde{h}, \widetilde{h}^b, \widetilde{h}^{a+\widetilde{m}ab})]$ is negligible, where $\widetilde{h} \xleftarrow{\$} \mathbb{G}$ and $\widetilde{m}$ is not asked to $\mathcal{O}(\cdot)$.*

B. Keyed-Verification Anonymous Credentials (KVAC)

The formal definition of KVAC [8] is formalized as follows.

• $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$. Take the security parameter 1^λ as input, this algorithm outputs the system public parameters pp, which is an implicit input of the following algorithms.

• $\mathsf{IKeyGen}(\mathsf{pp}) \to (\mathsf{isk}, \mathsf{ipar})$. Take pp as input, this algorithm outputs the issuer's secret key isk and public parameters ipar.

• $\mathsf{Issue}(\mathcal{I}(\mathsf{isk}, S) \leftrightarrow \mathcal{U}(\mathsf{ipar}, \mathsf{ATTR})) \to \mathsf{cred}$ is an interactive protocol where a user $\mathcal{U}$ can obtain a credential cred on attributes ATTR from an issuer $\mathcal{I}$.

• $\mathsf{Show}(\mathsf{ipar}, \mathsf{cred}, \mathsf{ATTR}, \phi) \leftrightarrow \mathsf{Verify}(\mathsf{isk}, \phi)$ is an interactive protocol between a user and a verifier. Show is executed by a user to generate a proof of possession π of a credential cred certifying some set of ATTR satisfying statement ϕ under the key corresponding to ipar, and Verify is executed by the verifier in possession of isk to verify the proof π claiming knowledge of a credential satisfying the statement ϕ.

C. Signature of Knowledge (SoK)

Signature of Knowledge (SoK) [34] is a special signature technique that enables the signer to sign messages in a zero-knowledge way. To be specific, SoK convinces the verifier that the signer knows the knowledge of a specific witness without revealing other irrelevant information. Compared with traditional digital signatures, the verification of a SoK does not require the signer's public key, and is suitable for anonymous scenarios. Let notation $SoK\{a : h = g^a\}(m)$ denotes a signature on message m, where the signer knows the secret a of witness $h = g^a$. An instantiation of the above SoK can be computed as follows: (1) the signer selects $\widetilde{a} \overset{\$}{\leftarrow} \mathbb{Z}_p^*$, computes $\widetilde{h} = g^{\widetilde{a}}$, $c = H(h||\widetilde{h}||m)$, $\overline{a} = \widetilde{a} - c \cdot a$ and returns $\pi = (\widetilde{h}, \overline{a}, c)$, where $H : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$ is a hash function. (2) The verifier examines $c \overset{?}{=} H(h||\widetilde{h}||m)$ and $\widetilde{h} \overset{?}{=} h^c g^{\overline{a}}$ and returns 1 if the above equations hold.

III. System and Threat Model of PkT-SIN

A. System Architecture

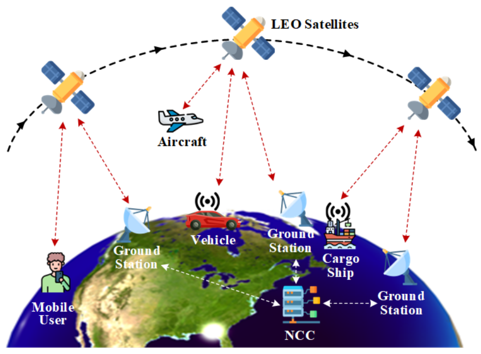


Fig. 1: System Architecture

The system architecture of PkT-SIN is presented in Fig. 1, which consists of a network control center (NCC), low earth orbit satellites (LEOs), ground stations (GSs) and satellite users (SUs). The functions and duties of each entity are illustrated below. The notations used in our system are listed in Table I.

Network control center (NCC) is responsible for setting up the SIN system, generating secret/public key pairs for LEOs and GSs, and providing registration service for SUs.

Low earth orbit satellites (LEOs) distributed in space are the access point for SUs. LEOs are tasked to authenticate SUs to prevent unauthorized service access, establish secure sessions between SU and GS, and forward messages for them. LEOs also provide seamless session handover service to ensure stable network connection for SUs.

Ground station (GS) distributed on Earth provides terrestrial network access for LEOs and SUs. It connects to NCC and provides a ground interface for LEOs.

Satellite user (SU) can be a mobile user, ship, vehicle and aircraft. To join the network, SUs first register with NCC to subscribe to network access services and then authenticate to LEOs to obtain network service.

TABLE I: Summary of Notations

Notation	Description
SIN	space information network
NCC	network control center
LEO	low earth orbit satellite
GS	ground station
SU	satellite user
$\mathcal{L}_{user}$	user information list
$\mathcal{L}_{tok}$	authentication token list
TP	time period
TIN	unique token identifier number
J_u	a counter for remaining authentication times
$EK_{L,U}$	symmetric key shared between LEO and SU
$EK_{L,G}$	symmetric key shared between LEO and GS
$\mathsf{cred}_{TP,k}$	anonymous credential with k-time authentication in TP
tok	authentication token
$x \overset{\$}{\leftarrow} \mathbb{Z}_p^*$	randomly choose an element x from $\mathbb{Z}_p^*$
$[1, k]$	integer set $\{1, 2, \cdots, k\}$

B. Security Model

The system depends on the following security assumptions.

NCC is an honest entity and trusted by the entities in the system, which cannot be compromised by any adversary. There exists secure channels between NCC and LEO, NCC and GS, NCC and SU, as well as LEO and GS. The secure channels can be established via TLS or SSL protocols.

LEOs and GSs are deemed semi-honest entities. Without loss of generality, they honestly provide network access service for the authorized SUs, but also attempt to collect users' real identities from the communications and explore the linkability of different sessions, which may breach users' privacy.

SUs may behave maliciously in the sense that they aim to access SIN services without valid credentials. They could forge credentials in order to pass the authentication. Authorized SUs may also forge extra tokens after k-time authentications have expired. SUs attempt to evade tracing in case of misbehavior.

We also assume polynomial-time adversaries, who can eavesdrop, tamper or interrupt the interactions among NCC, SUs, GSs and LEOs. They motivate to break the security of the proposed anonymous authentication scheme when the SUs access the network.

C. Security Requirements for PkT-SIN

The following security requirements should be satisfied in a PkT-SIN system.

• *Secure Key Establishment*. A secret session key should be established between GS and SU to protect the communications between them.

• *Mutual Authentication*. A GS should authenticate an SU to check their legitimacy, while an SU is able to authenticate a GS to prevent spoofing network services.

• *Key Forward/Backward Secrecy*. Since PkT-SIN provides handover mechanisms for three scenarios, key forward secrecy (KFS) and key backward secrecy (KBS) should be provided. KFS ensures that the knowledge of previous session keys is not helpful for the attacker to derive the future keys, which guarantees the privacy of future communications after the handover. KFS indicates that the compromise of a secure channel does not threaten the privacy of previous communications.

• *Anonymity*. User anonymity protects the real identity of SU from being leaked to GS or attackers.

• *Accountability*. The real identity of SU can be disclosed when SU is discovered to be a dishonest user that exceeds the time limit.

• *Unlinkability*. Unlinkability ensures that multiple network access service requests sent by the same SU cannot be linked, which protects the privacy of SU's position and trajectory.

• *Resistance to Replay Attacks*. Protect communications from malicious replaying of transmitted messages.

• *Resistance to Impersonation Attacks*. Prevent attackers from masquerading as legitimate users or entities within a PkT-SIN system.

• *Resistance to Man-in-the-Middle (MitM) Attacks*. Prevent an attacker from secretly intercepting and altering the communication between two parties who believe they are directly communicating with each other.

• *Resistance to Distributed Denial of Service (DDoS) Attacks*. Enable a system to withstand malicious attempts to overwhelm it with a flood of traffic, ensuring service availability.

IV. Periodic k-Time KVAC (PkT-KVAC)

A. Syntax definition

Periodic k-Time Keyed-Verification Anonymous Credential (PkT-KVAC) follows the definition of keyed-verification anonymous credentials (KVAC), where the credential issuer $\mathcal{I}$ is responsible to setup the system (algorithm Setup), issue anonymous credentials (algorithm Issue), validate authentication requests (algorithm Verify) and reveal dishonest users' identities (algorithm Reveal). The Issue algorithm outputs a credential cred to $\mathcal{U}$, which allows him to derive unlinkable tokens tok (algorithm Show) for anonymous authentication. Each token tok is bound to a unique token identifier number TIN, and the Show algorithm restricts $\mathcal{U}$ to have at most k TINs in each time period TP. If two tokens are found corresponding to the same TIN in a bounded period, the identity of token owner can be quickly revealed through algorithm Reveal.

• $\mathsf{Setup}(1^\lambda, 1^n) \to \mathsf{pp}$. Take the security parameter 1^λ and maximum number n of attributes in a credential as input, this algorithm outputs the system public parameters pp, which is an implicit input of the following algorithms.

• $\mathsf{IKeyGen}(\mathsf{pp}) \to (\mathsf{ipk}, \mathsf{isk})$. Take pp as input, this algorithm outputs the issuer's public key ipk and secret key isk.

• $\mathsf{UKeygen}(\mathsf{pp}, ID) \to (\mathsf{upk}, \mathsf{usk})$. Take pp and user's identity ID as input, this algorithm outputs the user's public key upk and secret key usk.

• $\mathsf{Issue}(\mathcal{U}(\mathsf{upk}, \mathsf{usk}, \mathsf{ATTR}) \leftrightarrow \mathcal{I}(\mathsf{isk}, TP, k)) \to (\mathsf{cred}_{TP,k}, \mathcal{D})$. $\mathcal{U}$ interacts with $\mathcal{I}$ to obtain an anonymous credential for k-time authentication in time period TP. $\mathcal{U}$ takes the user's secret/public key $\mathsf{usk}/\mathsf{upk}$ and attributes $\mathsf{ATTR} = \{\mathsf{attr}_i\}_{i\in[n]}$ as input; $\mathcal{I}$ takes the issuer secret key isk, time period TP and authentication time k as input. This algorithm outputs a credential $\mathsf{cred}_{TP,k}$ and a dispenser $\mathcal{D}$ to manage the authentication times.

• $\mathsf{Show}(\mathsf{upk}, \mathsf{usk}, \mathsf{cred}_{TP,k}, \mathsf{ATTR}_D, \mathcal{D}, M) \to (\mathsf{tok}, \mathsf{TIN}, \mathcal{D}')$. $\mathcal{U}$ takes the user public key upk, secret key usk, the anonymous credential $\mathsf{cred}_{TP,k}$, disclosed attribute subset $\mathsf{ATTR}_D \subseteq \mathsf{ATTR}$, a dispenser $\mathcal{D}$ and a message M as input. This algorithm outputs an authentication token tok corresponding to token identifier number TIN and an updated dispenser $\mathcal{D}'$.

• $\mathsf{Verify}(\mathsf{isk}, \mathsf{tok}, \mathsf{TIN}, \mathsf{ATTR}_D, M) \to 0/1$. $\mathcal{I}$ takes the issuer secret key isk, a token tok, a token series number TIN, the disclosed attributes ATTR_D and a message M as input. This algorithm outputs 1 to denote that $(\mathsf{TIN}, \mathsf{tok}, M)$ is valid, and 0 otherwise.

• $\mathsf{Reveal}(\mathsf{TIN}, \mathsf{tok}, \mathsf{tok}') \to (\mathsf{upk}, ID)$. If tok and tok' are found to be bound to the same TIN, $\mathcal{I}$ runs this algorithm to outputs a public key upk to reveal the identity ID of the dishonest user.

B. Security Requirements for PkT-KVAC

Here we present the security requirements for PkT-KVAC.

• *Correctness*. It requires that every anonymous credential generated by Issue for attribute set can be used to generate an authentication token for any statement satisfied by that attribute set.

• *Unforgeability*. An adversary should not be able to present an anonymous credential without having received one from the issuer first.

• *Anonymity*. The real identity of a user cannot be deduced from the authentication interactions if the user is honest and does not reuse authentication tokens with the same identifier number.

• *k-Detectability*. An adversary cannot generate more than k authentication tokens in one time period TP without being detected.

• *Exculpability*. An adversary cannot incriminate an honest user $\mathcal{U}$ for reusing an authentication token with token identifier number TIN_i.

• *Dishonest User Revealing*. The identity of dishonest user can be efficiently revealed when a reusing behavior (of authentication tokens) is discovered.

C. Construction

The construction of PkT-KVAC is described below. The SoKs in this scheme are instantiated in Supplemental Material B. The correctness proof of PkT-KVAC is shown in Supplemental Material C.1.

• $\mathsf{Setup}(1^\lambda, 1^n) \to \mathsf{pp}$. $\mathcal{I}$ generates a cyclic group $\mathbb{G}$ of prime order p, chooses a generator g of group $\mathbb{G}$, selects $\{g_i\}_{i\in[1,3]}, \{h_i\}_{i\in[1,3]}, \{u_i, \widetilde{u}_i\}_{i\in[0,n]} \xleftarrow{\$} \mathbb{G}$, and chooses hash function $H_1 : \{0,1\}^* \to \mathbb{Z}_p^*$. Set $g_0 = g$. Let l_k and l_{tp} be the bit length of k and TP such that $k \in [1, 2^{l_k} - 1]$, $TP \in [1, 2^{l_{tp}} - 1]$ and $2^{l_k + l_{tp} + 1} < p$. Function $f : \{0,1\} \times \{0,1\}^{l_{tp}} \times \{0,1\}^{l_k} \to \{0,1\}^{l_k + l_{tp} + 1}$ is defined as (see Fig. 2) : $f(a, b, c) = (a \cdot 2^{l_{tp}} + b) \cdot 2^{l_k} + c$. It outputs $\mathsf{pp} = (\mathbb{G}, p, \{g_i\}_{i\in[0,3]}, \{h_i\}_{i\in[1,3]}, \{u_i, \widetilde{u}_i\}_{i\in[0,n]}, H_1, f)$, which is an implicit input in the algorithms below.

• $\mathsf{IKeyGen}(\mathsf{pp}) \to (\mathsf{ipk}, \mathsf{isk})$. $\mathcal{I}$ selects $x_1, x_2, x_3, y_1, y_2, y_3, z_0, \cdots, z_n \xleftarrow{\$} \mathbb{Z}_p^*$, computes $X = g_1^{x_1} h_3^{y_3}$, $Y_1 = h_1^{y_1}$, $Y_2 =$

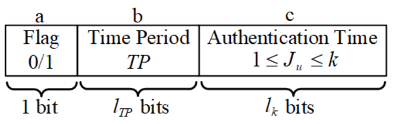


Fig. 2: Coding Format of TP and k in function f

$h_2^{y_2}$, $Z = g_0 / (g_2^{x_2} g_3^{x_3} \cdot (h_1 h_2)^{y_1 y_2 y_3} \cdot \prod_{i=0}^{n} u_i^{z_i})$, and outputs $\mathsf{ipk} = (X, Y_1, Y_2, Z)$ and $\mathsf{isk} = (\{x_i, y_i\}_{i \in [1,3]}, \{z_i\}_{i \in [0,n]})$.

• $\mathsf{UKeyGen}(\mathsf{pp}, ID) \rightarrow (\mathsf{upk}, \mathsf{usk})$. $\mathcal{U}$ selects $x_u \xleftarrow{\$} \mathbb{Z}_p^*$, computes $Y_u = g_0^{x_u}$, outputs $\mathsf{upk} = Y_u$ and $\mathsf{usk} = x_u$. $\mathcal{U}$ registers (ID, upk) to system manager, where ID is user's identity.

• $\mathsf{Issue}(\mathcal{U}(\mathsf{upk}, \mathsf{usk}, \mathsf{ATTR}) \leftrightarrow \mathcal{I}(\mathsf{isk}, TP, k)) \rightarrow (\mathsf{cred}_{TP,k}, \mathcal{D})$. The interaction of this algorithm consists of three steps.

(1) $\mathcal{U}$ selects random $s' \xleftarrow{\$} \mathbb{Z}_p^*$ to compute a commitment $\mathsf{Cm} = Y_1^{x_u} Y_2^{s'}$ and a signature of knowledge (SoK) $\Pi_U^1 = SoK\{(x_u, s') : Y_u = g_0^{x_u} \wedge \mathsf{Cm} = Y_1^{x_u} Y_2^{s'}\}(\mathsf{ATTR})$. $\mathcal{U}$ sends $(Y_u, \mathsf{Cm}, \mathsf{ATTR}, \Pi_U^1)$ to $\mathcal{I}$.

(2) If Π_U^1 is verified valid, $\mathcal{I}$ selects $s'', t \xleftarrow{\$} \mathbb{Z}_p^*$, $U \xleftarrow{\$} \mathbb{G}$, computes

$$V = g_1^{x_1} U^{x_2 + x_3 t} (\mathsf{Cm} \cdot Y_2^{s''})^{y_3} \cdot \widetilde{u}_0^{H_1(TP,k) z_0} \prod_{i=1}^{n} \widetilde{u}_i^{H_1(\mathsf{attr}_i) z_i},$$

$$\begin{aligned} \Pi_I = SoK\{(\{x_i, y_i\}_{i \in [1,3]}, \{z_i\}_{i \in [0,n]}) : X = g_1^{x_1} h_3^{y_3} \\ \wedge Z = g_0 / (g_2^{x_2} g_3^{x_3} (h_1 h_2)^{y_1 y_2 y_3} \prod_{i=0}^{n} u_i^{z_i}) \\ \wedge V = g_1^{x_1} U^{x_2} (U^t)^{x_3} (\mathsf{Cm} \cdot Y_2^{s''})^{y_3} \cdot \widetilde{u}_0^{H_1(TP,k) z_0} \\ \cdot \prod_{i=1}^{n} \widetilde{u}_i^{H_1(\mathsf{attr}_i) z_i}\}(s'', t, U, V, TP, k). \end{aligned}$$

Then, the tuple $(s'', t, U, V, TP, k, \Pi_I)$ is returned to $\mathcal{U}$.

(3) If Π_I is verified valid, $\mathcal{U}$ computes $s = s' + s''$ and stores the anonymous credential $\mathsf{cred}_{TP,k} = (s, t, U, V, TP, k)$. $\mathcal{U}$ initialize a dispenser $\mathcal{D} = \{1, 2, \cdots, k\}$.

• $\mathsf{Show}(\mathsf{upk}, \mathsf{usk}, \mathsf{cred}_{TP,k}, \mathsf{ATTR}_D, \mathcal{D}, M) \rightarrow (\mathsf{tok}, \mathsf{TIN}, \mathcal{D}')$. As shown in Algorithm 1, $\mathcal{U}$ checks whether the dispenser $\mathcal{D}$ is empty to ensure that there are remaining authentication times. This algorithm aborts when $\mathcal{D} = \emptyset$ (Line 1-2). Otherwise, $\mathcal{U}$ randomly selects an element J_u from $\mathcal{D}$ (Line 4) to compute $\alpha_0 = f(0, TP, J_u)$, $\alpha_1 = f(1, TP, J_u)$ (Line 5). As shown in Fig. 2, the elements α_0, α_1 consist of a flag bit, a l_{tp}-bits segment (carrying the information of TP) and a l_k-bits segment (carrying the information of J_u).

Next, SU randomizes the anonymous credential $\mathsf{cred}_{TP,k}$ to produce unlinkable elements $C_u = \{C_1, C_2, C_3, C_4\}$, $D_u = \{D_1, D_2\}$, $E_u = \{E_0, E_1, \cdots, E_n\}$ (line 6-14). Note that if an attribute attr_i does not need to be revealed, SU randomizes it by computing $E_i = u_i^r \widetilde{u}_i^{H_1(\mathsf{attr}_i)}$ (line 13-14); otherwise, it generates an independent element $E_i = u_i^r$ (line 11-12). To bind α_0, α_1 with an authentication token, SU computes $T_u = g_0^{1/(s+\alpha_0)}$ and $F_u = Y_u \cdot g_0^{w/(s+\alpha_1)}$, where T_u is set as the token identifier TIN, and F_u is utilized for tracing malicious users (line 15-16). Finally, SU computes a signature of knowledge Π_U^2 (line 18-21), and outputs $\mathsf{tok} = (J_u, C_u, D_u, E_u, F_u, \Pi_U^2, TP, k), \mathsf{TIN} = T_u$, $\mathcal{D}' = \mathcal{D} - J_u$ (line 22).

