E2E-AKMA: An End-to-End Secure and Privacy-Enhancing AKMA Protocol Against the Anchor Function Compromise

Yueming Li[1, 2], Long Chen[1, ✉], Qianwen Gao[1, 2], Zhenfeng Zhang[1, ✉]
1. Institute of Software, Chinese Academy of Sciences, Beijing, China
2. University of Chinese Academy of Sciences
Email: {yueming2021,chenlong,qianwen2021,zhenfeng}@iscas.ac.cn

Abstract

The Authentication and Key Management for Applications (AKMA) system represents a recently developed protocol established by 3GPP, which is anticipated to become a pivotal component of the 5G standards. AKMA enables application service providers to delegate user authentication processes to mobile network operators, thereby eliminating the need for these providers to store and manage authentication-related data themselves. This delegation enhances the efficiency of authentication procedures but simultaneously introduces certain security and privacy challenges that warrant thorough analysis and mitigation.

The 5G AKMA service is facilitated by the AKMA Anchor Function (AAnF), which may operate outside the boundaries of the 5G core network. A compromise of the AAnF could potentially allow malicious actors to exploit vulnerabilities, enabling them to monitor user login activities or gain unauthorized access to sensitive communication content. Furthermore, the exposure of the Subscription Permanent Identifier (SUPI) to external Application Functions poses substantial privacy risks, as the SUPI could be utilized to correlate a user's real-world identity with their online activities, thereby undermining user privacy.

To mitigate these vulnerabilities, we propose a novel protocol named E2E-AKMA, which facilitates the establishment of a session key between the User Equipment (UE) and the Application Function (AF) with end-to-end security, even in scenarios where the AAnF has been compromised. Furthermore, the protocol ensures that no entity, aside from the 5G core network, can link account activities to the user's actual identity. This architecture preserves the advantages of the existing AKMA scheme, such as eliminating the need for complex dynamic secret data management and avoiding reliance on specialized hardware (apart from standard SIM cards). Experimental evaluations reveal that the E2E-AKMA framework incurs an overhead of approximately 9.4% in comparison to the original 5G AKMA scheme, which indicates its potential efficiency and practicality for deployment.

1 Introduction

In mobile communication networks, authentication via the cellular network provides a higher level of security, greater convenience of use, and cost-effective deployment for application servers. One example of a delegated authentication system is AKMA, which has been proposed by the 3rd Generation Partnership Project (3GPP) and is specified in TS 33.535 [4]. This system is planned to be integrated into the 5G standards. AKMA enables application service providers to delegate user authentication tasks to mobile network operators, thereby eliminating the need for maintaining a separate confidential database.

As discussed in Section 2, the system involves three key participants: the Home Network (HN), User Equipment (UE), and Application Function (AF). The HN is established and managed by telecommunications operators such as AT&T, T-Mobile, and Verizon. UE refers to the user's mobile device, which includes a SIM card issued by the corresponding HN. The AF represents application service providers, such as social networking platforms and online shopping websites. The AF delegates the task of authenticating the UE to the respective HN to which the UE is subscribed. To facilitate this process, the HN deploys a specialized network element known as the AKMA Anchor Function (AAnF), which is designed to provide this specific service. This mechanism allows the AF to verify the identity of the UE through the HN without requiring access to additional information or knowledge about the user's actual identifier.

AKMA offers a promising solution for enhancing security and convenience in user authentication. Unlike password-based authentication methods, it does not require AF to store a private dynamic database, thereby reducing the risk of database breaches [14,28,30]. By leveraging cellular network-based authentication, these systems enable seamless logins for users and offload the burden of authentication-related data storage and maintenance from AFs. Furthermore, integrating AKMA into 5G makes it more secure than SMS OTP, as it does not involve the personal participant of the user, which decreases the risks of phishing attacks [13]. Additionally, the convenience of AKMA is further enhanced by not requiring users to prepare additional hardware, such as FIDO WebAuthn [29], since cell phones are ubiquitous.

AKMA is currently in its early stages of development and remains an active area of research. 3GPP Technical Report (TR) 33.835 [3] outlines seventeen critical issues (#1–#17) that must be resolved to enable the progression of AKMA. Among these, five key issues (#3, #5, #6, #7, and #16) are specifically related to the protocol layer and present substantial security and privacy challenges, thereby threatening the fundamental integrity of the protocol. While some of these issues can be addressed through minor adjustments to existing protocols, others appear to require significant modifications to the current architecture in order to be effectively resolved.[1]

Communication Security. Addressing Key Issue #6, which pertains to secure communication between UE and AFs, remains one of the most challenging problems in modern network security. Specifically, session keys used for UE-AF communication are accessible to the AAnF. This accessibility introduces potential vulnerabilities, as the AAnF could intercept communications between UEs and AFs. Notably, the AAnF resides within the Home Network but operates outside the 5G core network, lacking the same trust designation as core network components. This distinction further exacerbates security concerns, suggesting heightened risks [32].

Ensuring the confidentiality of session keys for UE-AF communication is therefore paramount, especially in scenarios where the AAnF may be compromised. However, this requirement presents significant challenges, as the trust foundation for UE-AF communication authentication is inherently tied to the AAnF. *If the AAnF is compromised, adversaries could exploit this vulnerability to execute man-in-the-middle attacks, thereby undermining the security of the entire communication scheme.*

Yang et al. [32] proposed the development of AKMA+, which aims to protect communication between UEs and AFs from attackers, including those within the HN. Nevertheless, recent studies [20] have demonstrated that if an adversary is able to compromise the AAnF—as assumed in the AKMA+ threat model—they can impersonate legitimate UEs to AFs. This vulnerability arises as a direct consequence of the centralized trust architecture inherent in AKMA+.

User Privacy. The other critical issues that warrant attention are Key Issue #5 (user privacy) and Key Issue #7 (protection of subscriber personal information in control and data traffic), both of which are intricately linked to the potential exposure of a user's identity and behavioral privacy. Notably, AFs can track the user's SUPI. The SUPI is directly associated with the user's actual identity, which poses a significant risk of privacy leakage. Furthermore, the AAnF has the authority to identify the user's intended application functions, as well as monitor the frequency and timing of their logins. This ability to track login activities could potentially reveal sensitive insights into the user's daily behavioral patterns.

On the other hand, Key Issue #5 simultaneously stipulates that when a network operator seeks to provide authentication and key management services to an application server, it must possess the capability to exchange subscriber-related information. This may enable the application server to identify its users effectively. Consequently, there arises a need for an alternative type of identifier (either permanent or temporary) to facilitate user identification between the 3GPP network and the application server. Additionally, the Mobile Network Operators(MNO) must be able to map this alternative identifier to the permanent identifier within its domain. This dual requirement creates a fundamental contradiction in protocol design: *while there is a necessity to safeguard user privacy, there is also a need to retain the ability to trace a user's true identity.*

In conclusion, it is of paramount importance to enhance the existing AKMA protocol to address critical security and privacy concerns. Firstly, the protocol must be fortified to guarantee secure and confidential communication between the UE and the AF, even in scenarios where a temporary security breach occurs within the HN. Secondly, the AKMA protocol should emphasize the protection of user privacy, ensuring that neither the AF nor a compromised AAnF can correlate a user's account activity with their real identity. At the same time, the HN must possess the capability to trace a user's real identity when necessary.

The preservation of the key advantages inherent in the existing AKMA protocol is of paramount importance. One key advantage of its design is eliminating the need for a private database. This significantly reduces the risk of database breaches, thereby enhancing overall security. Additionally, the protocol does not necessitate the acquisition of additional hardware, as it operates seamlessly on existing infrastructure, such as SIM cards. These features collectively contribute to improving security while also reducing the overhead associated with the dynamic storage requirements of AF.

1.1 Our Contributions

This paper presents an innovative framework for the AKMA scheme, referred to as End-to-End AKMA (E2E-AKMA). The primary objective of E2E-AKMA is to establish session keys for UE-AF communication with robust end-to-end security while simultaneously strengthening privacy protection mechanisms. In particular, this study focuses on scenarios where the AAnF may be compromised, addressing concerns raised by previous works, including [3, 32].