Algorithm 1: Show

```
   Input: upk, usk, $\mathsf{cred}_{TP,k}$, $\mathsf{ATTR}_D$, $\mathcal{D}$, $M$.
   Output: tok, TIN, $\mathcal{D}'$.
1  if $\mathcal{D} = \emptyset$ then
2      Abort;
3  else
4      Select $J_u$ from dispenser $\mathcal{D}$;
5      Set $\alpha_0 = f(0, TP, J_u)$, $\alpha_1 = f(1, TP, J_u)$;
6      Parse $\mathsf{cred}_{TP,k} = (s, t, U, V, TP, k)$;
7      Select $r \xleftarrow{\$} \mathbb{Z}_p^*$, compute:
8          $C_1 = g_0^r V$,  $C_2 = g_2^r U$,  $C_3 = g_3^r U^t$,  $C_4 = Z^r$;
9          $D_1 = Y_2^r \cdot h_1^{x_u}$,  $D_2 = Y_1^r \cdot h_2^s$,  $E_0 = u_0^r$;
10     for $attr_i \in ATTR$ do
11         if $attr_i \in ATTR_D$ then
12             Compute $E_i = u_i^r$;
13         else
14             Compute $E_i = u_i^r \widetilde{u}_i^{H_1(\mathsf{attr}_i)}$;
15     Set $w = H_1(C_u || D_u || E_u)$;
16     Compute $T_u = g_0^{1/(s+\alpha_0)}$, $F_u = Y_u \cdot g_0^{w/(s+\alpha_1)}$;
17     Set $\beta = 1/(s + \alpha_1)$, $\delta = -r \cdot \beta$;
18     Produce a signature of knowledge $\Pi_U^2$:
19         $\Pi_U^2 = SoK\{(x_u, s, t, r, w, \beta, \delta) : g_0 = T_u^{(s+\alpha_0)} \wedge$
20         $g_0 = (g_0^s h_1^r g_0^{\alpha_1})^\beta h_1^\delta \wedge D_1 D_2 C_4 = (Z Y_1 Y_2)^r \cdot h_1^{x_u} h_2^s$
21         $\wedge F_u = g_0^{x_u} \cdot (g_0^w)^\beta\}(\mathsf{ATTR}_D, TP, J_u, M)$;
22     Return tok $= (J_u, C_u, D_u, E_u, F_u, \Pi_U^2, TP, k)$, TIN $=$
         $T_u$, $\mathcal{D}' = \mathcal{D} - J_u$.
```

• $\mathsf{Verify}(\mathsf{isk}, \mathsf{tok}, \mathsf{TIN}, \mathsf{ATTR}_D, M) \rightarrow 0/1$. Parse $\mathsf{tok} = (J_u, C_u, D_u, E_u, F_u, \Pi_U^2)$, $\mathcal{I}$ first checks whether (a) TP is the identifier of current time period, (b) $1 \le J_u \le k$, (c) the signature of knowledge Π_U^2 is valid. Then, $\mathcal{I}$ calculates $\Gamma = (E_0 \cdot \widetilde{u}_0^{H_1(TP,k)})^{z_0} \cdot \prod_{\mathsf{attr}_i \in \mathsf{ATTR}_D} (E_i \cdot \widetilde{u}_i^{H_1(\mathsf{attr}_i)})^{z_i} \cdot \prod_{\mathsf{attr}_i \notin \mathsf{ATTR}_D} E_i^{z_i}$ and verifies whether the equation holds:

$$C_1 \overset{?}{=} g_1^{x_1} C_2^{x_2} C_3^{x_3} C_4 \cdot (D_1^{y_1 y_3} D_2^{y_2 y_3}) \cdot \Gamma.$$

If the equation holds, it outputs 1, otherwise it outputs 0.

• $\mathsf{Reveal}(\mathsf{TIN}, \mathsf{tok}, \mathsf{tok}') \rightarrow (\mathsf{upk}, ID)$. Note that the token series number $\mathsf{TIN} = g_0^{1/(s+\alpha_0)} = g_0^{1/(s+f(0,TP,J_u))}$ is determined by the secret s, time period identifier TP and $J_u \in [1, k]$. A legitimate user cannot produce more than k valid $\mathsf{TIN}s$ (passing SoK verification) within time period TP. If two tokens $\mathsf{tok} = (J_u, C_u, D_u, E_u, F_u, \Pi_U^2)$ and $\mathsf{tok}' = (J_u', C_u', D_u', E_u', F_u', (\Pi_U^2)')$ are bound to the same TIN (indicating that they are associated with the same (TP, J_u)), $\mathcal{I}$ computes $w = H_1(C_u || D_u || E_u)$, $w' = H_1(C_u' || D_u' || E_u')$, $F = (F_u / F_u')^{1/(w-w')} = g_0^{1/(s+\alpha_1)}$, and sends $\mathsf{upk} = T_2 / (F^w) = Y_u$ to system manager, who reveals user's identity ID according to the registration information.

D. Security Analysis

Theorem IV.1. *The PkT-KVAC scheme is unforgeable if the LRSW assumption holds.*

Sketch of the Proof. In the formal security proof of Theorem IV.1, we build a challenger $\mathcal{C}$ to interact with a PPT adversary $\mathcal{A}$. If $\mathcal{A}$ breaks the unforgeability of PkT-KVAC with non-negligible advantage, $\mathcal{C}$ outputs a tuple to break the LRSW assumption. Given a tuple $\mathcal{T} = (g, \widetilde{A} = g^a, \widetilde{B} = g^b)$ and a LRSW oracle $\mathcal{O}_{LRSW}(\cdot)$, $\mathcal{C}$ simulates PkT-KVAC as follows. $\mathcal{C}$ selects $\alpha, \beta, \gamma \xleftarrow{\$} \mathbb{Z}_p^*$, and generates pp by setting $g_0 = g$, $g_2 = g_0^{\alpha}$, $g_3 = g_0^{\beta}$, $h_1 = g_0^{\gamma}$. In $\mathsf{IKeyGen}$, $\mathcal{C}$ sets $Z = g_0/(\widetilde{A}^{\alpha}(\widetilde{A}\widetilde{B})^{\beta} \cdot (h_1h_2)^{y_1y_2y_3} \cdot \prod_{i=0}^{n} u_i^{z_i}) = g_0/(g_2^a g_3^{ab} \cdot (h_1h_2)^{y_1y_2y_3} \cdot \prod_{i=0}^{n} u_i^{z_i})$, which implies that $\mathcal{C}$ learns noting about the issuer secret key $x_2 = a$, $x_3 = ab$. In $\mathsf{UKeyGen}$, $\mathcal{C}$ selects $t \xleftarrow{\$} \mathbb{Z}_p^*$, queries $\mathcal{O}_{LRSW}(t) \rightarrow (U, U_1 = U^b, U_2 = U^{a+tab})$, and sets $\mathsf{upk} = U_1$, implying that $\mathsf{upk} = g_0^{x_u} = U^b$. To simulate Issue, $\mathcal{C}$ selects $s \xleftarrow{\$} \mathbb{Z}_p^*$, and computes $V = g_1^{x_1} \cdot U_2 \cdot ((U_1)^{\gamma y_1} \cdot Y_2^s)^{y_3} \cdot \widetilde{u}_0^{H_1(TP,k)z_0} \prod_{i=1}^{n} \widetilde{u}_i^{H_1(\mathsf{attr}_i)z_i}$. To simulate Show, $\mathcal{C}$ sets $D_1 = Y_2^r \cdot U_1^{\gamma} = Y_2^r \cdot h_1^{x_u}$ and runs the simulator $\mathcal{S}$ to produce Π_U^2. It can be seen that $\mathcal{C}$ perfectly simulates $\mathsf{cred}_{TP,k}$ and tok without holding x_2, x_3, x_u. If $\mathcal{A}$ outputs a forgery with a well-formed token $tok^* = (J_u^*, C_u^*, D_u^*, E_u^*, F_u^*, (\Pi_U^2)^*)$, $\mathcal{C}$ can run the extractor $\mathcal{E}$ to extract r^*, t^*, s^*, x_u^*, computes $U^* = C_2^*/g_2^{r^*}$, $U_1^* = g_0^{x_u^*} = (U^*)^b$, $U_2^* = C_1^*/(g_0^{r^*} g_1^{x_1} (Y_1^{x_u^*} Y_2^{s^*})^{y_3} \cdot \widetilde{u}_0^{H_1(TP,k)z_0} \prod_{i=1}^{n} \widetilde{u}_i^{H_1(\mathsf{attr}_i^*)z_i}) = (U^*)^{a+t^*ab}$, and outputs (U^*, U_1^*, U_2^*) to break the LRSW assumption.

Theorem IV.2. *The PkT-KVAC scheme is anonymous and unlinkable if the DDH assumption holds.*

Sketch of the Proof. In the formal security proof of Theorem IV.2, we build a challenger $\mathcal{C}$ to interact with a PPT adversary $\mathcal{A}$. If $\mathcal{A}$ breaks the unforgeability of PkT-KVAC with non-negligible advantage ϵ, $\mathcal{C}$ breaks the DDH assumption with non-negligible advantage $\frac{\epsilon}{2}$. Given a tuple $\mathcal{T} = (g, \widetilde{A} = g^{\mathsf{a}}, \widetilde{B} = g^{\mathsf{b}}, \widetilde{C} = g^{\mathsf{c}})$, $\mathcal{C}$ simulates PkT-KVAC as follows. In Setup, $\mathcal{C}$ samples $\{\alpha_i\}_{i\in[0,3]}, \beta_1, \beta_2\beta_3, \{\gamma_k, \widetilde{\gamma}_k\}_{k\in[0,n]} \xleftarrow{\$} \mathbb{Z}_p^*$, computes $g_i = g^{\alpha_i}$ for $i \in [1,3]$, $u_k = g^{\gamma_k}$, $\widetilde{u}_k = g^{\widetilde{\gamma}_k}$ for $k \in [0,n]$, $h_1 = g^{\beta_1}$, $h_3 = g^{\beta_3}$, and sets $g_0 = \widetilde{A}^{\alpha_0}$, $h_2 = \widetilde{A}^{\beta_2}$. Then, $\mathcal{C}$ performs $\mathsf{IKeyGen}$, $\mathsf{UKeyGen}$ and Show as usual. To generate a challenge token for $\mathcal{A}$, $\mathcal{C}$ computes $C_1^* = \widetilde{C}^{\alpha_0} V_b = g_0^{\mathsf{c}} V_b$, $C_2^* = \widetilde{B}^{\alpha_2} U_b = g_2^{\mathsf{b}} U_b$, $C_3^* = \widetilde{B}^{\alpha_3} U_b^{t_b} = g_3^{\mathsf{b}} U_b^{t_b}$, $D_1^* = \widetilde{C}^{\beta_2 y_2} \cdot h_1^{x_{u_b}} = (g^{\mathsf{c}})^{\beta_2 y_2} \cdot h_1^{x_{u_b}}$, $D_2^* = \widetilde{B}^{\beta_1 y_1} \cdot h_2^{s_b} = Y_1^{\mathsf{b}} \cdot h_2^{s_b}$, $C_4^* = \widetilde{C}^{\alpha_0} / \widetilde{B}^{\alpha_2 x_2 + \alpha_3 x_3} \cdot (\widetilde{B}^{\beta_1})^{y_1y_2y_3} \cdot \widetilde{B}^{\prod_{i=0}^{n} \gamma_i z_i} \widetilde{C}^{y_1y_2y_3}$. For $i = 0 \vee \mathsf{attr}_i \in \mathsf{ATTR}_D$, $\mathcal{C}$ computes $E_i^* = \widetilde{B}^{\gamma_i} = u_i^{\mathsf{b}}$; otherwise, it computes $E_i^* = \widetilde{B}^{\gamma_i} \widetilde{u}_i^{H_1(\mathsf{attr}_i)} = u_i^{\mathsf{b}} \widetilde{u}_i^{H_1(\mathsf{attr}_i)}$. If the DDH challenge $(g, \widetilde{A}, \widetilde{B}, \widetilde{C})$ is a random quadruple, the token components C_1^*, C_4^*, D_1^* are random elements and therefore tok^* is independent to the user public key $g_0^{x_u}$. Thus, if $\mathcal{A}$ outputs a $b^* = b$ with non-negligible advantage, $\mathcal{C}$ outputs 1 to denote that the tuple $\mathcal{T} = (g, \widetilde{A}, \widetilde{B}, \widetilde{C})$ is a DDH quadruple; otherwise, it outputs 0 to denote that T is a random quadruple.

Theorem IV.3. *The PkT-KVAC scheme is k-detectable if the LRSW assumption holds.*

Theorem IV.4. *The PkT-KVAC scheme is exculpable if the DL assumption holds.*

The formal security proofs of Theorem IV.1-IV.4 are provided in Supplemental Material C.2-C.5.

V. Proposed PkT-SIN

A. Initialization Phase

System Setup. $(NCC \rightarrow (\mathsf{pp}, H_2, H_3, \mathsf{SE}, \mathsf{DSS}, \mathcal{L}_{user}, \mathcal{L}_{tok}))$. The network control center NCC sets the security parameter λ and the maximum number n of attributes in a credential. NCC runs PkT-KVAC.$\mathsf{Setup}(1^{\lambda}, 1^n) \rightarrow \mathsf{pp}$ to generate the public parameters. NCC specifies the the duration of each time period TP, and defines the maximal authentication time k in each time period according to the duration of TP and the service time of each LEO. Define the key space $\mathcal{K}$ for secret session key, $n_{\mathcal{K}}$ as the bit length of secret session key, ℓ_{sid} as the length of session identifier, hash functions $H_2 : \{0,1\}^* \rightarrow \{0,1\}^{n_{\mathcal{K}}}$ and $H_3 : \{0,1\}^* \rightarrow \{0,1\}^{\ell_{sid}}$, a symmetric encryption scheme $\mathsf{SE} = (\mathsf{KeyGen}, \mathsf{SEnc}, \mathsf{SDec})$ and a digital signature scheme $\mathsf{DSS} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$. NCC initializes two empty lists $\mathcal{L}_{user}$ and $\mathcal{L}_{tok}$ for saving user registration information and authentication tokens respectively.

Low Earth Orbit Satellite Setup. $(NCC \rightarrow LEO : (sk_{leo}, pk_{leo}))$. NCC runs PkT-KVAC.$\mathsf{IKeyGen}(\mathsf{pp}) \rightarrow (\mathsf{isk}, \mathsf{ipk})$ to generate issuer key pair $(\mathsf{isk}, \mathsf{ipk})$. Select $x_{leo} \xleftarrow{\$} \mathbb{Z}_p^*$ and compute $Y_{leo} = g_0^{x_{leo}}$. NCC sets the secret key of LEOs as $sk_{leo} = (x_{leo}, \mathsf{isk})$ and public key $pk_{leo} = (Y_{leo}, \mathsf{ipk})$.

Ground Station Setup.

Step 1. $(NCC \rightarrow GS : (sk_{gs}, pk_{gs}))$. NCC selects $x_{gs} \xleftarrow{\$} \mathbb{Z}_p^*$, computes $Y_{gs} = g_0^{x_{gs}}$ and runs $\mathsf{DSS.KeyGen} \rightarrow (Dsk_{gs}, Dvk_{gs})$ to generate a sign-verify key pair. NCC sets GS's secret key as $sk_{gs} = (x_{gs}, Dsk_{gs})$ and public $pk_{gs} = (Y_{gs}, Dpk_{gs})$. GS sets the long-term secret key $sk_{gs} = (x_{gs}, Dsk_{gs})$, the public key $pk_{gs} = (Y_{gs}, Dvk_{gs})$.

Step 2. $(GS \rightarrow LEO : (ID_{gs}, R_{gs}, Sig_{gs}))$. GS generates an ephemeral Diffie-Hellman (DH) share $(r_{gs}, R_{gs} = g_0^{r_{gs}})$ in each time period TP, and signs (ID_{gs}, R_{gs}, TP) by running $Sig_{gs} = \mathsf{DSS.Sign}_{Dsk_{gs}}(ID_{gs}||R_{gs}||TP)$. The tuple $(ID_{gs}, R_{gs}, Sig_{gs})$ is sent to each connected LEO when the system is set up.

Satellite User Setup. $(SU \rightarrow (\mathsf{usk}, \mathsf{upk}, ID))$. SU runs PkT-KVAC.$\mathsf{UKeygen}(\mathsf{pp}, ID) \rightarrow (\mathsf{usk}, \mathsf{upk})$ to generate the user secret-public key pair $(\mathsf{usk}, \mathsf{upk})$, and registers (ID, upk) to NCC, whhich is inserted to $\mathcal{L}_{user}$.

B. Credential Issue Phase

$((SU \leftrightarrow NCC) \rightarrow (\mathsf{cred}, k, J_u))$. When a satellite user SU subscribes to the SIN service, it pays for the service according to the periodic authentication time k. Then, SU interacts with NCC to issue the credential by executing the interactive protocol PkT-KVAC.$\mathsf{Issue}(\mathcal{U}(\mathsf{usk}, \mathsf{upk}, \mathsf{ATTR}) \leftrightarrow \mathcal{I}(\mathsf{isk}, TP, k)) \rightarrow (\mathsf{cred}_{TP,k}, \mathcal{D})$ with NCC, which returns SU an anonymous credential $\mathsf{cred}_{TP,k}$ supporting periodic k-time authentication in time period TP. NCC packages the service purchase protocol as SVC (e.g., the service plan, validity period, periodic authentication time k) and inserts $(\mathsf{upk}, \mathsf{cred}_{TP,k}, \mathsf{SVC})$ to $\mathcal{L}_{user}$. Note that NCC shares the issuer secret key isk with the LEOs for credential verification in initialization phase.

C. Session Establishment Phase

The session establishment process (shown in Fig. 3) consists of three steps. Firstly, SU generates an authentication token tok and a Diffie-Hellman (DH) share (r_u, R_u) for session key generation. Next, LEO verifies the identity of SU, generates DH key $CT_{L,U}$ for LEO-SU and $CT_{L,G}$ for LEO-GS, and then utilizes these keys to generate ciphertexts. Lastly, SU and GS respectively derive a session identifier sid and a secure session key ssk to complete the secure session establishment.

Step 1: SU selects $r_u \xleftarrow{\$} \mathbb{Z}_p^*$ to compute $R_u = g_0^{r_u}$. SU runs $\mathsf{PkT\text{-}KVAC.Show}(\mathsf{usk}, \mathsf{cred}_{TP,k}, \mathsf{ATTR}_D, \mathcal{D}, M = R_u)$ to compute $(\mathsf{tok}, \mathsf{TIN}, \mathcal{D}')$. SU sends $(\mathsf{tok}, \mathsf{TIN}, ID_{gs}, R_u, \mathsf{ATTR}_D)$ to LEO. Note that R_u is included in SoK of $\mathsf{PkT\text{-}KVAC.Show}$ to avoid middle-in-the-man attacks (tampering R_u).

Step 2: Receiving the message from SU, LEO runs $\mathsf{PkT\text{-}KVAC.Verify}(\mathsf{isk}, \mathsf{tok}, \mathsf{TIN}, \mathsf{ATTR}_D, M)$ to verify the legitimacy of SU. If the verification passes, LEO computes a symmetric key $EK_{L,U} = H_2(R_u^{x_{leo}})$ shared between LEO and SU. Then, LEO chooses the nearest ground station GS for SU to communicate with and calculates $EK_{L,G} = H_2(R_{gs}^{x_{leo}})$ as a shared key between LEO and GS. LEO sends $\mathsf{CT}_{L,U} \leftarrow \mathsf{SE.SEnc}_{EK_{L,U}}(ID_{gs}, pk_{gs}, R_{gs}, Sig_{gs})$ to SU, and $\mathsf{CT}_{L,G} \leftarrow \mathsf{SE.SEnc}_{EK_{L,G}}(R_u, TP, J_u)$ to GS. Then, LEO sends the tuple $(\mathsf{tok}, \mathsf{TIN})$ to NCC via a secure channel, which is inserted to the list $\mathcal{L}_{tok}$.

Step 3: Receiving the message from LEO, SU computes $EK_{L,U} = H_2(Y_{leo}^{r_u})$ to recover $(ID_{gs}, pk_{gs}, R_{gs}, Sig_{gs})$. If $\mathsf{DSS.Ver}_{Dvk_{gs}}(ID_{gs}||R_{gs}||TP, Sig_{gs})$ returns 1, SU calculates the secret session key between SU and GS as $ssk = H_2(Y_{gs}^{r_u}, R_{gs}^{r_u})$. Similarly, GS computes $EK_{L,G} = H_2(Y_{leo}^{r_{gs}})$ to recover (R_u, TP, J_u), and derives $ssk = H_2(R_u^{x_{gs}}, R_u^{r_{gs}})$. SU and GS respectively computes the session identifier $sid = H_3(ssk, TP, J_u)$. Therefore, the secure session between GS and SU is established with session identifier and secret session key pair (sid, ssk).

D. Handover Phase

The LEOs are constantly moving in their fixed orbits above the Earth's surface, which causes a dynamic topology of the SIN system. This dynamic feature brings a challenge to maintain a continuous and private session. Thus, it is necessary to design a seamless handover mechanism to ensure the real-time secure communication and quality of service in SIN. Meanwhile, the handover speed caused by the movement of LEO can be calculated from the altitude of the orbiting satellite, which is periodic and predictable. On the other hand, the movement of user is another factor causing the session handover, especially the mobile uses (such as vehicles) moving at a high speed, which requires the handover between LEOs as well as ground stations to provide continuous service. These handover and session update scenarios are described as follows.