End-to-End Security. The E2E-AKMA framework incor-

[1]The remaining issues outlined in TR 33.835 focus on aspects such as the architectural framework, interface design, API functionality, and regulatory compliance of AKMA. These issues fall outside the scope of this study, as they cannot be effectively resolved through a protocol-centric approach.

porates robust security mechanisms aimed at preventing adversaries from impersonating users or gaining unauthorized access to their accounts to retrieve sensitive information. Importantly, these security measures remain effective even under circumstances where the AAnF on the HN is temporarily compromised. For an adversary to carry out malicious activities successfully, it would be necessary to simultaneously compromise both the UE and the AAnF, or compromise the UE while obtaining physical possession of the SIM card. This framework effectively addresses key issues 6 and 16, while implicitly resolving issue 3 as outlined in 3GPP TR 33.835 [3].

Privacy Enhancement. The E2E-AKMA framework addresses two critical privacy issues (Key Issues #5 and #7 in [3]) through strategic design improvements.

- **Protection against compromised AAnF.** When the AAnF is compromised, it remains unaware of the specific AF or account being accessed by the UE, minimizing exposure of sensitive account information and user behavioral patterns.
- **Prevention of cross-AF tracking.** The framework prevents UEs from being globally identifiable across different AFs, providing robust protection against re-identification. When a UE maintain accounts across multiple AFs, these AFs cannot associate or link the accounts to the same user during registration or login processes, even under AF collusion. Similarly, multiple accounts within a single AF cannot be correlated or traced back to the same UE.
- **Preservation of MNO Functionality**
  The proposed privacy enhancements are designed to ensure that the Mobile Network Operators (MNOs) retain their ability to identify users across the 3GPP network and associated application servers. Specifically, the 5G core network continues to support the mapping of user accounts to their SUPI. This capability enables MNOs to exchange subscriber-related information seamlessly while preserving critical auditing and management functionalities. Such functionalities are essential for maintaining operational efficiency and adhering to regulatory compliance requirements.

Deployability and Usability. Our protocol retains the established advantages of AKMA, particularly in terms of its deployment and operational procedures.

- **Elimination of Confidential Data Repositories**. The E2E-AKMA system eliminates the need for any party to manage large-scale repositories containing sensitive or confidential data. This approach not only mitigates the risk of data breaches but also significantly reduces deployment costs.
- **No Additional Secure Hardware Required.** The implementation of our E2E-AKMA system eliminates the need for additional secure hardware or any modifications to the existing SIM card infrastructure. This design facilitates effortless integration with current systems, offering a practical and cost-efficient solution for strengthening the security of the AKMA scheme.

Our experimental findings, conducted using a real mobile phone and cloud servers, demonstrate that the login time for the E2E-AKMA protocol is 464 milliseconds, representing only a 9.4% overhead compared to the 5G AKMA protocol. Furthermore, in real-world applications, the performance of the protocol can be enhanced further by leveraging hardware optimizations.

1.2 Technique Overview

Our E2E-AKMA protocol is designed based on several key observations and security considerations:

First, leveraging AF certificates for unidirectional authentication. We observe that AFs possess certificates in the 5G infrastructure, enabling UEs to authenticate AFs through TLS. Therefore, we only need to address the challenge of AF authenticating UE. We adopt a *trust on first use* approach where UEs register their public keys with AFs while keeping their private keys protected through distributed storage.

Second, mitigating single point of failure through distributed key management. We recognize that relying solely on AAnF for authentication creates a critical vulnerability - if AAnF is compromised, the entire system's security is at risk. To address this, we split the private key into two parts: one stored at AAnF and another at UE. Crucially, accessing the AAnF's key share requires SIM-based authentication through the 5G AKA channel. This design ensures that:1). If UE is compromised but the SIM card secrets remain secure, the adversary cannot invoke AAnF's private key for signing. 2). If AAnF is compromised, the adversary still lacks the UE's key share, preventing successful signature generation

Third, ensuring privacy against AAnF through cryptographic indistinguishability. To protect user privacy from AAnF, we ensure that for each account, AAnF uses the same public key for the same SUPI. This design prevents AAnF from correlating different public keys to distinct services. To achieve this goal, our solution employs a single signature generated interactively by both the AAnF and UE through our novel *Two-Party Oblivious Signature* (2POS) protocol. Each account is associated with a single public key PK_S, with the UE holding private key SK_1 and the AAnF holding SK_2. The UE authenticates himself to HN via 5G AKA and uses 2POS to generate signatures for AF challenges. The AAnF remains unaware of the generated signature, while the UE does not know SK_2 during this approach. Crucially, each SK_1 is unique to different accounts, whereas SK_2 remains consistent across all accounts for a UE. This ensures the AAnF cannot identify which AF the UE is authenticating to.

The 2POS protocol facilitates the generation of standard ECDSA signatures by distributing the secret keys between the AAnF and the UE. A central design principle of the protocol is that messages originating from the UE are encrypted using homomorphic encryption techniques. This ensures that the AAnF is unable to differentiate between various SK_1 values. Furthermore, since the AAnF does not possess the decryption key, it can not independently generating final signatures.

This mechanism effectively prevents unauthorized signature creation while simultaneously preserving user privacy.

Fourth, achieving unlinkability across AFs. For privacy against AFs, while AAnF uses the same public key for identical SUPIs, UE generates different public keys for each account. From AF's perspective, each account's public key appears independently and randomly generated, preventing cross-service user linking even if multiple AFs collude. More specifically, due to the design of the 2POS protocol, AFs cannot link accounts to the same UE since they cannot derive SK_1 and SK_2 from PK_S and signatures.

Fifth, enabling lawful traceability while preserving privacy. To support regulatory compliance, we have AAnF encrypt the SUPI using the core network's public key and include this ciphertext as part of the account's public identifier forwarded to AF. This design ensures that: 1).AFs cannot decrypt the SUPI, preserving user privacy at the service level. 2). The core network can decrypt and recover the SUPI for lawful interception when necessary. 3) Even if AAnF is malicious and provides incorrect SUPI encryption (e.g., using its own public key), AF's re-randomization process acts as a second layer of encryption, making malicious ciphertexts indistinguishable from random, preventing AAnF from tracking users through this field.

Additionally, to thwart phishing attacks, we hash the identity of AF together with the challenge as the message to be signed. This binds the identity of AF to the challenge, ensuring that the adversary cannot create a malicious AF$'$ to impersonate the legitimate AF, forwarding the challenge obtained from the legitimate AF to UE for signing, thereby launching a phishing attack against UE. This protection is particularly crucial in scenarios where a phishing website AF$'$ mimics a legitimate website AF, and the user mistakenly fails to verify the correct TLS certificate, inadvertently connecting to the malicious service.

1.3 Related Work

Comparison with AKMA+. The AKMA+ protocol proposed by Yang et al. [32] addresses several security and privacy vulnerabilities in the original 5G AKMA protocol. However, compared to our E2E-AKMA approach, AKMA+ exhibits critical limitations in both security and usability aspects.

The primary vulnerability of AKMA+ lies in its persistent dependence on centralized key storage within the AAnF. Specifically, all AKMA anchor keys ($K_{\mathrm{AKMA,i}}$) are stored in the AAnF, which introduces a critical single point of failure. In the event that the AAnF is compromised—a scenario explicitly considered within the threat model—adversaries can gain access to all stored keys. This access enables them to derive application keys (K_{AF}), decrypt previously recorded sessions, and impersonate legitimate users. Furthermore, even if privacy-hardened application keys are employed through Diffie-Hellman key exchange, the compromise of anchor keys would still allow adversaries to successfully impersonate the UE.

AKMA+ requires the AAnF to dynamically generate and manage multiple key-identifier pairs $(K_{AKMA,i}, \text{A-KID}_i)$ for each user to achieve indistinguishability. For k users with n pairs each, the AAnF must store and manage $N = n \times k$ shuffled pairs, significantly increasing storage requirements and operational complexity. Their experimental results show computation costs scaling linearly with n, reaching 78.206 ms for $n = 500$, which becomes problematic at scale.

In contrast, E2E-AKMA eliminates centralized key storage through a distributed trust model using two-party oblivious signatures. Neither the UE nor HN can independently generate valid signatures, removing single points of failure. Our approach achieves true end-to-end security even under the AAnF compromise, while maintaining only 9.4% overhead compared to standard AKMA and requiring no complex key management infrastructure.