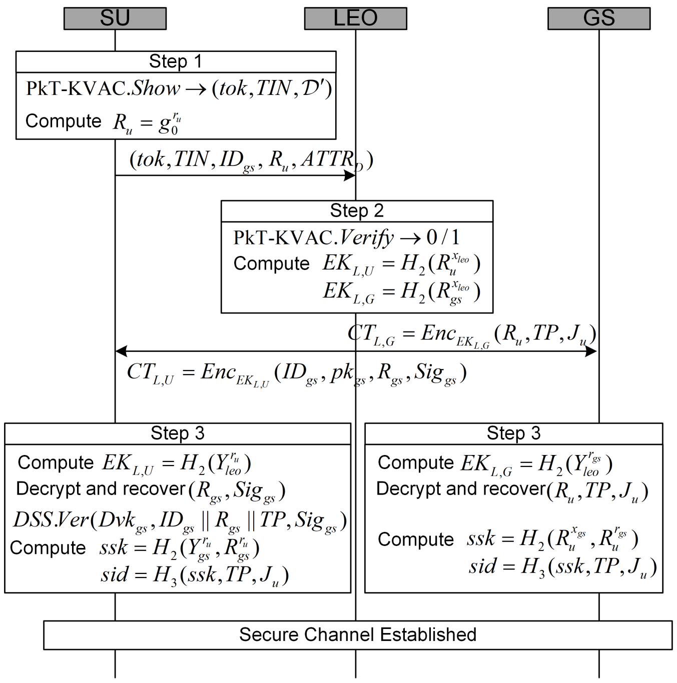

SU
LEO
GS
Step 1
$\mathrm{PkT\text{-}KVAC}.Show \rightarrow (tok, TIN, \mathcal{D}')$
Compute $R_u = g_0^{r_u}$
$(tok, TIN, ID_{gs}, R_u, ATTR_D)$
Step 2
$\mathrm{PkT\text{-}KVAC}.Verify \rightarrow 0/1$
Compute $EK_{L,U} = H_2(R_u^{x_{leo}})$
$EK_{L,G} = H_2(R_{gs}^{x_{leo}})$
$CT_{L,G} = Enc_{EK_{L,G}}(R_u, TP, J_u)$
$CT_{L,U} = Enc_{EK_{L,U}}(ID_{gs}, pk_{gs}, R_{gs}, Sig_{gs})$
Step 3
Compute $EK_{L,U} = H_2(Y_{leo}^{r_u})$
Decrypt and recover (R_{gs}, Sig_{gs})
$DSS.Ver(Dvk_{gs}, ID_{gs} \| R_{gs} \| TP, Sig_{gs})$
Compute $ssk = H_2(Y_{gs}^{r_u}, R_{gs}^{r_u})$
$sid = H_3(ssk, TP, J_u)$
Step 3
Compute $EK_{L,G} = H_2(Y_{leo}^{r_{gs}})$
Decrypt and recover (R_u, TP, J_u)
Compute $ssk = H_2(R_u^{x_{gs}}, R_u^{r_{gs}})$
$sid = H_3(ssk, TP, J_u)$
Secure Channel Established

Fig. 3: Workflow of Session Establishment

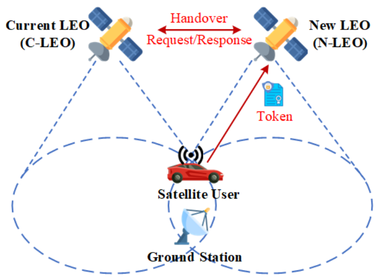


Fig. 4: LEO Handover for Mobile Users in Low Speed

1) S1: LEO Handover for Mobile Users in Low Speed: SU in low speed is relatively stationary compared with the constantly moving LEOs. In scenario 1 (S1), LEO handover is the dominant factor due to the transferring of satellite coverage. As shown in Fig. 4, the handover authentication occurs when the current LEO (C-LEO) is leaving SU's coverage area and cannot continue to provide SIN service, where the satellite user SU anonymously switches his connection to a new LEO (N-LEO). Assume that there are secure communication channels between NCC and LEO. The process of LEO handover (shown in Fig. 5) consists of two steps. Firstly, SU generates an unlinkable authentication token tok' to authenticate itself to the new satellite N-LEO. If the authentication is successful, N-LEO transmits the ciphertext $\mathsf{CT}'_{L,G}$ that includes the session identifier sid to GS, enabling SU and GS to resume their previous communication session.

Step 1: According to the service purchase protocol (in §V-B), SU is allowed to perform at most k anonymous handover authentication in TP. If the dispenser $\mathcal{D}$ is not empty, SU selects $r'_u \xleftarrow{\$} \mathbb{Z}_p^*$ to compute $R'_u = g_0^{r'_u}$, and runs $\mathsf{PkT\text{-}KVAC.Show}(\mathsf{usk}, \mathsf{cred}, \mathsf{ATTR}_D, \mathcal{D}, M = (sid, R'_u)) \rightarrow$

$(\mathsf{tok}, \mathsf{TIN}, \mathcal{D}')$ to generate an authentication token. SU sends the ciphertext of $\mathsf{CT}'_{L,U} = \mathsf{SE.SEnc}_{EK_{L,U}}(\mathsf{tok}, \mathsf{TIN}, M)$ along with $(ID_{gs}, R'_u, \mathsf{ATTR}_D)$ to N-LEO as the handover authentication request.

Step 2: Recovering the message, N-LEO runs PkT-KVAC.$\mathsf{Verify}(\mathsf{isk}, \mathsf{tok}, \mathsf{TIN}, \mathsf{ATTR}_D, M)$ to verify tok. If the verification passes, SU continues the secure session with GS through N-LEO using the current session identifier and secret session key pair (sid, ssk).

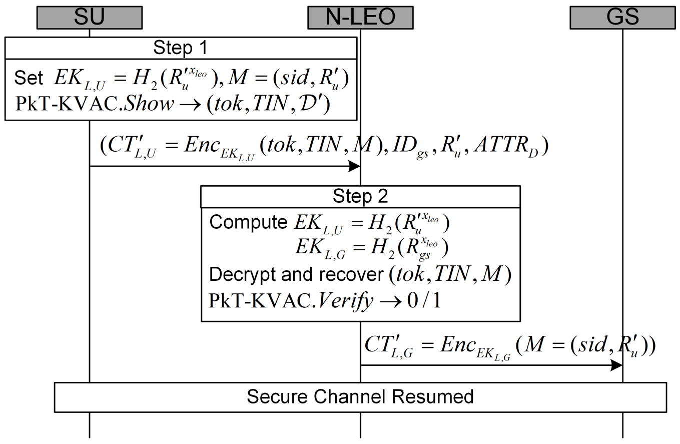

SU
N-LEO
GS
Step 1
Set $EK_{L,U} = H_2(R_u'^{x_{leo}}), M = (sid, R'_u)$
PkT-KVAC.$Show \rightarrow (tok, TIN, \mathcal{D}')$
$(CT'_{L,U} = Enc_{EK_{L,U}}(tok, TIN, M), ID_{gs}, R'_u, ATTR_D)$
Step 2
Compute $EK_{L,U} = H_2(R_u'^{x_{leo}})$
$EK_{L,G} = H_2(R_{gs}^{x_{leo}})$
Decrypt and recover (tok, TIN, M)
PkT-KVAC.$Verify \rightarrow 0/1$
$CT'_{L,G} = Enc_{EK_{L,G}}(M = (sid, R'_u))$
Secure Channel Resumed

Fig. 5: Workflow of Handover Scenario 1

2) S2: GS Handover for Mobile Users in High Speed: When the satellite user moves at a high speed (e.g., various types of vehicles), the nearest ground stations to mobile users are kept on changing in this scenario (S2). As shown in Fig. 6, SU is under the coverage of the same LEO, but switches from the current ground station (GS) to a new GS (GS′) due to the the fast movement. The LEO needs to help the satellite user to establish a new secure session with GS′ using an updated session identifier and secret session key (sid', ssk'). The process of GS handover (shown in Fig. 7) consists of two steps. When detecting the most recent GS change, LEO sends encrypted handover notifications to SU and GS, which include the GS identifier $ID_{gs'}$, the DH share $R_{gs'}$, the existing session ID sid, and other information. Similar to § V-C, SU and GS asynchronously generate a new session identifier sid' a session key ssk'.

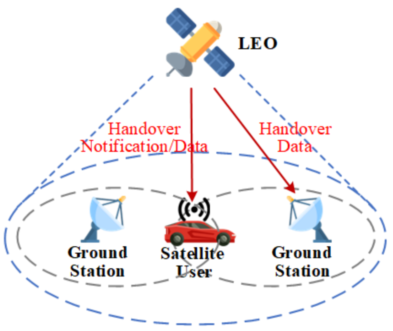


Fig. 6: GS Handover for Users in High Speed

Step 1:

(1) When LEO detects that the nearest ground station to satellite user has changed to GS′, LEO sends the ciphertext of $\mathsf{SE.SEnc}_{EK_{L,U}}(ID_{gs'}, pk_{gs'}, R_{gs'}, Sig_{gs'})$ and a GS switching notification to SU.

(2) LEO calculates $EK_{L,G'} = H_2(R_{gs'}^{x_{leo}})$ as a shared key between LEO and GS′, sends the ciphertexts of $\mathsf{CT}_{L,G'} = \mathsf{SE.SEnc}_{EK_{L,G'}}(sid, R_u, TP, J_u)$ to GS′, and sends $\mathsf{CT}_{L,U} \leftarrow \mathsf{SE.SEnc}_{EK_{L,U}}(ID_{gs'}, pk_{gs'}, R_{gs'}, Sig_{gs'})$ to SU.

Step 2:

(1) Receiving the notification and ciphertext from LEO, SU recovers $(ID_{gs'}, pk_{gs'}, R_{gs'}, Sig_{gs'})$ using $EK_{L,U}$. If $\mathsf{DSS.Ver}_{Dvk_{gs'}}(ID_{gs'} || R_{gs'}, Sig_{gs'})$ returns 1, SU updates the secret session key (between SU and N-GS) as $ssk' = H_2(Y_{gs'}^{r_u}, R_{gs'}^{r_u})$, and the new session identifier as $sid' = H_3(sid, ssk', TP, J_u)$.

(2) Receiving the ciphertext from LEO, GS′ calculates $EK_{L,G'} = H_2(Y_{leo}^{r_{gs'}})$ to recovering (sid, R_u, TP, J_u). GS′ computes $ssk' = H_2(R_u^{x_{gs'}}, R_u^{r_{gs'}})$. Then, GS′ computes the new session identifier $sid' = H_3(sid, ssk', TP, J_u)$.

Therefore, SU and GS′ shares the updated secure session information (sid', ssk').

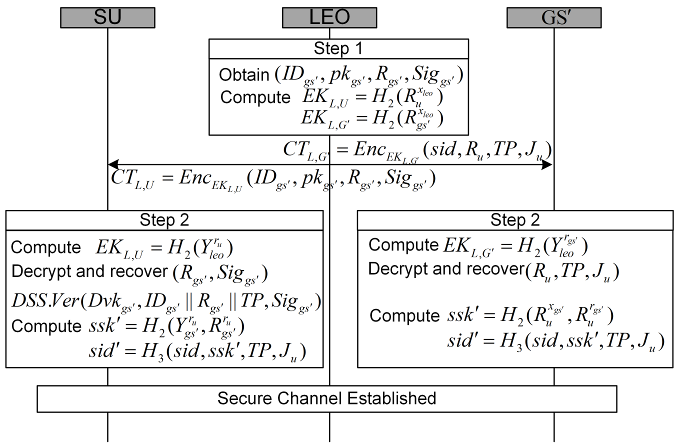

SU
LEO
GS′
Step 1
Obtain $(ID_{gs'}, pk_{gs'}, R_{gs'}, Sig_{gs'})$
Compute $EK_{L,U} = H_2(R_u^{x_{leo}})$
$EK_{L,G'} = H_2(R_{gs'}^{x_{leo}})$
$CT_{L,G'} = Enc_{EK_{L,G'}}(sid, R_u, TP, J_u)$
$CT_{L,U} = Enc_{EK_{L,U}}(ID_{gs'}, pk_{gs'}, R_{gs'}, Sig_{gs'})$
Step 2
Compute $EK_{L,U} = H_2(Y_{leo}^{r_u})$
Decrypt and recover $(R_{gs'}, Sig_{gs'})$
$DSS.Ver(Dvk_{gs'}, ID_{gs'} || R_{gs'} || TP, Sig_{gs'})$
Compute $ssk' = H_2(Y_{gs'}^{r_u}, R_{gs'}^{r_u})$
$sid' = H_3(sid, ssk', TP, J_u)$
Step 2
Compute $EK_{L,G'} = H_2(Y_{leo}^{r_{gs'}})$
Decrypt and recover(R_u, TP, J_u)
Compute $ssk' = H_2(R_u^{x_{gs'}}, R_u^{r_{gs'}})$
$sid' = H_3(sid, ssk', TP, J_u)$
Secure Channel Established

Fig. 7: Workflow of Handover Scenario 2

3) S3: LEO and GS Handover for Users in High Speed: For the satellite users with high speed, such as aircrafts and vehicles, the handover may include the satellite coverage transfer and also the grand station switching (shown in Fig. 8). In this scenario (S3), the secure session not only switches from current LEO to a new LEO, but also from GS to GS′. SU is requested to authenticate to the new satellite and then update the secure session information with the new ground station. In this scenario, SU executes the session establishment protocol in §V-C (shown in Fig. 3) with N-LEO and GS′ to derive the new session identifier and secret session key pair (sid', ssk').

E. Dishonest User Revealing Phase

The network control center NCC monitors the authentication record $(\mathsf{TIN}, \mathsf{tok})$ sent from satellites and inserts it to $\mathcal{L}_{tok}$. If two tokens tok and tok' are found to be bound to the same token series number TIN, NCC runs PkT-KVAC.$\mathsf{Reveal}(\mathsf{TIN}, \mathsf{tok}, \mathsf{tok}') \rightarrow \mathsf{upk}$ to disclose the user

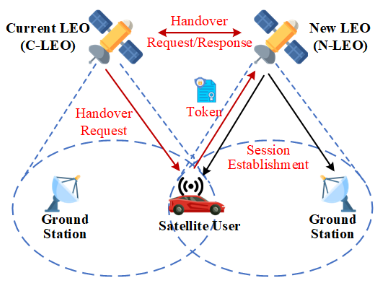


Fig. 8: LEO and GS Handover for Users in High Speed

public key upk. Then, it searches the user list $\mathcal{L}_{user}$ using upk and outputs (ID, upk).

Discussion 1. PkT-SIN provides periodic k-time anonymous authentication. In pay-as-you-go model, the value of k maybe determined by the subscription fee paid by SU. Higher subscription fees enable SU to obtain a larger value of k for each time period. The specific value of k for each SU should be determined by the SIN operators.

Discussion 2. Hash-based lightweight protocols [4, 38, 39] provide no unlinkability for SU against malicious NCC and LEOs. For instance, the schemes in [4, 38, 39] relying on hash-based lightweight protocols only offer anonymity against external eavesdroppers, but fail to achieve unlinkability for users against malicious NCC and LEOs due to the exposure of user's public key or even shared secret key.

Protocols in [3, 5, 7] and PkT-SIN tackle this problem through randomization and zero-knowledge proofs (ZKP) techniques. For instance, in PkT-KVAC.Show, the authentication token tok is essentially a randomization of the anonymous credential $\mathsf{cred}_{TP,k}$ based on a random nonce r, ensuring that it cannot be linked to $\mathsf{cred}_{TP,k}$. Besides, the Show algorithm utilizes ZKP to prove that $\mathcal{U}$ possesses the knowledge x_u in tok without revealing user's secret key $\mathsf{usk} = x_u$. Leveraging these techniques, PkT-SIN achieves anonymity and unlinkability against third-party attackers as well as malicious NCC and LEO.

VI. Security Analysis

Theorem VI.1. *PkT-SIN realizes authenticated key exchange (AKE) if the DDH assumption holds, PkT-KVAC and DSS are unforgeable, SE satisfies confidentiality, and hash functions are random oracles.*

Proof. When an adversary outputs a session in the $\mathsf{Test\text{-}Session}$ query (defined in the security model in Supplemental Material D), there must be a partner instance, otherwise, we can make use of the adversary to break the unforgeability of PkT-KVAC and DSS or the confidentiality of SE. We detail the proof in Supplemental Material E.

Theorem VI.2. *PkT-SIN satisfies anonymity for satellite users if PkT-KVAC satisfies anonymity and the token series numbers are not reused.*

Proof. The anonymity of PkT-SIN mainly relies on that of PkT-KVAC. We detail the proof in Supplemental Material E.

The other security requirements of PkT-SIN are discussed below.

• *Mutual Authentication*. In PkT-SIN, mutual authentication between the GS and SU is achieved. In the session establishment phase, GS is able to authenticate the legitimacy of SU by invoking PkT-KVAC.Verify algorithm to verify whether the authentication token tok and identifier TIN are valid or not. Due to the unforgeability of PkT-KVAC, adversaries cannot pass the verification by forging a valid tuple $(\mathsf{tok}, \mathsf{TIN})$. On the other hand, SU authenticates GS using the signature $Sig_{gs} = \mathsf{DSS.Sign}_{Dsk_{gs}}(ID_{gs}||R_{gs}||TP)$, which is guaranteed by the unforgeability of DSS. Therefore, PkT-SIN provides a secure mutual authentication mechanism between GS and SU.

• *Key Forward/Backward Secrecy*. PkT-SIN leverages the decisional Diffie-Hellman (DDH) mechanism to achieve key forward secrecy (KFS) and key backward secrecy (KBS) between GS and SU. In the session establishment phase, the secret session key is computed as $ssk = H_2(Y_{gs}^{r_u}, R_{gs}^{r_u}) = H_2(R_u^{x_{gs}}, R_u^{r_{gs}}) = H_2(g_0^{r_u \cdot x_{gs}}, g_0^{r_u \cdot r_{gs}})$, where $r_u, r_{gs} \in_R \mathbb{Z}_p^*$ are randomly selected. If an attacker attempts to obtain ssk, he has to derive r_u and r_{gs} from R_u and R_{gs}, respectively, which is equivalent to solving the DL hardness problem. It is evident that the calculation of current ssk is independent of the previous secret session key. Hence, the attacker cannot deduce the previous session key from the current one. Similarly, the compromise of the current session key does result in the disclosure of the future key. Therefore, PkT-SIN achieves KFS/KBS to protect the secret session keys.

• *User Anonymity and Unlinkability*. The anonymity and unlinkability of PkT-KVAC are formally proved in Theorem IV.2. PkT-SIN inherits these security properties from PkT-KVAC since the satellite users authenticate to the ground stations by showing their authentication tokens (generated using PkT-KVAC.Show algorithm).

• *Accountability*. Once the satellite user produces more than k valid TINs within the same period TP, this misbehavior will be discovered by detecting repeating token series numbers. In our system, TIN is associated with TP, $J_u \in [1, k]$ and an encoding function f. The real identity of a malicious satellite user can be disclosed using the two tokens $(\mathsf{tok}, \mathsf{tok}')$ associated with the same TIN, where accountability is applied.

• *Unlinkability*. PkT-SIN inherits the unlinkability from PkT-KVAC. The authentication token generation algorithm TokGen of PkT-KVAC derives a token tok from anonymous credential $\mathsf{cred}_{TP,k}$ with re-randomization operations in order to achieve unlinkability. The formal security proof of PkT-KVAC's unlinkability is shown in Supplemental Material B. It ensures that the multiple authentications of SU cannot be linked together.

• *Resistance to Replay Attacks*. PkT-SIN is designed for periodic k-time authentication in a designated time period. If an attacker replays an authentication token of a legitimate SU, it can be discovered by detecting repeating token series numbers. If the repeated tuple $(\mathsf{tok}, \mathsf{TIN})$ is discovered, the network access request will be rejected to resist a replay attack. In this case, the identity of legitimate SU will not be revealed since the attacker submits the same tuple $(\mathsf{tok}, \mathsf{TIN})$ to GS.

TABLE II: Comparison of Secure Space Information Systems

No.	Security requirements/Properties	[7]	[6]	[5]	[4]	[3]	[33]	PkT-SIN
C1	k-time anonymous authentication	×	×	×	×	×	√	√
C2	Handover	×	×	√	√	×	×	√
C3	Accountability	√	×	√	√	√	√	√
C4	Reveal violator's identity without TTP	×	N.A.	×	×	×	√	√
C5	Mutual authentication	√	×	×	√	√	×	√
C6	Anonymity	√	√	√	√	√	√	√
C7	Unlinkability	√	×	√	√	√	√	√
C8	Selective attribute disclosure	√	√	×	×	×	×	√
C9	Key forward/backward secrecy	×	N.A.	√	√	√	N.A.	√
C10	Resistance to replay attacks	√	√	√	√	√	√	√
C11	Resistance to impersonation attacks	√	×	×	√	√	×	√
C12	Resistance to MitM attacks	√	×	×	√	√	×	√
C13	Resistance to DDoS attacks	×	×	×	×	×	√	√

TTP: Trusted Third Party; N.A.: Not Applicable.

The attacker is unable to forge a valid tuple $(\mathsf{tok}', \mathsf{TIN})$ due to the unforgeability of PkT-KVAC.

• *Resistance to Impersonation Attacks*. Suppose an attacker attempts to impersonate a legitimate SU to obtain network access service. For this purpose, the attacker has to forge an authentication token that can pass the verification of PkT-KVAC.Verify algorithm. However, the attacker is not able to derive a valid token without SU's private key. On the other hand, the signature generated by DSS prevents the impersonation of GS. Hence, the impersonation attack cannot succeed.

• *Resistance to MitM Attacks*. A MitM attacker attempts to secretly intercept and alter the communication between SU and GS who believe they are directly communicating with each other. The message R_u sent by SU cannot be forged by the attacker since it is authenticated in tok. The above analysis of resistance to impersonation attacks indicates a MitM attacker will fail to impersonate either party. Therefore, PkT-SIN can resist MitM attack.

• *Resistance to DDoS Attacks*. PkT-SIN effectively resists DDoS attacks due to it limits the number of authentication attempts a user can make in a defined time period, thus preventing unauthorized users from overwhelming the system with excessive requests and ensuring continuous service availability. Additionally, the system has mechanisms for quickly detecting and identifying users who exceed the limits, further enhancing its resilience against DDoS attacks. The ability of PkT-SIN to resist DDoS attacks does not decrease or even fail if legitimate users require a large amount of access. This is because a legitimate user needs to pay a large amount of subscription fee to the SIN operator in order to to get a larger k value in pay-as-you-go model, which is discussed in § V.

• *Resistance to Eavesdropping Attacks*. PkT-SIN utilizes a symmetric encryption scheme to prevent session eavesdropping. Specifically, the messages exchanged between LEO and GS are encrypted using the symmetric key $EK_{L,G}$, while the messages between LEO and SU are encrypted with the symmetric key $EK_{L,U}$. The messages $(\mathsf{tok}, \mathsf{TIN}, ID_{gs}, R_u, \mathsf{ATTR}_D)$ sent in plaintext are signed with the signature of knowledge algorithm, preventing the attacker to launch MitM attacks or impersonation attacks. Once the secure session is established, the messages exchanged between SU and GS are encrypted using the secret session key ssk.

• *Resistance to Device Physical Capture Attacks*. LEO operates at an altitude between approximately 500 to 2,000 km, orbiting the planet at a speed of 7,000 to 8,000 km/h. It is highly impractical to mount device physical capture attacks against LEOs. GS serves as a critical ground infrastructure that is typically well-secured and closely monitored to protect it from unauthorized access and physical capture. On the other hand, SU devices may subject to device physical capture attacks. In the pay-as-you-go business model, attackers cannot access additional SIN services without making further payments once their existing paid services are depleted. Therefore, this model does not result in additional financial losses for the SIN system.

VII. Performance Analysis

A. Comparison

We compare PkT-SIN with several existing anonymous authentication protocols for SINs [3–7, 33] in terms of security properties. As shown in Table II, our comparison focuses on k-time anonymous authentication (C1), handover (C2), accountability (C3), revealing violator's identity without TTP (C4), mutual authentication (C5), anonymity (C6), unlinkability (C7), selective attribute disclosure (C8), key forward/backward secrecy (C9), resistance to replay attacks (C10), resistance to impersonation attacks (C11), resistance to MitM attacks (C12), and resistance to DDoS attacks (C13).

A key feature of PkT-SIN and the work in [33] is the concept of periodic k-time anonymous authentication (C1). This feature empowers each user to anonymously present their credentials a maximum of k times during any single time period. This attribute is of paramount importance in enabling pay-as-you-use. Periodic k-time authentication is crucial to resist DDoS attacks (C13) by limiting the number of authentication attempts each user can make within any defined time period, preventing malicious users from overwhelming LEOs with excessive requests and enabling quick detection of unauthorized access. Except [33], the other protocols [3–7] in our comparison do not offer this feature, rendering them susceptible to DDoS attacks.

TABLE III: Computation and Communication Cost of PkT-KVAC and PkT-SIN

	Algorithms	Parameter	Phase	Computation cost	Communication cost
PkT-KVAC	Setup	pp	–	–	$(2n+9)\lvert\mathbb{G}\rvert$
	IKeyGen	(isk,ipk)	–	$(n+8)t_e$	$(n+7)\lvert\mathbb{Z}_p\rvert+4\lvert\mathbb{G}\rvert$
	UKeyGen	(usk,upk)	–	t_e	$\lvert\mathbb{Z}_p\rvert$
	Issue	Cm	–	$2t_e$	$\lvert\mathbb{G}\rvert$
		Π_U^1	Proof	$3t_e+t_{h_1}$	$2\lvert\mathbb{Z}_p\rvert+2\lvert\mathbb{G}\rvert$
			Verify	$5t_e$	–
		Π_I	Proof	$(2n+11)t_e+(n+1)t_{h_1}$	$(n+7))\lvert\mathbb{Z}_p\rvert+3\lvert\mathbb{G}\rvert$
			Verify	$(2n+14)t_e+(n+1)t_{h_1}$	–
		$\mathsf{cred}_{TP,k}$	–	$(n+5)t_e+(n+1)t_{h_1}$	$2\lvert\mathbb{Z}_p\rvert+2\lvert\mathbb{G}\rvert$
	TokGen	tok	–	$(n+22)t_e+t_{h_1}$	$8\lvert\mathbb{Z}_p\rvert+(n+14)\lvert\mathbb{G}\rvert$
	Verify	–	–	$(n+23)\cdot t_e$	–
	Reveal	–	–	$2t_e+2t_{h_1}$	–

	Phases	Operations	Computation cost	Communication cost
PkT-SIN	Initialization	System Setup	–	$(2n+9)\lvert\mathbb{G}\rvert$
		Low Earth Orbit Satellite Setup	t_e	$\lvert\mathbb{G}\rvert$
		Ground Station Setup	$t_{\mathsf{DSS.KeyGen}}+t_{\mathsf{DSS.Sig}}+2t_e$	$\lvert\mathsf{Sig}\rvert+2\lvert\mathbb{G}\rvert$
		Satellite User Setup	t_e	$\lvert\mathbb{G}\rvert$
	Issue	Credential Issue	$(n+5)t_e+(n+1)t_{h_1}$	$2\lvert\mathbb{Z}_p\rvert+2\lvert\mathbb{G}\rvert$
	Session Est.	Step 1-3	$t_{\mathsf{DSS.Ver}}+2t_{\mathsf{SE.SEnc}}+2t_{\mathsf{SE.SDec}}$ $+(2n+51)t_e+2t_{h_1}+3t_{h_2}+t_{h_3}$	$2\lvert\mathsf{CT}_{\mathsf{Enc}}\rvert$ $+16\lvert\mathbb{Z}_p\rvert+(2n+32)\lvert\mathbb{G}\rvert$
	Handover	LEO Handover for Users in Low Speed	$(2n+48)t_e+2t_{h_1}$	$8\lvert\mathbb{Z}_p\rvert+(n+16)\lvert\mathbb{G}\rvert$
		GS Handover for Users in High Speed	$t_{\mathsf{DSS.Ver}}+2t_{\mathsf{SE.SEnc}}+2t_{\mathsf{SE.SDec}}$ $+6t_e+4t_{h_2}+2t_{h_3}$	$2\lvert\mathsf{CT}_{\mathsf{Enc}}\rvert$
		LEO and GS Handover for Users in High Speed	$t_{\mathsf{DSS.Ver}}+2t_{\mathsf{SE.SEnc}}+2t_{\mathsf{SE.SDec}}$ $+(2n+51)t_e+2t_e+3t_{h_2}+t_{h_3}$	$2\lvert\mathsf{CT}_{\mathsf{Enc}}\rvert$ $+16\lvert\mathbb{Z}_p\rvert+(2n+32)\lvert\mathbb{G}\rvert$

t_e: time for exponent in group $\mathbb{G}$; t_{h_1}: time for hash H_1 calculation; t_{h_2}: time for hash H_2 calculation; t_{h_3}: time for hash H_3 calculation;
$|\mathbb{Z}_p|$: size of element in $\mathbb{Z}_P$; $|\mathbb{G}|$: size of element in $\mathbb{G}$; n: attribute number;
$t_{\mathsf{DSS.KeyGen}}$, $t_{\mathsf{DSS.Sig}}$, $t_{\mathsf{DSS.Ver}}$: execution time for KeyGen, Sig, Ver algorithms in DSS scheme, resp.; $|\mathsf{Sig}|$: size of signature in DSS scheme;
$t_{\mathsf{SE.SEnc}}$, $t_{\mathsf{SE.SDec}}$: execution time for SEnc, SDec algorithms in SE scheme, resp.; $|\mathsf{CT}_{\mathsf{Enc}}|$: size of ciphertext in SE scheme;

TABLE IV: Computation/Storage/Propagation Overheads Comparison

Schemes	Computation Costs (Session Establishment)		Storage Overheads		Total Authentication Delay
	LEO	SU	LEO	SU	
[3]	$13t_e+5t_p$	$13t_e+2t_p$	$\lvert\mathbb{Z}_p\rvert+5\lvert\mathbb{G}\rvert$	$\lvert\mathbb{Z}_p\rvert+6\lvert\mathbb{G}\rvert$	$26t_e+7t_p+2\cdot T_{L,U}$
[5]	$11t_e+2t_p$	$15t_e$	$\lvert\mathbb{Z}_p\rvert+5\lvert\mathbb{G}\rvert\rvert$	$\lvert\mathbb{Z}_p\rvert+8\lvert\mathbb{G}\rvert$	$26t_e+2t_p+2\cdot T_{L,U}$
[7]	$2t_e+4t_p$	$8t_e+2t_p$	$\lvert\mathbb{Z}_p\rvert+5\lvert\mathbb{G}\rvert\rvert$	$\lvert\mathbb{Z}_p\rvert+7\lvert\mathbb{G}\rvert$	$11t_e+6t_p+2\cdot T_{L,U}$
PkT-SIN	$26t_e$	$27t_e$	$7\lvert\mathbb{Z}_p\rvert+12\lvert\mathbb{G}\rvert\rvert$	$3\lvert\mathbb{Z}_p\rvert+14\lvert\mathbb{G}\rvert$	$53t_e+2\cdot T_{L,U}$

t_e/t_p: time for exponential/pairing computation; $|\mathbb{Z}_p|/|\mathbb{G}|$: size of element in $\mathbb{Z}_p/\mathbb{G}$;
$T_{L,U}$: the time cost of signal propagation between LEO and SU

Most of other protocols in our comparison support accountability (C3), anonymity (C6), unlinkability (C7), and resistance to replay attacks (C10). However, the functionalities related to handover (C2), revealing any violator's identity with TTP involvement (C4), mutual authentication (C5), and selective attribute disclosure (C8) are not comprehensively supported. While certain protocols [3–5, 7] do provide accountability (C3), they necessitate the involvement of a TTP to disclose the identity of a violator. Consequently, they fall short of satisfying C4. Finally, the protocols proposed in [5, 6, 33] do not support mutual authentication (C5), rendering them incapable of resisting impersonation attacks (C11) and MitM attacks (C12).

After conducting a comprehensive comparison, it becomes evident that PkT-SIN stands out as the only secure communication protocol that successfully meets all the security requirements and desirable properties for SIN.

B. Theoretical Analysis

We give out the theoretical analysis of PkT-KVAC and PkT-SIN to analyze the computation/communication costs (shown in Table III). In PkT-KVAC scheme, the UKeyGen algorithm is highly efficient, requiring only one exponentiation computation. On the other hand, the Setup and IKeyGen algorithms exhibit linear growth with attribute number n. In the Issue algorithm, $\mathcal{U}$ initiates the process by generating a commitment Cm (for the secret key $\mathsf{upk}=x_u$) and a SoK Π_U^1, which consumes minimal and constant time. Subsequently, $\mathcal{I}$ verifies Π_U^1 using 5 exponentiation computations computations, and generates SoK Π_I (comprising $n+7$ elements from $\mathbb{Z}_p$ and 3 elements from $\mathbb{G}$). $\mathcal{U}$ verifies Π_I, and obtains the credential $\mathsf{cred}_{TP,k}$ with a constant size of $2|\mathbb{Z}_p|+2|\mathbb{G}|$. The computation and communication costs of TokGen and Verify algorithms linearly increase with n.

In PkT-SIN, the setup algorithms for low Earth orbit satellites, ground stations, and satellite users during the initialization phase are highly efficient. The overheads for protocol setup and credential issuance are identical to those in PkT-KVAC. The session establishment phase comprises the algorithms DSS.Ver, SE.SEnc, SE.SDec, PkT-KVAC.Show, and PkT-KVAC.Verify, with the overheads increasing linearly with n. PkT-SIN encompasses three handover scenarios. Among these scenarios, the 2nd scenario is notably the most efficient. Comparing the computational costs of the other two handover scenarios is challenging due to the varying parameter n. The communication overhead for the 1st scenario is lower than that of the 3rd one.

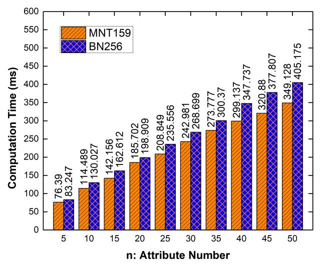
Table IV presents a comparative analysis of the computation costs (during session establishment), storage overheads, and total authentication delay for schemes [3, 5, 7] and PkT-SIN. Since the schemes [3, 5, 7] do not support attribute-based authentication, the comparison sets $n = 1$ for them indicating that there is only one attribute (i.e., identity) for SU. In comparison to schemes [3, 5, 7], PkT-SIN inherits the efficiency advantage of the KVAC primitive to avoid expensive pairing computations for LEO and SU, which effectively reduces the computation overheads for authentication token generation and verification. We also performed an experimental comparison for LEO under curves MNT159 and BN256, which is depicted in Fig. 10. The comparison shows that the storage overheads of PkT-SIN are slightly higher than the other schemes [3, 5, 7], with measurement values shown in Fig. 11. The total authentication delay includes the computation overheads of LEO, SU, and the signal propagation delay $2 \cdot T_{L,U}$. By eliminating time-consuming bilinear pairing operations, the total authentication delay of PkT-SIN is among the lowest in comparison to other schemes, as evidenced by the experimental results shown in Fig. 12.

Algo.	n	**Computation and Communication Costs**			
		MNT159 (80-bit Security)		BN256 (100-bit Security)	
		Comp. (ms)	Comm. (KB)	Comp. (ms)	Comm. (KB)
Setup	10	6.05	0.850	6.494	3.625
IKeyGen	10	14.426	0.615	18.205	1.164
UKeyGen	–	0.713	0.059	1.865	0.164
Reveal	–	2.37	–	3.947	–
Issue	10	113.442	0.850	116.238	1.906
	20	179.34	1.143	191.173	2.297
	30	238.613	1.436	262.401	2.688
	40	288.827	1.729	337.054	3.078
	50	342.06	2.021	417.347	3.469
TokGen	10	28.784	0.938	35.44	3.398
	20	35.575	1.230	58.573	4.648
	30	46.242	1.523	64.589	5.898
	40	55.06	1.816	70.424	7.148
	50	71.917	2.109	79.685	8.398
Verify	10	42.859	–	52.955	–
	20	59.127	–	75.033	–
	30	72.036	–	112.181	–
	40	95.082	–	108.188	–
	50	109.874	–	122.702	–

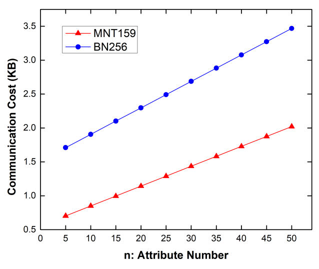
TABLE V: Performance of PkT-KVAC

Algo.	n	**Computation and Communication Costs**			
		MNT159 (80-bit Security)		BN256 (100-bit Security)	
		Comp. (ms)	Comm. (KB)	Comp. (ms)	Comm. (KB)
System Setup	10	19.728	1.464	22.4	4.789
LEO Setup	10	0.726	0.674	1.065	1.328
GS Setup	–	3.789	0.264	4.189	0.695
User Setup	–	0.692	0.117	0.878	0.328
Handover S1	10	71.631	1.143	79.494	4.188
Handover S2	10	9.631	0.264	11.716	0.781
Handover S3	10	80.796	1.318	81.022	4.680

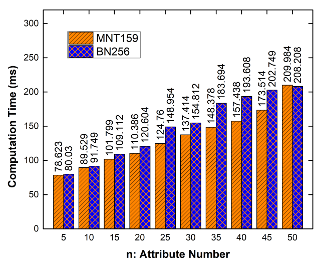
TABLE VI: Performance of PkT-SIN



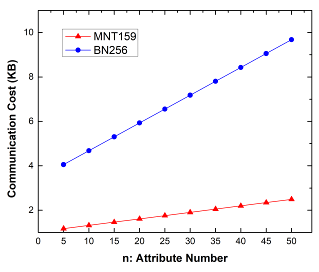
(a) Comp. Cost of Issue



(b) Comm. Cost of Issue



(c) Comp. Cost of Session Est.



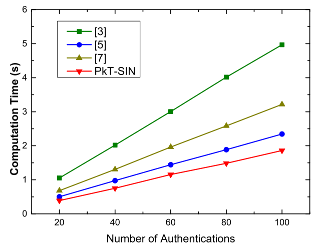
(d) Comm. Cost of Session Est.

Fig. 9: Evaluation of Issue and Session Establishment

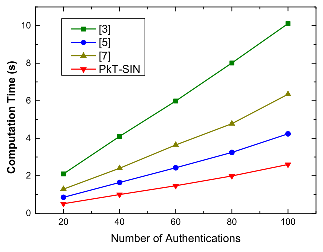
C. *Experimental Evaluation*

We evaluate the performance of PkT-KVAC and PkT-SIN on two elliptic curves selected from the MIRACL library [35] for evaluation, including MNT159 (80-bit security) and BN256 (100-bit security). We use AES-CTR with 128-bit keys to instantiate the symmetric encryption scheme SE in PkT-SIN, and use Schnorr signature [36] as the digital signature scheme DSS. The source code of our experiments is written in C/C++. We conduct experiments on a laptop with Intel® Core™ i7-6700HQ CPU and 4GB RAM running 64-bit Ubuntu 16.04.05 LTS. For each test case, we report an average of over 100 executions.

We present the performance of PkT-KVAC in Table V, which contains the computation/communication (abbreviated as Comp./Comm.) overheads of diverse algorithms. As depicted in Table III-V, the execution times for UKeyGen and Reveal remain constant, with computational costs of 1.865 ms and 3.947 ms on BN256, respectively. For the evaluations of Issue, TokGen and Verify, we set the attribute number n varying from 10 to 50. When $n = 30$, Issue takes 238.613 ms/1.436 KB, 262.401 ms/2.688 KB to achieve 80-bit, 100-bit security, respectively. It is evident that TokGen and Verify are more efficient than Issue. When $n = 30$,

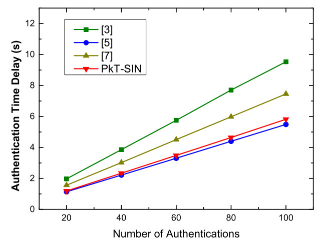


(a) Comp. Cost under MNT159

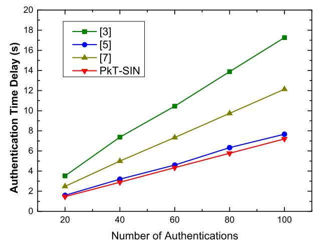


(b) Comp. Cost under BN256

Fig. 10: LEO Computation Cost Comparison

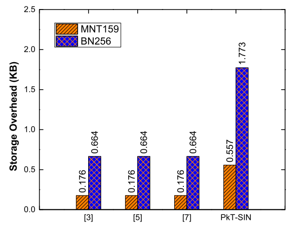


(a) Delay under MNT159

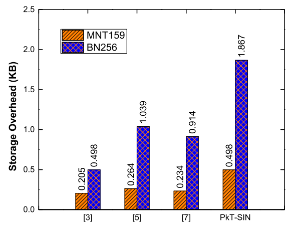


(b) Delay under BN256

Fig. 12: Total Authentication Delay Comparison



(a) Storage Cost of LEO



(b) Storage Cost of SU

Fig. 11: Storage Overhead Comparison

the computation/communication costs of TokGen are 46.242 ms/1.523 KB and 64.589 ms/5.898 KB for MNT159 and BN256, respectively. The calculation time for Verify is less than 123 ms.

Table VI and Figure 9 presents the performance of PkT-SIN in initialization, credential issue, session establishment and handover phases. During the initialization phase, the system setup, LEO setup, GS setup, and user setup algorithms collectively take less than 29 ms in computation time and entail communication costs below 8 KB. The computation during the credential issue phase is primarily dominated by the Issue algorithm of the PkT-KVAC scheme, which completes in under 406 ms with a communication cost of 3.47 KB.

The session establishment phase comprises three steps. In Step 1, the satellite user presents an authentication token to the LEO satellite by invoking PkT-KVAC.Show. In Step 2, LEO verifies the token using PkT-KVAC.Verify, computes symmetric keys using Diffie-Hellman shares, encrypts the messages, and transmits the ciphertexts to the SU and GS, respectively. In Step 3, SU and GS decrypt the ciphertexts to derive the secret session key individually. For $n = 50$, the session establishment phase incurs computation and communication overheads of 209.984 ms and 2.490 KB for MNT159, and 268.208 ms and 9.680 KB for BN256.

The handover phase encompasses three scenarios based on the speed of the satellite user and the handover entities. Table VI demonstrates that scenario 2 exhibits the highest efficiency during handover, requiring less than 11.72 ms. The performance of scenarios 1 and 3 is comparable, with resource usage not exceeding 81.03 ms and 4.68 KB.

In Fig. 10, 11, and 12, we present a comparison of computation costs, storage overheads and total authentication delay among PkT-SIN and schemes [3, 5, 7], under the curves MNT-159 and BN256. Fig. 10 compares the computation costs of LEO during session establishment. Owing to the avoidance of time-consuming pairing operations, PkT-SIN exhibits the best performance at both security levels, amounting to 1.859 s and 2.599 s for 100 authentications under MNT159 and BN256, respectively. Therefore, PkT-SIN offers an better solution for LEO devices with limited computational resources. Fig. 11 presents a comparison of storage overheads. In comparison to schemes [3, 5, 7], PkT-SIN has a higher storage cost, where the storage costs for LEO and SU are 1.773 KB and 1.867 KB respectively under the BN256 curve. Fig. 12 presents a comparison of the total authentication delay during session establishment, in which the signature propagation delay $T_{L,U}$ is set to 10 ms to align with the setting in scheme [3]. PkT-SIN requires a total delay of 5.818 s and 7.194 s to complete 100 session establishments under MNT159 and BN256, respectively. Along with scheme [5], PkT-SIN exhibits the lowest total authentication delay among the compared schemes.

In summary, comprehensive performance evaluations underscore the high efficiency of both PkT-KVAC and PkT-SIN.

VIII. Conclusion

In this paper, we proposed PkT-KVAC, a novel keyed-verification anonymous credential primitive. Leveraging PkT-KVAC, we developed PkT-SIN, a secure communication protocol for SIN. Compared to other secure communication protocols for SIN, PkT-SIN not only exhibits low computation costs but also eliminates redundant interactions and delays caused by reapplying access permissions. In terms of security, PkT-SIN is outstanding as it achieves a range of security properties, including secure key establishment, mutual authentication, anonymity, unlinkability, selective attribute disclosure, key forward/backward secrecy, accountability, and resistance to replay/impersonation/man-in-the-middle/DDoS attacks. A limitation of PkT-SIN is that it cannot resist post-quantum attacks. We leave it as future work to develop a post-quantum secure periodic k-times authentication scheme using lattice-based cryptography, code-based cryptography or multivariate cryptography.

Acknowledgment

Yang Yang and Robert Deng are supported by National Natural Science Foundation of China under Grant No. 62372110, Fujian Provincial Natural Science of Foundation under Grant 2023J02008, Lee Kong Chian Chair Professor Fund and AXA Research Fund. Yingjiu Li is supported by the Ripple University Blockchain Research Initiative. Guomin Yang is supported by the Singapore Ministry of Education (MOE) Academic Research Fund (AcRF) Tier 1 grant and the Lee Kong Chian Fellowship awarded by Singapore Management University. HweeHwa Pang is supported by Lee Kong Chian Chair Professor Fund.

References

[1] H. Guo, J. Li, J. Liu, N. Tian, and N. Kato, "A survey on space-air-ground-sea integrated network security in 6g," *IEEE CST*, vol. 24, no. 1, pp. 53–87, 2021.

[2] "Satellite Communication Market Size, Share & Trends Analysis Report By Component (Equipment, Services), By Application (Broadcasting, Airtime), By Vertical, By Region, And Segment Forecasts, 2022 - 2030," 2020, [Online]. Available: https://www.grandviewresearch.com/industry-analysis/satellite-communication-market.

[3] Q. Yang, K. Xue, J. Xu, J. Wang, F. Li, and N. Yu, "Anfra: Anonymous and fast roaming authentication for space information network," *IEEE TIFS*, vol. 14, no. 2, pp. 486–497, 2018.

[4] K. Xue, W. Meng, S. Li, D. S. Wei, H. Zhou, and N. Yu, "A secure and efficient access and handover authentication protocol for internet of things in space information networks," *IEEE IoTJ*, vol. 6, no. 3, pp. 5485–5499, 2019.

[5] X. Liu, A. Yang, C. Huang, Y. Li, T. Li, and M. Li, "Decentralized anonymous authentication with fair billing for space-ground integrated networks," *IEEE TVT*, vol. 70, no. 8, pp. 7764–7777, 2021.

[6] D. Liu, H. Wu, C. Huang, J. Ni, and X. Shen, "Blockchain-based credential management for anonymous authentication in sagvn," *IEEE JSAC*, 2022.

[7] D. Liu, H. Wu, J. Ni, and X. Shen, "Efficient and anonymous authentication with succinct multi-subscription credential in sagvn," *IEEE TITS*, vol. 23, no. 3, pp. 2863–2873, 2022.

[8] M. Chase, S. Meiklejohn, and G. Zaverucha, "Algebraic macs and keyed-verification anonymous credentials," in *CCS 2014*, 2014, pp. 1205–1216.

[9] M. Chase, T. Perrin, and G. Zaverucha, "The signal private group system and anonymous credentials supporting efficient verifiable encryption," in *CCS 2020*, 2020, pp. 1445–1459.

[10] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," in *CRYPTO 2004*. Springer, 2004, pp. 41–55.

[11] D. Pointcheval and O. Sanders, "Short randomizable signatures," in *CT-RSA 2016*. Springer, 2016, pp. 111–126.

[12] G. Fuchsbauer, C. Hanser, and D. Slamanig, "Structure-preserving signatures on equivalence classes and constant-size anonymous credentials," *JoC*, vol. 32, no. 2, pp. 498–546, 2019.

[13] S. Cakaj, "The parameters comparison of the "starlink" leo satellites constellation for different orbital shells," *FCN*, vol. 2, p. 643095, 2021.

[14] "KA-SAT Network cyber attack overview," 2022, [Online]. Available: https://news.viasat.com/blog/corporate/ka-sat-network-cyber-attack-overview.

[15] H. Cruickshank, "A security system for satellite networks," in *Fifth International Conference on Satellite Systems for Mobile Communications and Navigation, 1996.* IET, 1996, pp. 187–190.

[16] M.-S. Hwang, C.-C. Yang, and C.-Y. Shiu, "An authentication scheme for mobile satellite communication systems," *ACM SIGOPS Operating Systems Review*, vol. 37, no. 4, pp. 42–47, 2003.

[17] Y.-F. Chang and C.-C. Chang, "An efficient authentication protocol for mobile satellite communication systems," *ACM SIGOPS Operating Systems Review*, vol. 39, no. 1, pp. 70–84, 2005.

[18] M. Qi, J. Chen, and Y. Chen, "A secure authentication with key agreement scheme using ecc for satellite communication systems," *IJSCN*, vol. 37, no. 3, pp. 234–244, 2019.

[19] I. Altaf, M. A. Saleem, K. Mahmood, S. Kumari, P. Chaudhary, and C.-M. Chen, "A lightweight key agreement and authentication scheme for satellite-communication systems," *IEEE Access*, vol. 8, pp. 46 278–46 287, 2020.

[20] C. Poomagal and G. Sathish Kumar, "Ecc based lightweight secure message conveyance protocol for satellite communication in internet of vehicles (iov)," *WPC*, vol. 113, no. 2, pp. 1359–1377, 2020.

[21] D. Dharminder, P. K. Dadsena, P. Gupta, and S. Sankaran, "A post quantum secure construction of an authentication protocol for satellite communication," *IJSCN*, 2022.

[22] J. Lei, Z. Han, M. Á. Vázquez-Castro, and A. Hjorungnes, "Secure satellite communication systems design with individual secrecy rate constraints," *IEEE TIFS*, vol. 6, no. 3, pp. 661–671, 2011.

[23] M. Lin, Z. Lin, W.-P. Zhu, and J.-B. Wang, "Joint beamforming for secure communication in cognitive satellite terrestrial networks," *IEEE JSAC*, vol. 36, no. 5, pp. 1017–1029, 2018.

[24] Z. Lin, M. Lin, J. Ouyang, W.-P. Zhu, A. D. Panagopoulos, and M.-S. Alouini, "Robust secure beamforming for multibeam satellite communication systems," *IEEE TVT*, vol. 68, no. 6, pp. 6202–6206, 2019.

[25] G. Cui, Q. Zhu, L. Xu, and W. Wang, "Secure beamforming and jamming for multibeam satellite systems with correlated wiretap channels," *IEEE TVT*, vol. 69, no. 10, pp. 12 348–12 353, 2020.

[26] C. Guo, C. Gong, H. Xu, L. Zhang, and Z. Han, "A dynamic handover software-defined transmission control scheme in space-air-ground integrated networks," *IEEE TWC*, 2022.

[27] Y. Fan, G. Wu, K.-C. Li, and A. Castiglione, "Robust end hopping for secure satellite communication in moving target defense," *IEEE IoTJ*, 2022.

[28] I. Teranishi, J. Furukawa, and K. Sako, "K-times anonymous authentication," in *CRYPTO 2004*. Springer, 2004, pp. 308–322.

[29] L. Nguyen and R. Safavi-Naini, "Dynamic k-times anonymous authentication," in *ACNS 2005*. Springer, 2005, pp. 318–333.

[30] U. Chaterjee, D. Mukhopadhyay, and R. S. Chakraborty, "3paa: A private puf protocol for anonymous authentication," *IEEE TIFS*, vol. 16, pp. 756–769, 2020.

[31] J. Huang, W. Susilo, F. Guo, G. Wu, Z. Zhao, and Q. Huang, "An anonymous authentication system for pay-as-you-go cloud computing," *IEEE TDSC*, vol. 19, no. 2, pp. 1280–1291, 2020.

[32] J. Camenisch, S. Hohenberger, M. Kohlweiss, A. Lysyanskaya, and M. Meyerovich, "How to win the clonewars: efficient periodic n-times anonymous authentication," in *CCS 2006*, 2006, pp. 201–210.

[33] B. Lian, G. Chen, M. Ma, and J. Li, "Periodic k-times anonymous authentication with efficient revocation of violator's credential," *IEEE TIFS*, vol. 10, no. 3, pp. 543–557, 2014.

[34] M. Chase and A. Lysyanskaya, "On signatures of knowledge," in *CRYPTO 2006*. Springer, 2006, pp. 78–96.

[35] "Miracl: Multiprecision integer and rational arithmetic c/c++ library," *https://github.com/miracl/MIRACL*, [Accessed: 01-Jun-2021].

[36] C. P. Schnorr, "Efficient signature generation by smart cards," *JoC*, vol. 4, pp. 161–174, 1991.

[37] G. Couteau and M. Reichle, "Non-interactive keyed-verification anonymous credentials," in *PKC 2019*. Springer, 2019, pp. 66–96.

[38] M. Wazid, A. K. Das, and S. Shetty, "An authentication and key management framework for secure and intelligent transportation of internet of space things," *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[39] Y. Wang, W. Zhang, X. Wang, M. K. Khan, and P. Fan, "Security enhanced authentication protocol for space-ground integrated railway networks," *IEEE Transactions on Intelligent Transportation Systems*, 2023.

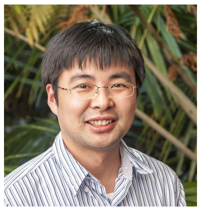
[40] R. Ma, J. Cao, D. Feng, and H. Li, "Laa: lattice-based access authentication scheme for iot in space information networks," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 2791–2805, 2019.

[41] J. Guo, Y. Du, X. Wu, and M. Li, "An anti-quantum authentication protocol for space information networks based on ring learning with errors," *Journal of Communications and Information Networks*, vol. 6, no. 3, pp. 301–311, 2021.

Yang Yang received the B.Sc. degree from Xidian University, Xi'an, China, in 2006 and Ph.D. degrees from Xidian University, China, in 2011. She is a full professor with College of Computer Science and Big Data, Fuzhou University. She is also a senior research scientist with School of Computing and Information System, Singapore Management University. Her research interests are in the area of information security and privacy protection. She has published more than 60 papers in TIFS, TDSC, TSC, TCC, TII, etc. She is Senior Member of IEEE.

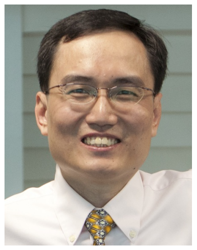
Wenyi Xue received the B.Sc. degree from Fujian University, Fuzhou, China, in 2020. Now, he is pursing Ph.D degree under supervision of Prof. Yang Yang in College of Computer Science and Big Data, Fuzhou University, Fuzhou, China. His research interests are in the area of privacy protection and zero-knowledge proof.

Jianfei Sun received his Ph.D. degree from the University of Electronic Science and Technology of China (UESTC). He is currently a research fellow at the School of Computer Science and Engineering, Nanyang Technological University. His research interests include network security and IoT security. He has published many papers on IEEE TDSC, IEEE TIFS, IEEE TII, IEEE TCC, IEEE TVT, IEEE IoTJ, Inf. Sci, IEEE Systems, etc.



Guomin Yang is an Associate Professor at the School of Computing and Information Systems, Singapore Management University. He completed PhD in Computer Science from City University of Hong Kong in 2009. He has been an Associate Professor at the University of Wollongong, Australia. His research interests are applied cryptography and privacy-enhancing technologies. He was a program co-chair of ACISP 2018 and ACISP 2022.

Yingjiu Li obtained his PhD degree from George Mason University in 2003. He had been a faculty member at Singapore Management University from 2003 to 2019. Now, He is Ripple Professor with Department of Computer and Information Science, University of Oregon. His research interests include IoT security and privacy, mobile security, and data security and privacy. He has published more than 130 papers in Cybersecurity, and co-authored two academic books.



Hwee Hwa Pang received the BSc (first class honors) and MS degrees from the National University of Singapore, in 1989 and 1991, respectively, and the PhD degree from the University of Wisconsin-Madison, in 1994, all in computer science. He is a professor with School of Computing and Information Systems, Singapore Management University. His current research interests include database management systems, data security, and information retrieval.

Robert H. Deng is AXA Chair Professor of Cybersecurity in the School of Computing and Information Systems, Singapore Management University. His research interests include data security, network and system security. He has served/is serving on the editorial boards of many international journals in security, such as IEEE Transactions on Information Forensics and Security, IEEE Transactions on Dependable and Secure Computing, etc. He is Fellow of IEEE.

Supplemental Material

A. PkT-KVAC: Security Models

Following the definitions in [8, 32, 37], we define *correctness*, *unforgeability*, *anonymity*, *unlinkability*, *k-detectability* and *exculpability* for PkT-KVAC. The lists in the security models are given in Table VII.

Definition A.1 **(Correctness)**. Let $\mathcal{D}$ be the universe of user identity, and Ω be the universe of attribute sets. Then a PkT-KVAC scheme is *correct* for $\mathcal{D}, \Omega$ if all $ID \in \mathcal{D}$, all $\mathsf{ATTR} \subseteq \Omega$, for all security parameter λ,

$$\Pr\left[\begin{array}{l}
\mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda, 1^n), (\mathsf{ipk},\mathsf{isk}) \xleftarrow{\$} \mathsf{IKeyGen}(\mathsf{pp}) \\
(\mathsf{upk},\mathsf{usk}) \xleftarrow{\$} \mathsf{UKeyGen}(\mathsf{pp}, ID) \\
(\mathsf{cred}_{TP,k}, \mathcal{D}) \xleftarrow{\$} \\
\qquad \mathsf{Issue}(\mathcal{U}(\mathsf{upk},\mathsf{usk},\mathsf{ATTR}) \leftrightarrow \mathcal{I}(\mathsf{isk}, TP, k)) \\
(\mathsf{tok},\mathsf{TIN},\mathcal{D}') \xleftarrow{\$} \\
\qquad \mathsf{Show}(\mathsf{upk},\mathsf{usk},\mathsf{cred}_{TP,k},\mathsf{ATTR}_D,\mathcal{D},M) \\
(\mathsf{tok}',\mathsf{TIN},\mathcal{D}') \xleftarrow{\$} \\
\qquad \mathsf{Show}(\mathsf{upk},\mathsf{usk},\mathsf{cred}_{TP,k},\mathsf{ATTR}_D,\mathcal{D},M) \\
\mathsf{Verify}(\mathsf{isk},\mathsf{tok},\mathsf{TIN},\mathsf{ATTR}_D,M) \to 1 \\
\mathsf{Verify}(\mathsf{isk},\mathsf{tok}',\mathsf{TIN},\mathsf{ATTR}_D,M) \to 1 : \\
\mathsf{Reveal}(\mathsf{TIN},\mathsf{tok},\mathsf{tok}') \to \bot
\end{array}\right]$$

$\le \nu(\lambda)$, where ν is a negligible function.

Definition A.2 **(Unforgeability)**. A PkT-KVAC scheme satisfies *unforgeability* if for any PPT adversary $\mathcal{A}$, there exists a negligible function ν such that $\mathsf{Adv}^{\mathsf{unforge}}_{\mathsf{PkT\text{-}KVAC}}(\lambda) \overset{\mathsf{def}}{=}$

$$\Pr\left[b=1 \left| \begin{array}{l}
\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^n), \\
(\mathsf{ipk},\mathsf{isk}) \leftarrow \mathsf{IKeyGen}(\mathsf{pp}) \\
(ID^*, \mathsf{tok}^*, \mathsf{TIN}^*, \mathsf{ATTR}^*_D, M^*) \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(\mathsf{pp},\mathsf{ipk}) \\
b \leftarrow \mathsf{Verify}(\mathsf{isk},\mathsf{tok}^*,\mathsf{TIN}^*,\mathsf{ATTR}^*_D,M^*) \\
\text{return } b \text{ if } (\mathsf{tok}^*,\mathsf{TIN}^*,\mathsf{ATTR}^*_D,M^*) \notin \mathcal{L}_{\mathsf{show}} \\
\qquad\qquad\qquad\qquad \wedge ID^* \notin \mathcal{L}_{\mathsf{corrupt}} \\
\text{else abort}
\end{array}\right.\right]$$

$\le \nu(\lambda)$, where the oracle set $\mathcal{O} = \{\mathsf{UKeyGen}, \mathsf{Issue}, \mathsf{Show}, \mathsf{Verify}, \mathsf{Reveal}, \mathsf{Corrupt}\}$ is implemented by $\mathsf{UKeyGen}(\mathsf{pp})$, $\mathsf{Issue}(\mathsf{isk},\mathsf{ATTR},TP,k,\cdot)$, $\mathsf{Show}(\mathsf{usk},\mathsf{cred},\mathsf{ATTR}_D,\cdot)$, $\mathsf{Verify}(\mathsf{isk},\cdot)$, $\mathsf{Reveal}(\cdot)$ and $\mathsf{Corrupt}(\cdot)$.

In the following, we define a security model for "***unlinkability***" of PkT-KVAC, with the understanding that the model inherently encompasses the notion of "***anonymity***" within it.

Definition A.3 **(Unlinkability)**. A PkT-KVAC scheme satisfies *unlinkability* if for any PPT adversary $\mathcal{A}$, there exists a negligible function ν such that $\mathsf{Adv}^{\mathsf{unlink}}_{\mathsf{PkT\text{-}KVAC}}(\lambda) \overset{\mathsf{def}}{=}$

$$\Pr\left[b'=b \left| \begin{array}{l}
\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^n), \\
(\mathsf{ipk},\mathsf{isk}) \leftarrow \mathsf{IKeyGen}(\mathsf{pp}) \\
((ID^*_0, \mathsf{ATTR}^*_0), (ID^*_1, \mathsf{ATTR}^*_1), \mathsf{ATTR}^*_D, TP^*, \\
\qquad k^*, \mathcal{D}^*, M^*) \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(\mathsf{pp},\mathsf{ipk},\mathsf{isk}) \\
\text{abort if } \mathsf{ATTR}^*_D \nsubseteq \mathsf{ATTR}^*_{0/1} \\
b \xleftarrow{\$} \{0,1\} \\
(\mathsf{tok}^*, \mathsf{TIN}^*, \mathcal{D}'^*) \leftarrow \\
\qquad \mathsf{Show}(\mathsf{upk}^*_b, \mathsf{usk}^*_b, \mathsf{cred}^*_b, \mathsf{ATTR}^*_D, \mathcal{D}^*, M^*) \\
b' \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(\mathsf{pp},\mathsf{tok}^*,\mathsf{TIN}^*) \\
\text{return } b' \text{ if} \\
\qquad (\mathsf{cred}^*_{0/1}, \mathsf{ATTR}^*_{0/1}, TP^*, k^*, \mathcal{D}^*) \in \mathcal{L}_{\mathsf{issue}} \\
\qquad\qquad \wedge \{ID^*_0, ID^*_1\} \notin \mathcal{L}_{\mathsf{corrupt}} \\
\text{else abort}
\end{array}\right.\right]$$

$\le \nu(\lambda)$, where the oracle set $\mathcal{O} = \{\mathsf{UKeyGen}, \mathsf{Issue}, \mathsf{Show}, \mathsf{Verify}, \mathsf{Reveal}, \mathsf{Corrupt}\}$ is implemented by $\mathsf{UKeyGen}(\mathsf{pp})$, $\mathsf{Issue}(\mathsf{isk},\mathsf{ATTR},TP,k,\cdot)$, $\mathsf{Show}(\mathsf{usk},\mathsf{cred},\mathsf{ATTR}_D,\cdot)$, $\mathsf{Verify}(\mathsf{isk},\cdot)$, $\mathsf{Reveal}(\cdot)$ and $\mathsf{Corrupt}(\cdot)$.

In the following security model of "k-**Detectability**", an adversary is allowed to collude with n users. If the adversary succeeds in being accepted by some verifier in more than kn authentications, the adversary wins. Here, k is the number of times the verifier allows access for each user, n is the number of users who collude with the adversary, and the kn token series numbers are sent to $\mathcal{L}_{k\text{-}\mathsf{show}}$ before output phase.

Definition A.4 **(k-Detectability)**. A PkT-KVAC scheme satisfies *k-detectability* if for any PPT adversary $\mathcal{A}$, there exists a negligible function ν such that $\mathsf{Adv}^{k\text{-}\mathsf{detect}}_{\mathsf{PkT\text{-}KVAC}}(\lambda) \overset{\mathsf{def}}{=}$

$$\Pr\left[b=1 \left| \begin{array}{l}
\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^n), \\
(\mathsf{ipk},\mathsf{isk}) \leftarrow \mathsf{IKeyGen}(\mathsf{pp}) \\
(\mathsf{tok}^*, \mathsf{TIN}^*, \mathsf{ATTR}^*_D, M^*) \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(\mathsf{pp},\mathsf{ipk}) \\
b \leftarrow \mathsf{Verify}(\mathsf{isk},\mathsf{tok}^*,\mathsf{TIN}^*,\mathsf{ATTR}^*_D,M^*) \\
\text{return } b \text{ if } \mathsf{TIN}^* \notin \mathcal{L}_{k\text{-}\mathsf{show}} \\
\text{else abort}
\end{array}\right.\right]$$

$\le \nu(\lambda)$, where the oracle set $\mathcal{O} = \{\mathsf{UKeyGen}, \mathsf{Issue}, \mathsf{Show}, \mathsf{Verify}, \mathsf{Reveal}\}$ is implemented by $\mathsf{UKeyGen}(\mathsf{pp})$, $\mathsf{Issue}(\mathsf{isk},\mathsf{ATTR},TP,k,\cdot)$, $\mathsf{Show}(\mathsf{usk},\mathsf{cred},\mathsf{ATTR}_D,TP,\cdot)$, $\mathsf{Verify}(\mathsf{isk},\cdot)$ and $\mathsf{Reveal}(\cdot)$.

Definition A.5 **(Exculpability)**. A PkT-KVAC scheme satisfies *exculpability* if for any PPT adversary $\mathcal{A}$, there exists a negligible function ν such that $\mathsf{Adv}^{\mathsf{exculpate}}_{\mathsf{PkT\text{-}KVAC}}(\lambda) \overset{\mathsf{def}}{=}$

$$\Pr\left[b=1 \left| \begin{array}{l}
\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^n), \\
(\mathsf{ipk},\mathsf{isk}) \leftarrow \mathsf{IKeyGen}(\mathsf{pp}) \\
(\mathsf{ATTR}^*_D, M^*, \mathsf{TIN}^*, \mathsf{tok}^*, \widetilde{\mathsf{tok}}^*) \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(\mathsf{pp},\mathsf{ipk},\mathsf{isk}) \\
(\mathsf{upk}^*, ID^*) \leftarrow \mathsf{Reveal}(\mathsf{TIN}^*, \mathsf{tok}^*, \widetilde{\mathsf{tok}}^*) \\
\text{return } 1 \text{ if } ID^* \notin \mathcal{L}_{\mathsf{corrupt}} \\
\qquad \wedge \mathsf{Verify}(\mathsf{isk},\mathsf{tok}^*,\mathsf{TIN}^*,\mathsf{ATTR}^*_D,M^*) \to 1 \\
\qquad \wedge \mathsf{Verify}(\mathsf{isk},\widetilde{\mathsf{tok}}^*,\mathsf{TIN}^*,\mathsf{ATTR}^*_D,M^*) \to 1 \\
\text{else abort}
\end{array}\right.\right]$$

$\leq \nu(\lambda)$, where the oracle set $\mathcal{O} = \{\mathsf{UKeyGen}, \mathsf{Issue}, \mathsf{Show}, \mathsf{Verify}, \mathsf{Reveal}, \mathsf{Corrupt}\}$ is implemented by $\mathsf{UKeyGen}(\mathsf{pp})$, $\mathsf{Issue}(\mathsf{isk}, \mathsf{ATTR}, TP, k, \cdot)$, $\mathsf{Show}(\mathsf{usk}, \mathsf{cred}, \mathsf{ATTR}_D, \cdot)$, $\mathsf{Verify}(\mathsf{isk}, \cdot)$, $\mathsf{Reveal}(\cdot)$ and $\mathsf{Corrupt}(\cdot)$.

List	Description
$\mathcal{L}_{\mathsf{issue}}$	credentials that have been issued
$\mathcal{L}_{\mathsf{show}}$	tokens that have been shown
$\mathcal{L}_{k\text{-}\mathsf{show}}$	each issued credentials are shown k times
$\mathcal{L}_{\mathsf{honest}}$	users that are honest
$\mathcal{L}_{\mathsf{corrupt}}$	users that are corrupted

TABLE VII: Lists in Security Experiments of PkT-KVAC

B. PkT-KVAC: Instantiation of SoK

The signature of knowledges (SoKs) Π_U^1, Π_U^2, Π_I are be instantiated as follows.

•Π_U^1: (1) $\mathcal{U}$ selects $\hat{x}_u, \hat{s}' \xleftarrow{\$} \mathbb{Z}_p^*$, computes $\hat{Y}_u = g_0^{\hat{x}_u}, \widehat{\mathsf{Cm}} = Y_1^{\hat{x}_u} Y_2^{\hat{s}'}, c = H_1(Y_u || \hat{Y}_u || \mathsf{Cm} || \widehat{\mathsf{Cm}} || \mathsf{ATTR})$, $\overline{x}_u = \hat{x}_u - c \cdot x_u, \overline{s}' = \hat{s}' - c \cdot s'$, sends $(\hat{Y}_u, \widehat{\mathsf{Cm}}, \overline{x}_u, \overline{s}')$ to $\mathcal{I}$. (2) $\mathcal{I}$ recovers $c = H_1(Y_u || \hat{Y}_u || \mathsf{Cm} || \widehat{\mathsf{Cm}} || \mathsf{ATTR})$ and checks $\hat{Y}_u \stackrel{?}{=} Y_u^c g_0^{\overline{x}_u}, \widehat{\mathsf{Cm}} \stackrel{?}{=} (\mathsf{Cm})^c Y_1^{\overline{x}_u} Y_2^{\overline{s}'}$.

•Π_U^2: (1) $\mathcal{U}$ selects $\hat{x}_u, \hat{s}, \hat{t}, \hat{r}, \hat{w}, \hat{\beta}, \hat{\delta} \xleftarrow{\$} \mathbb{Z}_p^*$, computes $\Lambda_0 = g_0^s h_1^r$, $\Lambda_1 = (ZY_1Y_2)^{\hat{r}} \cdot h_1^{\hat{x}_u} \cdot h_2^{\hat{s}}$, $\Lambda_2 = T_u^{\hat{s}}$, $\Lambda_3 = (\Lambda_0 \cdot g_0^{\alpha_1})^{\hat{\beta}} h_1^{\hat{\delta}}$, $\Lambda_4 = g_0^w$, $\hat{\Lambda}_4 = g_0^{\hat{w}}$, $\Lambda_5 = g_0^{\hat{x}_u} \cdot (\Lambda_4)^{\hat{\beta}}$, $c = H_1(J_u || C_u || D_u || E_u || F_u || \Lambda_0 || \Lambda_1 || \Lambda_2 || \Lambda_3 || \Lambda_4 || \hat{\Lambda}_4 || \Lambda_5 || \mathsf{ATTR}_D || TP || J_u)$, $\overline{x}_u = \hat{x}_u - c \cdot x_u$, $\overline{s} = \hat{s} - c \cdot s$, $\overline{t} = \hat{t} - c \cdot t$, $\overline{r} = \hat{r} - c \cdot r$, $\overline{w} = \hat{w} - c \cdot w$, $\overline{\beta} = \hat{\beta} - c \cdot \beta$, $\overline{\delta} = \hat{\delta} - c \cdot \delta$, sends $(\Lambda_0, \Lambda_1, \Lambda_2, \Lambda_3, \Lambda_4, \hat{\Lambda}_4, \Lambda_5, \overline{x}_u, \overline{s}, \overline{t}, \overline{r}, \overline{w}, \overline{\beta}, \overline{\delta})$ to $\mathcal{I}$. (2) $\mathcal{I}$ recovers c and checks $\Lambda_1 \stackrel{?}{=} (D_1 D_2 C_4)^c (ZY_1Y_2)^{\overline{r}} \cdot h_1^{\overline{x}_u} \cdot h_2^{\overline{s}}$, $\Lambda_2 \stackrel{?}{=} (g_0 / T_u^{\alpha_0})^c T_u^{\overline{s}}$, $\Lambda_3 \stackrel{?}{=} g_0^c (F'_u g_0^{\alpha_1})^{\overline{\beta}} h_1^{\overline{\delta}}$, $\hat{\Lambda}_4 = \Lambda_4^c g_0^{\overline{w}}$, $\Lambda_5 = F_u^c g_0^{\overline{x}_u} \cdot (\Lambda_4)^{\overline{\beta}}$.

•Π_I: (1) $\mathcal{I}$ selects $\{\hat{x}_i\}_{i \in [1,3]}, \hat{y}, \hat{y}_3, \{\hat{z}_i\}_{i \in [0,n]} \xleftarrow{\$} \mathbb{Z}_p^*$, computes $\hat{X} = g_1^{\hat{x}_1} h_3^{\hat{y}_3}$, $\Lambda_6 = g_2^{\hat{x}_2} g_3^{\hat{x}_3} (h_1 h_2)^{\hat{y}} \prod_{i=1}^n u_i^{\hat{z}_i}$,

$$\hat{V} = g_1^{\hat{x}_1} U^{\hat{x}_2} (U^t)^{\hat{x}_3} (\mathsf{Cm} \cdot Y_2^{s''})^{\hat{y}_3} \cdot (\widetilde{u}_0^{H_1(TP,k)})^{\hat{z}_0} \cdot \prod_{i=1}^{n} (\widetilde{u}_i^{H_1(\mathsf{attr}_i)})^{\hat{z}_i},$$

$c = H_1(X || Z || V || \hat{X} || \Lambda_6 || \hat{V} || s'' || t || U)$, $\overline{x}_i = \hat{x}_i - c \cdot x_i$, $\overline{y} = \hat{y} - c \cdot y$, $\overline{y}_3 = \hat{y}_3 - c \cdot y_3$, $\overline{z}_i = \hat{z}_i - c \cdot z_i$, sends $(\hat{X}, \Lambda_6, \hat{V}, \{\overline{x}_i\}_{i \in [1,3]}, \overline{y}, \overline{y}_3, \{\overline{z}_i\}_{i \in [0,n]})$ to $\mathcal{U}$. (2) $\mathcal{U}$ recovers c and checks $\hat{X} \stackrel{?}{=} X^c g_1^{\overline{x}_1} h_3^{\overline{y}_3}$, $\Lambda_6 \stackrel{?}{=} (g_0/Z)^c g_2^{\overline{x}_2} g_3^{\overline{x}_3} (h_1 h_2)^{\overline{y}} \prod_{i=1}^n u_i^{\overline{z}_i}$, $\hat{V} \stackrel{?}{=} V^c g_1^{\overline{x}_1} U^{\overline{x}_2} (U^t)^{\overline{x}_3} (\mathsf{Cm} \cdot Y_2^{s''})^{\overline{y}_3} \cdot (\widetilde{u}_0^{H_1(TP,k)})^{\overline{z}_0} \cdot \prod_{i=1}^n (\widetilde{u}_i^{H_1(\mathsf{attr}_i)})^{\overline{z}_i}$.

C. PkT-KVAC: Security Proof

C.1. Correctness Proof

The correctness proof of PkT-KVAC consists of the following aspects: 1) the Verify algorithm holds if the authentication token is valid; 2) the Reveal algorithm is able to disclose the identity of a dishonest user who reuses tokens with the same token identifier number.

1) The Verify algorithm is correct as the following equation holds:

$$\begin{aligned}\Gamma =& (E_0 \cdot \widetilde{u}_0^{H_1(TP,k)})^{z_0} \prod_{\mathsf{attr}_i \in \mathsf{ATTR}_D} (E_i \cdot \widetilde{u}_i^{H_1(\mathsf{attr}_i)})^{z_i} \prod_{\mathsf{attr}_i \notin \mathsf{ATTR}_D} E_i^{z_i} \\ =& u_0^{r \cdot z_0} \cdot \widetilde{u}_0^{H_1(TP,k) z_0} \cdot \prod_{i=1}^{n} u_i^{r \cdot z_i} \widetilde{u}_i^{H_1(\mathsf{attr}_i) z_i}.\end{aligned}$$

Then, we have

$$\begin{aligned}& g_1^{x_1} C_2^{x_2} C_3^{x_3} C_4 \cdot (D_1^{y_1 y_3} D_2^{y_2 y_3}) \cdot \Gamma \\ =& g_1^{x_1} (g_2^r U)^{x_2} (g_3^r U^t)^{x_3} Z^r \cdot ((Y_2^r \cdot h_1^{x_u})^{y_1 y_3} (Y_1^r \cdot h_2^s)^{y_2 y_3}) \cdot \Gamma \\ =& Z^r \cdot (g_2^{x_2})^r (g_3^{x_3})^r (Y_2^{y_1 y_3} Y_1^{y_2 y_3})^r (\prod_{i=0}^{n} u_i^{z_i})^r \\ & \cdot g_1^{x_1} U^{x_2} (U^t)^{x_3} (Y_1^{x_u} Y_2^s)^{y_3} \cdot \widetilde{u}_0^{H_1(TP,k) z_0} \cdot \prod_{i=1}^{n} \widetilde{u}_i^{H_1(\mathsf{attr}_i) z_i} \\ =& g_0^r \cdot g_1^{x_1} U^{x_2} (U^t)^{x_3} (Y_1^{x_u} Y_2^s)^{y_3} \cdot \widetilde{u}_0^{H_1(TP,k) z_0} \cdot \prod_{i=1}^{n} \widetilde{u}_i^{H_1(\mathsf{attr}_i) z_i} \\ =& g_0^r V = C_1.\end{aligned}$$

2) The Reveal algorithm is correct as the following equation holds:

$$F_u / (F^w) = Y_u \cdot g_0^{w/(s+\alpha_1)} / (F^w) = Y_u.$$

The computation of Π_U^2 is correct as the following equations hold.

$$\begin{aligned}(ZY_1Y_2)^r \cdot h_1^{x_u} \cdot h_2^s &= (Y_2^r \cdot h_1^{x_u}) \cdot (Y_1^r \cdot h_2^s) \cdot Z^r \\ &= D_1 D_2 C_4, \\ T_u^{(s+\alpha_0)} &= (g_0^{1/(s+\alpha_0)})^{(s+\alpha_0)} = g, \\ (F'_u g_0^{\alpha_1})^\beta h_1^\delta &= (g_0^s h_1^r g_0^{\alpha_1})^\beta h_1^\delta \\ &= g \cdot h_1^{r/(s+\alpha_1)} h_1^\delta = g, \\ g_0^{x_u} \cdot (g_0^w)^\beta &= Y_u \cdot g_0^{w/(s+\alpha_1)} = F_u.\end{aligned}$$

C.2. Unforgeability Proof

Theorem 4.1. The PkT-KVAC scheme is *unforgeable* if the LRSW assumption holds.

Proof. Given a tuple $\mathcal{T} = (g, \widetilde{A} = g^a, \widetilde{B} = g^b)$ and a LRSW oracle $\mathcal{O}_{LRSW}(\cdot)$ which inputs $m \in \mathbb{Z}_p^*$ and outputs (h, h^b, h^{a+mab}). The challenger $\mathcal{C}$ samples $g_1, \{h_i\}_{i \in [2,3]}, \{u_i, \widetilde{u}_i\}_{i \in [0,n]} \xleftarrow{\$} \mathbb{G}$ and $\alpha, \beta, \gamma \xleftarrow{\$} \mathbb{Z}_p^*$, sets $g_0 = g$, $g_2 = g_0^\alpha$, $g_3 = g_0^\beta$, $h_1 = g_0^\gamma$. The tuple $(\mathbb{G}, p, \{g_i\}_{i \in [0,3]}, \{h_i\}_{i \in [1,3]}, \{u_i, \widetilde{u}_i\}_{i \in [0,n]}, H_1, f)$ is output as the public parameter pp. Besides, $\mathcal{C}$ initailizes empty lists $\mathcal{L}_{\mathsf{issue}}$, $\mathcal{L}_{\mathsf{show}}$, $\mathcal{L}_{\mathsf{honest}}$, $\mathcal{L}_{\mathsf{corrupt}}$, and constructs an extractor $\mathcal{E}$ and a simulator $\mathcal{S}$ for the zero-knowledge proof system.

Issuer key generation phase: $\mathcal{C}$ selects $x_1, y_1, y_2, y_3, z_0, \cdots, z_n \xleftarrow{\$} \mathbb{Z}_p^*$, and computes $X = g_1^{x_1} h_3^{y_3}$, $Y_1 = h_1^{y_1}$, $Y_2 = h_2^{y_2}$. $\mathcal{C}$ simulates the element Z by computing $Z =$

$g_0/(\widetilde{A}^{\alpha}(\widetilde{A}\widetilde{B})^{\beta} \cdot (h_1h_2)^{y_1y_2y_3} \cdot \prod_{i=0}^{n} u_i^{z_i}) = g_0/(g_2^a g_3^{ab} \cdot (h_1h_2)^{y_1y_2y_3} \cdot \prod_{i=0}^{n} u_i^{z_i})$ and outputs $\mathsf{ipk} = (X, Y_1, Y_2, Z)$. Note that the equation implies that $\mathcal{C}$ learns noting about the issuer secret key $x_2 = a$, $x_3 = ab$.

Oracle query phase: $\mathcal{C}$ answers the oracle queries $\mathcal{O} = \{\mathsf{UKeyGen}, \mathsf{Issue}, \mathsf{Show}, \mathsf{Verify}, \mathsf{Reveal}\}$ as follows.

•UKeyGen: $\mathcal{C}$ selects $t \xleftarrow{\$} \mathbb{Z}_p^*$ and queries $\mathcal{O}_{LRSW}(t) \rightarrow (U, U_1 = U^b, U_2 = U^{a+tab})$. $\mathcal{C}$ sets $\mathsf{upk} = U_1$, which indicates that $\mathsf{upk} = g_0^{x_u} = U^b$. Then, $\mathcal{C}$ inserts (ID, t, U, U_1, U_2) to $\mathcal{L}_{\mathsf{honest}}$.

•Issue: If ID has not been queried beforehand and the entry identified by ID exists in $\mathcal{L}_{\mathsf{honest}}$, $\mathcal{C}$ selects $s \xleftarrow{\$} \mathbb{Z}_p^*$, computes $V = g_1^{x_1} \cdot U_2 \cdot ((U_1)^{\gamma y_1} \cdot Y_2^s)^{y_3} \cdot \widetilde{u}_0^{H_1(TP,k)z_0} \prod_{i=1}^{n} \widetilde{u}_i^{H_1(\mathsf{attr}_i)z_i} = g_1^{x_1} \cdot U^{a+tab} \cdot ((h_1^{y_1 \cdot x_u}) \cdot Y_2^s)^{y_3} \cdot \widetilde{u}_0^{H_1(TP,k)z_0} \prod_{i=1}^{n} \widetilde{u}_i^{H_1(\mathsf{attr}_i)z_i}$, where $h_1^{x_u} = g_0^{\gamma \cdot x_u} = (g_0^{x_u})^{\gamma} = U_1^{\gamma}$. $\mathcal{C}$ initialize a dispenser $\mathcal{D} = \{1, 2, \cdots, k\}$, sets $\mathsf{cred}_{TP,k} = (s, t, U, V, TP, k)$. The tuple $(ID, \mathsf{usk}, \mathsf{cred}_{TP,k}, \mathsf{ATTR}, TP, k, \mathcal{D})$ is inserted to $\mathcal{L}_{\mathsf{issue}}$ and $(\mathsf{cred}_{TP,k}, \mathcal{D})$ is returned to $\mathcal{A}$.

•Show: $\mathcal{C}$ retrieves $(ID, \mathsf{usk}, \mathsf{cred}_{TP,k}, \mathsf{ATTR}, TP, k, \mathcal{D})$ from $\mathcal{L}_{\mathsf{issue}}$ and aborts if such tuple does not exist. If $\mathcal{D}$ is not empty, $\mathcal{C}$ selects J_u from the dispenser $\mathcal{D}$ and runs algorithm Show to compute $(C_u, D_u, E_u, T_u, F_u)$, where $D_1 = Y_2^r \cdot U_1^{\gamma} = Y_2^r \cdot h_1^{x_u}$. Then, $\mathcal{C}$ runs the simulator $\mathcal{S}$ to produce Π_U^2, returns $(\mathsf{TIN} = T_u, \mathsf{tok} = (J_u, C_u, D_u, E_u, F_u, \Pi_U^2, TP, k))$ and inserts $(ATTR_D, M, \mathsf{tok}, \mathsf{TIN})$ to $\mathcal{L}_{\mathsf{show}}$. Besides, $\mathcal{C}$ updates $\mathcal{L}_{\mathsf{issue}}$ with the updated dispenser $\mathcal{D}' = \mathcal{D} \backslash \{J_u\}$.

•Verify: The oracle query aborts if the entry identified by TIN does not exist in $\mathcal{L}_{\mathsf{show}}$. $\mathcal{C}$ return 1 if the tuple $(\mathsf{tok}, \mathsf{TIN}, \mathsf{ATTR}_D, M) \in \mathcal{L}_{\mathsf{show}}$ and returns 0 otherwise.

•Corrupt: $\mathcal{C}$ removes the tuple (ID, t, U, U_1, U_2) from $\mathcal{L}_{\mathsf{honest}}$. Then, $\mathcal{C}$ randomly picks $\delta \xleftarrow{\$} \mathbb{Z}_p^*$, inserts $(ID, \mathsf{upk} = U_1, \delta)$ to $\mathcal{L}_{\mathsf{corrupt}}$ and returns $\mathsf{usk} = \delta$ to answer the query.

•Reveal: $\mathcal{C}$ performs the oracle query by calling the Reveal algorithm directly.

Forge phase: $\mathcal{A}$ outputs a forged tuple $(ATTR_D^*, M^*, \mathsf{tok}^*, \mathsf{TIN}^*)$, where $tok^* = (J_u^*, C_u^*, D_u^*, E_u^*, F_u^*, (\Pi_U^2)^*)$. $\mathcal{C}$ aborts if $(ATTR_D^*, M^*, \mathsf{tok}^*, \mathsf{TIN}^*) \in \mathcal{L}_{\mathsf{show}} \vee (\Pi_U^2)^*$ is invalid. If tok^* is a well-formed token, we have

$$\begin{aligned} C_2^* =& g_2^{r^*} U^*, \\ C_1^* =& g_0^{r^*} V^* = g_0^{r^*} g_1^{x_1} (U^*)^{a+t^*ab} (Y_1^{x_u^*} Y_2^{s^*})^{y_3} \cdot \\ & \widetilde{u}_0^{H_1(TP,k)z_0} \cdot \prod_{i=1}^{n} \widetilde{u}_i^{H_1(\mathsf{attr}_i^*)z_i}, \end{aligned}$$

where $Y_1^{x_u^*} = g_0^{\gamma y_1 \cdot x_u^*} = (U^*)^{b\gamma y_1} = (U_1^*)^{\gamma y_1}$. Here, we implicitly set $U_1^* = (U^*)^b$. Then, $\mathcal{C}$ runs the extractor $\mathcal{E}$ to extract r^*, t^*, s^*, x_u^*, computes $U^* = C_2^*/g_2^{r^*}$, $U_1^* = g_0^{x_u^*}$,

$$\begin{aligned} U_2^* =& C_1^*/(g_0^{r^*} g_1^{x_1} (Y_1^{x_u^*} Y_2^{s^*})^{y_3} \cdot \widetilde{u}_0^{H_1(TP,k)z_0} \prod_{i=1}^{n} \widetilde{u}_i^{H_1(\mathsf{attr}_i^*)z_i}) \\ =& (U^*)^{a+t^*ab}. \end{aligned}$$

If t^* is not queried to $\mathcal{O}_{LRSW}(\cdot)$. $\mathcal{C}$ outputs the tuple (U^*, U_1^*, U_2^*) to break the LRSW assumption. Otherwise, $\mathcal{C}$ computes $\Gamma_1 = \widetilde{A}^{x_u^*} = (g_0^{x_u^*})^a = ((U^*)^b)^a = (U^*)^{ab}$, $\Gamma_2 = U_2/\Gamma_1^{t^*} = (U^*)^{a+t^*ab}/(U^*)^{t^*ab} = (U^*)^a$, outputs $(U^*, U_1, \Gamma_2 * (\Gamma_1)^{\widetilde{t}^*} = (U^*)^{a+\widetilde{t}^*ab})$ with a random $\widetilde{t}^*$ that has not been queried.

Probability Analysis: Let $\mathbb{E}_{LRSW}$ be the event that the output is a well-constructed solution for the LRSW assumption, $\mathbb{E}_{\mathcal{A}}$ be the event that $\mathcal{A}$ wins the game. $\epsilon' = Pr[\mathbb{E}_{LRSW}] = Pr[\mathbb{E}_{LRSW}|\overline{\mathbb{E}}_{\mathcal{A}}] \cdot Pr[\overline{\mathbb{E}}_{\mathcal{A}}] + Pr[\mathbb{E}_{LRSW}|\mathbb{E}_{\mathcal{A}}] \cdot Pr[\mathbb{E}_{\mathcal{A}}] = 0 + Pr[\mathbb{E}_{LRSW}|\mathbb{E}_{\mathcal{A}}] \cdot Pr[\mathbb{E}_{\mathcal{A}}] = 0 + \epsilon = \epsilon$. $\square$

C.3. Anonymity and Unlinkability Proof

Theorem 4.2. The PkT-KVAC scheme is *anonymous* and *unlinkable* if the DDH assumption holds.

Proof. Given a DDH challenge $\mathcal{T} = (g, \widetilde{A} = g^{\mathsf{a}}, \widetilde{B} = g^{\mathsf{b}}, \widetilde{C} = g^{\mathsf{c}})$, $\mathcal{C}$ samples $\{\alpha_i\}_{i \in [0,3]}, \beta_1, \beta_2 \beta_3, \{\gamma_k, \widetilde{\gamma}_k\}_{k \in [0,n]} \xleftarrow{\$} \mathbb{Z}_p^*$, computes $g_i = g^{\alpha_i}$ for $i \in [1,3]$, $u_k = g^{\gamma_k}$, $\widetilde{u}_k = g^{\widetilde{\gamma}_k}$ for $k \in [0,n]$, $h_1 = g^{\beta_1}$, $h_3 = g^{\beta_3}$, and sets $g_0 = \widetilde{A}^{\alpha_0}$, $h_2 = \widetilde{A}^{\beta_2}$. The tuple $(\mathbb{G}, p, \{g_i\}_{i \in [0,3]}, \{h_i\}_{i \in [1,3]}, \{u_i, \widetilde{u}_i\}_{i \in [0,n]}, H_1, f)$ is output as the public parameter *pp*. Besides, $\mathcal{C}$ initailizes empty lists $\mathcal{L}_{\mathsf{issue}}$, $\mathcal{L}_{\mathsf{show}}$, $\mathcal{L}_{\mathsf{honest}}$, $\mathcal{L}_{\mathsf{corrupt}}$ and constructs an extractor $\mathcal{E}$ and a simulator $\mathcal{S}$ for the zero-knowledge proof system.

Issuer key generation phase: $\mathcal{C}$ runs the algorithm $(\mathsf{ipk}, \mathsf{isk}) \leftarrow \mathsf{IKeyGen}(\mathsf{pp})$ as usual. The issuer key pair $(\mathsf{ipk}, \mathsf{isk})$ is sent to the adversary $\mathcal{A}$.

Oracle query phase: $\mathcal{C}$ answers the oracle queries $\mathcal{O} = \{\mathsf{UKeyGen}, \mathsf{Issue}, \mathsf{Show}, \mathsf{Verify}, \mathsf{Reveal}\}$ as follows.

•UKeyGen: $\mathcal{C}$ runs $\mathsf{UKeyGen}(\mathsf{pp}, ID) \rightarrow (\mathsf{upk}, \mathsf{usk})$, inserts $(ID, \mathsf{upk}, \mathsf{usk})$ to $\mathcal{L}_{\mathsf{honest}}$ and returns upk to $\mathcal{A}$.

•Issue: If ID has not been queried beforehand and the entry identified by ID exists in $\mathcal{L}_{\mathsf{honest}}$, $\mathcal{C}$ runs Issue with isk to produce $(\mathsf{cred}_{TP,k}, \mathcal{D})$, inserts $(ID, \mathsf{usk}, \mathsf{cred}_{TP,k}, \mathsf{ATTR}, TP, k, \mathcal{D})$ to $\mathcal{L}_{\mathsf{issue}}$, and returns $(\mathsf{cred}_{TP,k}, \mathcal{D})$ to $\mathcal{A}$.

•Show: $\mathcal{C}$ retrieves $(ID, \mathsf{usk}, \mathsf{cred}_{TP,k}, \mathsf{ATTR}, TP, k, \mathcal{D})$ from $\mathcal{L}_{\mathsf{issue}}$ and aborts if such tuple does not exist. If $\mathcal{D}$ is not empty, $\mathcal{C}$ generates $(\mathsf{tok}, \mathsf{TIN}, \mathcal{D}')$ by executing the Show algorithm, inserts $(\mathsf{tok}, \mathsf{TIN}, \mathsf{ATTR}_D, M)$ to $\mathcal{L}_{\mathsf{show}}$, updates $\mathcal{L}_{\mathsf{issue}}$ with $\mathcal{D}'$ and returns $(\mathsf{tok}, \mathsf{TIN}, \mathcal{D}')$ to $\mathcal{A}$.

•Corrupt: $\mathcal{C}$ transfers the tuple $(ID, \mathsf{upk}, \mathsf{usk})$ from $\mathcal{L}_{\mathsf{honest}}$ to $\mathcal{L}_{\mathsf{corrupt}}$ and returns usk to answer the query.

•Verify, Reveal: $\mathcal{C}$ performs the oracle query by calling the corresponding algorithms directly.

Challenge phase: $\mathcal{A}$ outputs two challenges $(ID_0^*, \mathsf{ATTR}_0^*)$, $(ID_1^*, \mathsf{ATTR}_1^*)$ and parameters $\mathsf{ATTR}_D^*, TP^*, k^*, \mathcal{D}^*, M^*)$, with the restriction that $(ID_0^*, \mathsf{cred}_0^*, \mathsf{ATTR}_0^*, TP^*, k^*, \mathcal{D}^*) \in \mathcal{L}_{\mathsf{issue}}$, $(ID_1^*, \mathsf{cred}_1^*, \mathsf{ATTR}_1^*, TP^*, k^*, \mathcal{D}^*) \in \mathcal{L}_{\mathsf{issue}}$, and $\{ID_0^*, ID_1^*\} \notin \mathcal{L}_{\mathsf{corrupt}}$. To answer the challenge, $\mathcal{C}$ selects $b \in \{0,1\}$, parses $\mathsf{cred}_b = (s_b, t_b, U_b, V_b, TP^*, k^*)$, generates $(\mathsf{tok}^*, \mathsf{TIN}^*)$ by computing:

$$\begin{aligned} C_1^* =& \widetilde{C}^{\alpha_0} V_b = g_0^{\mathsf{c}} V_b, \\ C_2^* =& \widetilde{B}^{\alpha_2} U_b = g_2^{\mathsf{b}} U_b, \\ C_3^* =& \widetilde{B}^{\alpha_3} U_b^{t_b} = g_3^{\mathsf{b}} U_b^{t_b}, \end{aligned} \qquad \begin{aligned} D_1^* =& \widetilde{C}^{\beta_2 y_2} \cdot h_1^{x_{u_b}} = (g^{\mathsf{c}})^{\beta_2 y_2} \cdot h_1^{x_{u_b}}, \\ D_2^* =& \widetilde{B}^{\beta_1 y_1} \cdot h_2^{s_b} = Y_1^{\mathsf{b}} \cdot h_2^{s_b}, \end{aligned}$$

$$\begin{aligned}C_4^* =&\frac{\widetilde{C}^{\alpha_0}}{\widetilde{B}^{\alpha_2 x_2+\alpha_3 x_3}\cdot(\widetilde{B}^{\beta_1})^{y_1y_2y_3}\cdot\widetilde{B}^{\prod_{i=0}^{n}\gamma_i z_i}\widetilde{C}^{y_1y_2y_3}}\\ =&\frac{g_0^{\mathsf{c}}}{(g_2^{x_2}g_3^{x_3}\cdot h_1^{y_1y_2y_3}\cdot\prod_{i=0}^{n}u_i^{z_i})^{\mathsf{b}}\cdot g^{\mathsf{c}\cdot\beta_2 y_1y_2y_3}},\end{aligned}$$

$$\begin{aligned}E_i^* =&\widetilde{B}^{\gamma_i}=u_i^{\mathsf{b}}, && i=0\vee \mathsf{attr}_i\in\mathsf{ATTR}_D,\\ E_i^* =&\widetilde{B}^{\gamma_i}\widetilde{u}_i^{H_1(\mathsf{attr}_i)}=u_i^{\mathsf{b}}\widetilde{u}_i^{H_1(\mathsf{attr}_i)}, && \mathsf{attr}_i\notin\mathsf{ATTR}_D.\end{aligned}$$

Then, $\mathcal{C}$ selects an element J_u^* from $\mathcal{D}^*$, computes $\alpha_0^*=f(0,TP^*,J_u^*)$, $\alpha_1^*=f(1,TP^*,J_u^*)$, $T_u^*=g_0^{1/(s_b+\alpha_0^*)}$, $F_u^*=Y_{u_b}\cdot g_0^{w^*/(s_b+\alpha_1^*)}$, where $w^*=H_1(C_u^*||D_u^*||E_u^*)$. $\mathcal{C}$ runs the simulator $\mathcal{S}$ to simulate the proof $(\Pi_U^2)^*$ and returns $tok^*=(J_u^*,C_u^*,D_u^*,E_u^*,F_u^*,(\Pi_U^2)^*,TP^*,k^*)$, $\mathsf{TIN}^*=T_u^*$.

It is obvious that (C_u^*,D_u^*,E_u^*) perfectly simulates the token components (C_u,D_u,E_u) and satisfies the relation that

$$\begin{aligned}C_1^* =&g_1^{x_1}(C_2^*)^{x_2}(C_3^*)^{x_3}(C_4^*)\cdot((D_1^*)^{y_1y_3}(D_2^*)^{y_2y_3})\cdot\\ &(E_0^*\cdot\widetilde{u}_0^{H_1(TP^*,k^*)})^{z_0}\cdot\prod_{\mathsf{attr}_i^*\in\mathsf{ATTR}_D}(E_i^*\cdot\widetilde{u}_i^{H_1(\mathsf{attr}_i^*)})^{z_i}\cdot\\ &\prod_{\mathsf{attr}_i^*\notin\mathsf{ATTR}_D}(E_i^*)^{z_i}.\end{aligned}$$

If the DDH challenge $(g,\widetilde{A},\widetilde{B},\widetilde{C})$ is a random quadruple, the token components C_1^*,C_4^*,D_1^* are random elements and therefore tok^* is independent to the user public key $g_0^{x_u}$.

Oracle query phase 2: $\mathcal{C}$ answers the oracle queries $\mathcal{O}=\{\mathsf{UKeyGen},\mathsf{Issue},\mathsf{Show},\mathsf{Verify},\mathsf{Reveal}\}$ as usual and aborts if $\mathcal{A}$ queries these oracles with $\{(\mathsf{upk}_i^*,\mathsf{usk}_i^*,\mathsf{cred}_i^*)\}_{i\in\{0,1\}}$.

Output phase: $\mathcal{A}$ outputs a bit b^*. If $b^*=b$, $\mathcal{C}$ outputs 1 to denote that the tuple $\mathcal{T}=(g,\widetilde{A},\widetilde{B},\widetilde{C})$ is a DDH quadruple; otherwise, it outputs 0 to denote that T is a random quadruple.

Probability Analysis: Let $\mathbb{E}_{\mathcal{A}}$ be the event that $\mathcal{A}$ wins the game, $\mathbb{E}_D$ be the case that $\mathcal{T}=(g,\widetilde{A},\widetilde{B},\widetilde{C})$ is a DDH quadruple, $\mathbb{E}_R$ be the case that T is a random quadruple. $\epsilon'=|Pr[\mathbb{E}_D\wedge\mathbb{E}_{\mathcal{A}}]+Pr[\mathbb{E}_R\wedge\overline{\mathbb{E}_{\mathcal{A}}}]-\frac{1}{2}|=|Pr[\mathbb{E}_D|\mathbb{E}_{\mathcal{A}}]\cdot Pr[\mathbb{E}_{\mathcal{A}}]+Pr[\mathbb{E}_R|\overline{\mathbb{E}}_{\mathcal{A}}]\cdot Pr[\overline{\mathbb{E}}_{\mathcal{A}}]-\frac{1}{2}|=|1\cdot\epsilon+(\frac{1}{2}\cdot(1-\epsilon))-\frac{1}{2}|=\frac{\epsilon}{2}$. □

C.4. k-Detectability Proof

Theorem 4.3. The PkT-KVAC scheme is *k-detectable* if the LRSW assumption holds.

Proof. Given a tuple $\mathcal{T}=(g,\widetilde{A}=g^a,\widetilde{B}=g^b)$ and a LRSW oracle $\mathcal{O}_{LRSW}(\cdot)$ which inputs $m\in\mathbb{Z}_p^*$ and outputs (h,h^b,h^{a+mab}). The challenger $\mathcal{C}$ samples $g_1,\{h_i\}_{i\in\{1,3\}},\{u_i,\widetilde{u}_i\}_{i\in[0,n]}\xleftarrow{\$}\mathbb{G}$ and $\alpha,\beta,\gamma\xleftarrow{\$}\mathbb{Z}_p^*$, sets $g_0=g$, $g_2=g_0^{\alpha}$, $g_3=g_0^{\beta}$, $h_2=g_0^{\gamma}$. The tuple $(\mathbb{G},p,\{g_i\}_{i\in[0,3]},\{h_i\}_{i\in[1,3]},\{u_i,\widetilde{u}_i\}_{i\in[0,n]},H_1,f)$ is output as the public parameter pp. Besides, $\mathcal{C}$ initailizes empty lists $\mathcal{L}_{\mathsf{issue}}$, $\mathcal{L}_{k\text{-}\mathsf{show}}$, $\mathcal{L}_{\mathsf{corrupt}}$, and constructs an extractor $\mathcal{E}$ and a simulator $\mathcal{S}$ for the zero-knowledge proof system.

Issuer key generation phase: $\mathcal{C}$ selects $x_1,y_1,y_2,y_3,z_0,\cdots,z_n\xleftarrow{\$}\mathbb{Z}_p^*$, and computes $X=g_1^{x_1}h_3^{y_3}$, $Y_1=h_1^{y_1}$, $Y_2=h_2^{y_2}$. $\mathcal{C}$ simulates the element Z by computing $Z=g_0/(\widetilde{A}^{\alpha}(\widetilde{A}\widetilde{B})^{\beta}\cdot(h_1h_2)^{y_1y_2y_3}\cdot\prod_{i=0}^{n}u_i^{z_i})=g_0/(g_2^{a}g_3^{ab}\cdot(h_1h_2)^{y_1y_2y_3}\cdot\prod_{i=0}^{n}u_i^{z_i})$ and outputs $\mathsf{ipk}=(X,Y_1,Y_2,Z)$. Note that the equation implies that $\mathcal{C}$ learns noting about the issuer secret key $x_2=a$, $x_3=ab$.

Oracle query phase: $\mathcal{C}$ answers the oracle queries $\mathcal{O}=\{\mathsf{UKeyGen},\mathsf{Issue},\mathsf{Show},\mathsf{Verify},\mathsf{Reveal}\}$ as follows.

•UKeyGen: $\mathcal{C}$ runs $\mathsf{UKeyGen}(\mathsf{pp},ID)\rightarrow(\mathsf{upk},\mathsf{usk})$, inserts $(ID,\mathsf{upk},\mathsf{usk})$ to $\mathcal{L}_{\mathsf{corrupt}}$ and returns $(\mathsf{upk},\mathsf{usk})$ to $\mathcal{A}$.

•Issue: If ID has not been queried beforehand and the entry identified by ID exists in $\mathcal{L}_{\mathsf{corrupt}}$, $\mathcal{C}$ selects $t\xleftarrow{\$}\mathbb{Z}_p^*$, queries $\mathcal{O}_{LRSW}(t)\rightarrow(U,U_1=U^b,U_2=U^{a+tab})$ and computes $V=g_1^{x_1}\cdot U_2\cdot(Y_1^{x_u}\cdot(U_1)^{\gamma y_2})^{y_3}\cdot\widetilde{u}_0^{H_1(TP,k)z_0}\prod_{i=1}^{n}\widetilde{u}_i^{H_1(\mathsf{attr}_i)z_i}$. Note that the form of credential component V implies that $U_1^{\gamma y_2}=Y_2^s$, $U_1=U^b=g_0^s$ with a random s. Then, $\mathcal{C}$ initializes a dispenser $\mathcal{D}=\{1,2,\cdots,k\}$, selects $\widetilde{s}\xleftarrow{\$}\mathbb{Z}_p^*$ and inserts $\{\mathsf{TIN}_i=g_0^{1/(\widetilde{s}+f(0,TP,i))}\}_{i\in\mathcal{D}}$ to $\mathcal{L}_{k\text{-}\mathsf{show}}$. $\mathcal{C}$ sets $\mathsf{cred}_{TP,k}=(\widetilde{s},t,U,V,TP,k)$, inserts the tuple $(ID,\mathsf{usk},\mathsf{cred}_{TP,k},\mathsf{ATTR},TP,k,\mathcal{D})$ to $\mathcal{L}_{\mathsf{issue}}$ and returns $(\mathsf{cred}_{TP,k},\mathcal{D})$ to $\mathcal{A}$.

•Show: $\mathcal{C}$ retrieves $\mathsf{cred}_{TP,k}=(\widetilde{s},t,U,V,TP,k)$ from $\mathcal{L}_{\mathsf{issue}}$ and runs $\mathsf{Show}\rightarrow(\mathsf{tok},\mathsf{TIN},\ \mathcal{D}')$, where the token component is computed as $D_2=Y_1^r\cdot h_2^{\widetilde{s}}$, $T_u=g_0^{1/(\widetilde{s}+\alpha_0)}$, $F_u=Y_u\cdot g_0^{w/(\widetilde{s}+\alpha_1)}$. $\mathcal{C}$ updates $\mathcal{L}_{\mathsf{issue}}$ with $\mathcal{D}'$ and returns $(\mathsf{tok},\mathsf{TIN})$.

•Verify: If Π_U^2 is valid, $\mathcal{C}$ runs the extractor $\mathcal{E}$ to extract $(x_u,\widetilde{s},t,r,w)$, computes $V=C_1/g_0^r$, $U=C_2/g_2^r$. If $\mathsf{cred}_{TP,k}=(\widetilde{s},t,U,V,TP,k)$ exists in $\mathcal{L}_{\mathsf{issue}}$ and $\mathsf{ATTR}_D\subseteq\mathsf{ATTR}$, $\mathcal{C}$ returns 1. Otherwise, $\mathcal{C}$ returns 0.

•Reveal: $\mathcal{C}$ performs the oracle query by calling the Reveal algorithm directly.

Forge phase: $\mathcal{A}$ outputs a forged tuple $(ATTR_D^*,M^*,\mathsf{tok}^*,\mathsf{TIN}^*)$, where $tok^*=(J_u^*,C_u^*,D_u^*,E_u^*,F_u^*,(\Pi_U^2)^*)$. $\mathcal{C}$ aborts if $\mathsf{TIN}^*\in\mathcal{L}_{k\text{-}\mathsf{show}}\vee(\Pi_U^2)^*$ is invalid. If tok^* is a well-formed token, we have

$$\begin{aligned}C_2^* =&g_2^{r^*}U^*,\\ C_1^* =&g_0^{r^*}V^*=g_0^{r^*}g_1^{x_1}(U^*)^{a+t^*ab}(Y_1^{x_u^*}Y_2^{s^*})^{y_3}\cdot\\ &\widetilde{u}_0^{H_1(TP,k)z_0}\cdot\prod_{i=1}^{n}\widetilde{u}_i^{H_1(\mathsf{attr}_i^*)z_i},\end{aligned}$$

where $Y_2^{s^*}=g_0^{\gamma y_2\cdot s^*}=(U^*)^{b\gamma y_2}=(U_1^*)^{\gamma y_2}$. Then, $\mathcal{C}$ runs the extractor $\mathcal{E}$ to extract r^*,t^*,s^*,x_u^*, computes $U^*=C_2^*/g_2^{r^*}$, $U_1^*=g_0^{s^*}$,

$$\begin{aligned}U_2^* =&C_1^*/(g_0^{r^*}g_1^{x_1}(Y_1^{x_u^*}Y_2^{s^*})^{y_3}\cdot\widetilde{u}_0^{H_1(TP,k)z_0}\prod_{i=1}^{n}\widetilde{u}_i^{H_1(\mathsf{attr}_i^*)z_i})\\ =&(U^*)^{a+t^*ab}.\end{aligned}$$

If t^* is not queried to $\mathcal{O}_{LRSW}(\cdot)$. $\mathcal{C}$ outputs the tuple (U^*,U_1^*,U_2^*) to break the LRSW assumption. Otherwise, $\mathcal{C}$ computes $\Gamma_1=\widetilde{A}^{s^*}=(g_0^{s^*})^a=((U^*)^b)^a=(U^*)^{ab}$, $\Gamma_2=U_2/\Gamma_1^{t^*}=(U^*)^{a+t^*ab}/(U^*)^{t^*ab}=(U^*)^a$, outputs $(U^*,U_1,\Gamma_2*(\Gamma_1)^{\widetilde{t}^*}=(U^*)^{a+\widetilde{t}^*ab})$ with a random $\widetilde{t}^*$ that has not been queried.

Probability Analysis: Let $\mathbb{E}_{LRSW}$ be the event that the output is a well constructed solution for the LRSW assumption, $\mathbb{E}_{\mathcal{A}}$ be the event that $\mathcal{A}$ wins the game. $\epsilon'=Pr[\mathbb{E}_{LRSW}]=$

$Pr[\mathbb{E}_{LRSW}|\overline{\mathbb{E}}_{\mathcal{A}}] \cdot Pr[\overline{\mathbb{E}}_{\mathcal{A}}] + Pr[\mathbb{E}_{LRSW}|\mathbb{E}_{\mathcal{A}}] \cdot Pr[\mathbb{E}_{\mathcal{A}}] = 0 + Pr[\mathbb{E}_{LRSW}|\mathbb{E}_{\mathcal{A}}] \cdot Pr[\mathbb{E}_{\mathcal{A}}] = 0 + \epsilon = \epsilon.$ □

C.5. Exculpability Proof

Theorem 4.4. The PkT-KVAC scheme is *exculpable* if the DL assumption holds.

Proof. Given a DL challenge $(g_0, \widetilde{A} = g_0^a)$, the challenger $\mathcal{C}$ runs $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$, $(\mathsf{ipk}, \mathsf{isk}) \leftarrow \mathsf{IKeyGen}(\mathsf{pp})$ and returns $(\mathsf{pp}, \mathsf{ipk}, \mathsf{isk})$ to $\mathcal{A}$. Besides, $\mathcal{C}$ initailizes empty lists $\mathcal{L}_{\mathsf{UKey}}$, $\mathcal{L}_{\mathsf{registerd}}$, $\mathcal{L}_{\mathsf{issue}}$, $\mathcal{L}_{\mathsf{show}}$, and constructs an extractor $\mathcal{E}$ and a simulator $\mathcal{S}$ for the zero-knowledge proof system.

Oracle query phase: $\mathcal{C}$ answers the oracle queries $\mathcal{O} = \{\mathsf{UKeyGen}, \mathsf{Issue}, \mathsf{Show}, \mathsf{Verify}, \mathsf{Reveal}\}$ as follows.

•UKeyGen: $\mathcal{C}$ selects $r_u \xleftarrow{\$} \mathbb{Z}_p^*$, computes $Y_u = \widetilde{A}^{r_u} = g_0^{a \cdot r_u}$ and returns $\mathsf{upk} = Y_u$ to $\mathcal{A}$. The tuple (ID, upk, r_u) is inserted to $\mathcal{L}_{\mathsf{honest}}$.

•Issue: If ID has not been queried beforehand and the entry identified by ID exists in $\mathcal{L}_{\mathsf{honest}}$, $\mathcal{C}$ runs Issue with isk to produce $(\mathsf{cred}_{TP,k}, \mathcal{D})$, inserts $(\mathsf{upk}, \mathsf{usk}, \mathsf{cred}_{TP,k}, \mathsf{ATTR}, TP, k, \mathcal{D})$ to $\mathcal{L}_{\mathsf{issue}}$, and returns $(\mathsf{cred}_{TP,k}, \mathcal{D})$ to $\mathcal{A}$.

•Show: $\mathcal{C}$ retrieves $(ID, \mathsf{usk}, \mathsf{cred}_{TP,k}, \mathsf{ATTR}, TP, k, \mathcal{D})$ from $\mathcal{L}_{\mathsf{issue}}$ and aborts if such tuple does not exist. If $\mathcal{D}$ is not empty, $\mathcal{C}$ selects J_u from the dispenser $\mathcal{D}$ and runs algorithm Show to compute $(C_u, D_u, E_u, T_u, F_u)$. Then, $\mathcal{C}$ runs the simulator $\mathcal{S}$ to produce Π_U^2, returns $(\mathsf{TIN} = T_u, \mathsf{tok} = (J_u, C_u, D_u, E_u, F_u, \Pi_U^2, TP, k))$ and inserts $(ATTR_D, M, \mathsf{tok}, \mathsf{TIN})$ to $\mathcal{L}_{\mathsf{show}}$. Besides, $\mathcal{C}$ updates $\mathcal{L}_{\mathsf{issue}}$ with the updated dispenser $\mathcal{D}' = \mathcal{D} \backslash \{J_u\}$.

•Corrupt: $\mathcal{C}$ removes the tuple (ID, upk, r_u) from $\mathcal{L}_{\mathsf{honest}}$. Then, $\mathcal{C}$ randomly picks $\delta \xleftarrow{\$} \mathbb{Z}_p^*$, inserts $(ID, \mathsf{upk}, \delta)$ to $\mathcal{L}_{\mathsf{corrupt}}$ and returns $\mathsf{usk} = \delta$ to answer the query.

•Verify: The oracle query aborts if the entry identified by TIN does not exist in $\mathcal{L}_{\mathsf{show}}$. $\mathcal{C}$ return 1 if the tuple $(\mathsf{tok}, \mathsf{TIN}, \mathsf{ATTR}_D, M) \in \mathcal{L}_{\mathsf{show}}$ and returns 0 otherwise.

•Reveal: $\mathcal{C}$ performs the oracle query by calling the corresponding algorithm directly.

Forge phase: $\mathcal{A}$ outputs a forged tuple $(\mathsf{ATTR}_D^*, M^*, TIN^*, \mathsf{tok}^*, \widetilde{\mathsf{tok}}^*)$ to frame a honest token pair (TIN^*, tok^*) for double-showing. If $(\mathsf{upk}^*, ID^*) \leftarrow \mathsf{Reveal}(TIN^*, \mathsf{tok}^*, \widetilde{\mathsf{tok}}^*)$ exists in $\mathcal{L}_{\mathsf{honest}}$ and $\mathsf{Verify}(\mathsf{isk}, \widetilde{\mathsf{tok}}^*, \mathsf{TIN}^*, \mathsf{ATTR}_D^*, M^*)$ returns 1, we have

$$\mathsf{upk}^* = Y_u^* = g_0^{x_u^*} = g_0^{a \cdot r_u^*}.$$

Parsing $\widetilde{\mathsf{tok}}^* = (J_u^*, C_u^*, D_u^*, E_u^*, F_u^*, (\Pi_U^2)^*, TP^*, k^*)$, $\mathcal{C}$ runs the extractor $\mathcal{E}$ to extract x_u^* from $(\Pi_U^2)^*$, outputs $a^* = x_u^*/r_u^*$ to break the DL assumption.

Probability Analysis: Let $\mathbb{E}_{DL}$ be the event that a^* is the solution for DL problem, $\mathbb{E}_{\mathcal{A}}$ be the event that $\mathcal{A}$ wins the exculpability game. $\epsilon' = Pr[\mathbb{E}_{DL}] = Pr[\mathbb{E}_{DL}|\overline{\mathbb{E}}_{\mathcal{A}}] \cdot Pr[\overline{\mathbb{E}}_{\mathcal{A}}] + Pr[\mathbb{E}_{DL}|\mathbb{E}_{\mathcal{A}}] \cdot Pr[\mathbb{E}_{\mathcal{A}}] = 0 + Pr[\mathbb{E}_{DL}|\mathbb{E}_{\mathcal{A}}] \cdot Pr[\mathbb{E}_{\mathcal{A}}] = 0 + \epsilon = \epsilon.$ □

D. PkT-SIN: Security Models

We formalize the security model for PkT-SIN, which is utilized to prove that PkT-SIN can achieve authenticated key exchange (AKE) in a random oracle model.

Protocol Participants. Let the symbol U^ρ denote the ρ-th instance of participant U in the PkT-SIN, i.e., NCC, LEO, GS or SU.

Adversary Capability. We capture all of the adversary's attack capabilities in real-world to have full control over the public network communication, including revealing some secrets in the protocol, intercepting or tampering with the channel messages, replaying, delaying, injecting or dropping data packets, interleaving messages from different sessions, etc.

Protocol Execution. An adversary $\mathcal{A}$ is modeled as a PPT machine with a distinguished query tape to issue a set of session exposure queries for gaining the ephemeral and long-term secrets possessed by participants.

• $\mathsf{Extract}(U^\rho)$: In this security game, an adversary $\mathcal{A}$ can get the public/private key pair corresponding to the participant U^ρ.

• $\mathsf{Send}(U^\rho, M)$: transmits a message M to U^ρ, who executes the protocol and returns the operation result to adversary $\mathcal{A}$. If the message in the query causes the protocol to execute or abort, it will be made known to $\mathcal{A}$.

• $\mathsf{Execute}(\{U_i\}_{i=1}^n)$: executes a complete protocol among the entities $(U_1, \cdots, U_n)$. The adversary captures all messages transmitted over the public network. Hence, the query to Execute oracle models passive eavesdropping capability of the adversary.

• $\mathsf{Reveal}(U^\rho, sid)$: returns the secret session key (associated with the participant U^ρ and session identifier sid) to the adversary $\mathcal{A}$.

• $\mathsf{Corrupt}(U^\rho, sid)$: returns all information (including ephemeral secrets and private key) held by U^ρ.

• $\mathsf{TestSession}(U^\rho, sid)$: This oracle is used to model key secrecy. A random bit $b \in \{0, 1\}$ is selected to respond to this query. If $b = 1$, the target session key is returned to $\mathcal{A}$. Otherwise, a random value picked from the secret session key space is returned.

Session Exposure. A session (U^ρ, sid) is said to be *exposed* if the adversary makes the following queries.

- The adversary makes a Reveal query on the session.
- The adversary makes a Corrupt query on U^ρ before the session has expired.

Session Freshness. A session (U^ρ, sid) is said to be *fresh* if itself is not exposed and all its matching sessions are not exposed.

Definition D.1 Let $\mathsf{Succ}_{\mathsf{PkT\text{-}SIN}}^{\mathsf{AKE}}(\mathcal{A})$ denote the event that $\mathcal{A}$ makes a single Test query with the restriction that the queried session (U^ρ, sid) is fresh, and finally outputs a bit $b' = b$, where b is the random value selected in the Test query. A PkT-SIN system is secure if for any PPT adversary $\mathcal{A}$, there exists a negligible function ν such that $\mathsf{Adv}_{\mathsf{PkT\text{-}SIN}}^{\mathsf{AKE}}(\mathcal{A}) \overset{\mathsf{def}}{=} 2\mathsf{Pr}[\mathsf{Succ}_{\mathsf{PkT\text{-}SIN}}^{\mathsf{AKE}}(\mathcal{A})] - 1 \leq \nu(\lambda)$.

E. PkT-SIN Security Proof

Theorem 6.1 PkT-SIN realizes authenticated key exchange (AKE) if the DDH assumption holds, PkT-KVAC and DSS are unforgeable, SE satisfies confidentiality, and hash functions are random oracles.

Proof. We setup the AKE proof in terms of a series of games, where a challenger interacts with the adversary confronting it with a counterfeit TestSession challenge in the spirit of Definition D.1.

In the security proof, assume that $1/P$ is the maximum probability that the input and output messages from two different sessions are the same. We denote q_{exe} as the number of Execute queries, and q_{snd} as the number of Send queries.

Let $\mathcal{B}_1$ be the adversary against the confidentiality of SE, $\mathcal{B}_2$ be the adversary against the unforgeability of PkT-KVAC, $\mathcal{B}_3$ be the adversary against the unforgeability of DSS, $\mathcal{B}_4$ be the adversary against the security of the DDH problem and $\mathcal{B}_5$ be the adversary against the security of hash functions (modeled as random oracles).

– Game_0. This first game corresponds to a real attack, in which all the parameters are chosen as in the actual scheme. A random bit $b \in \{0,1\}$ is selected. When $b = 1$, the real session key is returned as a response to the TestSession query. Otherwise, a random key from the key space is returned as the session key. By definition, $\mathsf{Adv}_0(\mathcal{A}) = \mathsf{Adv}^{\mathsf{AKE}}_{\mathsf{PkT\text{-}SIN}}(\mathcal{A})$.

– Game_1. This game is the same as Game_0, except that if two different sessions output exactly the same message and have the same partner, the protocol halts. Hence, we have

$$\mathsf{Adv}_0(\mathcal{A}) \leq \mathsf{Adv}_1(\mathcal{A}) + (q_{exe} + q_{snd})^2/P.$$

– Game_2. In this game, the simulation is modified in the following way: for a query to Execute oracle, it would return ciphertexts $\mathsf{CT}_{L,U}$ and $\mathsf{CT}_{L,G}$, or values CT_{R_1} and CT_{R_2} chosen uniformly at random in the ciphertext space of $\mathcal{SE}$.SEnc are returned instead. An adversary $\mathcal{A}$ not holding the corresponding secret keys $\mathsf{EK}_{L,U}$ and $\mathsf{EK}_{L,G}$ has a negligible advantage in distinguishing between this game and the previous one, because of the privacy of the SE. Therefore, we have

$$\mathsf{Adv}_1(\mathcal{A}) \leq \mathsf{Adv}_2(\mathcal{A}) + 2 \cdot \mathbf{Adv}^{\mathsf{conf}}_{\mathsf{SE}}(\mathcal{B}_1).$$

– Game_3. In this game, the simulation is modified in the following way: for a query to Execute oracle, it would return authentication token and identifier $(\mathsf{tok}, \mathsf{TIN})$ for satellite user, or values chosen uniformly at random in the token space and identifier space of PkT-KVACShow are returned instead. An adversary $\mathcal{A}$ not holding the corresponding user secret key usk has a negligible advantage in distinguishing between this game and the previous one, because of the unforgeability of PkT-KVAC. On the other hand, the modification made in the previous game ensures that $\mathcal{A}$ has not obtained any usk by simulating an execution of PkT-KVAC.UKeyGen. Therefore, we have

$$\mathsf{Adv}_2(\mathcal{A}) \leq \mathsf{Adv}_3(\mathcal{A}) + \cdot\mathbf{Adv}^{\mathsf{unforge}}_{\mathsf{PkT\text{-}KVAC}}(\mathcal{B}_2).$$

– Game_4. This game is the same as Game_3, except that the calculations of the DSS signatures are replaced by random values, and we have

$$\mathsf{Adv}_3(\mathcal{A}) \leq \mathsf{Adv}_4(\mathcal{A}) + \mathbf{Adv}^{\mathsf{unforge}}_{\mathsf{DSS}}(\mathcal{B}_3).$$

– Game_5. This game is the same as Game_4, except that (R_u, Y_{gs}) is replace by the DDH tuple $(g_0^a, g_0^{b_1})$, and (R_u, R_{gs}) by the DDH tuple $(g_0^a, g_0^{b_2})$ on group G_2. Then, the calculation of $R_u^{x_{gs}}$ is replaced by $T_1 \in \mathbb{G}$, and $R_u^{r_{gs}}$ by $T_2 \in \mathbb{G}$. As a result, we have

$$\mathsf{Adv}_4(\mathcal{A}) \leq \mathsf{Adv}_5(\mathcal{A}) + 2\mathbf{Adv}_{\mathsf{DDH}}(\mathcal{B}_4).$$

– Game_6. This game is the same as Game_5, except that the hash functions (H_1, H_2, H_3, H_4) are replaced by random oracles, and we have $\mathsf{Adv}_5(\mathcal{A}) = \mathsf{Adv}_6(\mathcal{A})$.

Combining the above results, we have

$$\begin{aligned}
&\mathbf{Adv}^{\mathsf{AKE}}_{\mathsf{PkT\text{-}SIN}}(\mathcal{A}) = \mathsf{Adv}_0(\mathcal{A}) \\
&\leq (q_{exe} + q_{snd})^2/P + (q_{exe} + 3q_{snd})^2/p + 2q_{snd}^2/|\mathcal{R}| \\
&\quad + 2(q_{exe} + q_{snd})(\mathbf{Adv}^{\mathsf{conf}}_{\mathsf{SE}}(\mathcal{B}_1) + \mathbf{Adv}^{\mathsf{unforge}}_{\mathsf{PkT\text{-}KVAC}}(\mathcal{B}_2) \\
&\qquad + \mathbf{Adv}^{\mathsf{unforge}}_{\mathsf{DSS}}(\mathcal{B}_3)) \\
&\quad + 2(q_{exe} + q_{snd})\mathbf{Adv}_{\mathsf{DDH}}(\mathcal{B}_4)).
\end{aligned}$$

This completes the proof. □

Theorem 6.2 PkT-SIN satisfies anonymity for satellite users if PkT-KVAC satisfies anonymity and the token series numbers are not reused.

Proof. Suppose that an adversary $\mathcal{A}$ is able to obtain messages $(\mathsf{tok}, \mathsf{TIN}, R_u)$ in session establishment phase of PkT-SIN, where the tuple $(\mathsf{tok}, \mathsf{TIN})$ is generated by PkT-KVAC.Show algorithm. User's real identity is hidden in tok. If $\mathcal{A}$ can obtain the user's real identity from tok, it indicates that $\mathcal{A}$ can break the anonymity of PkT-KVAC, or the token series numbers are reused by satellite user. Since the anonymity of PkT-KVAC is proved in Supplemental Material C, $\mathcal{A}$ cannot derive the user's real identity from tok. If a satellite user does not reuse the token series numbers (i.e., behaves well), the adversary $\mathcal{A}$ has no chance to reveal the user's identity using PkT-KVAC.Reveal algorithm. Therefore, the proposed PkT-SIN system satisfies anonymity for satellite users.

The anonymity of satellite users in the handover phase is also guaranteed, which is omitted due to a similar proof process. □