	5G AKMA	**AKMA+**	**Ours**
Compromise UE storage to impersonate user (SIM card safe)	○	◑	●
Compromise AAnF to impersonate user	○	◑	●
AAnF without maintaining a secure dynamic database	○	○	●

○: Not satisfied; ●: Satisfied.
◑: Claimed to satisfy this property, but has known attacks.

Table 1: Comparison of three protocols.

Other Related Works. Several studies have addressed privacy concerns in AKMA protocols. Khan et al. [17] analyzed the AKMA protocol's working mechanism and identified linking issues between HN and AF. Their subsequent work [16] introduced a privacy mode for AKMA that achieves unlinkability between HN and AF by routing communication through the UE instead of direct HN-AF connection. However, their scheme requires group operations on SIM card secret keys, preventing deployment on existing devices. Yang et al. [31] conducted a formal analysis of 5G AKMA focusing on authentication and key management, acknowledging the HN-AF privacy issues we address but without providing solutions. Akman et al. [5] proposed Privacy-Enhanced AKMA considering malicious AF scenarios where adversarial AFs intercept UE communications to request keys from HN. They also addressed AF collusion for user account linking by introducing new identifiers to replace Generic Public Subscription Identifiers, achieving AF-unlinkability. However, their scheme still allows HN to determine which AF users are accessing, leaving the HN-side privacy issue unresolved.

Structurally, E2E-AKMA functions as a delegated authentication system, analogous to Single Sign-On (SSO) protocols like OIDC [24] and SAML [23]. In this model, the UE, AF, and HN correspond to the User, Relying Party (RP), and Identity Provider (IdP), respectively. However, unlike typical web-based SSO, E2E-AKMA is rooted in the mobile network trust infrastructure. It leverages the existing 5G AKA protocol and SIM-based credentials to bootstrap trust, enabling the HN (IdP) to assist in user authentication while ensuring end-to-end security and privacy.

2 AKMA Protocol Overview

2.1 Original AKMA Design Philosophy

The 5G AKMA protocol, as specified in 3GPP TS 33.535 [4], adopts a delegated authentication architecture that enables AFs to leverage mobile network operator authentication capabilities without requiring direct credential management.

Core Design Principles. As illustrated in Figure 1, the AKMA protocol follows a three-party model involving UE, HN, and AF. The design leverages the existing 5G authentication infrastructure through a centralized AAnF that serves as the trust anchor for key derivation and distribution.

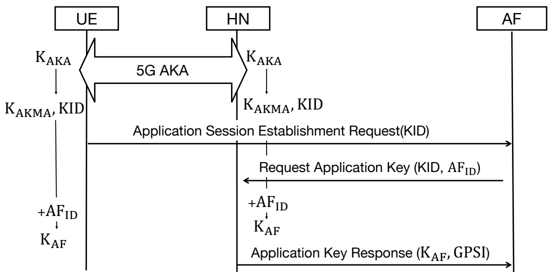

UE
HN
AF
K_{AKA}
5G AKA
K_{AKA}
K_{AKMA}, KID
K_{AKMA}, KID
Application Session Establishment Request(KID)
Request Application Key (KID, AF_{ID})
$+AF_{ID}$
K_{AF}
$+AF_{ID}$
K_{AF}
Application Key Response (K_{AF}, GPSI)

Figure 1: 5G AKMA Protocol Overview

The protocol workflow proceeds as follows: (1) After successful 5G AKA primary authentication, both UE and AAnF derive the AKMA anchor key K_{AKMA} and its identifier (A-KID); (2) When accessing an AF, the UE presents the A-KID to initiate session establishment; (3) The AF requests the corresponding application key K_{AF} from the AAnF via the Network Exposure Function (NEF); (4) The AAnF derives K_{AF} using K_{AKMA} and AF identifier (AF_ID), then provides it to the AF along with user identity information (SUPI/GPSI); (5) Both UE and AF independently derive the same K_{AF} for secure communication.

Architectural Benefits. This centralized approach offers several advantages: elimination of complex credential databases at AFs, seamless integration with existing 5G infrastructure, and reduced authentication overhead for resource-constrained IoT devices. The protocol enables "single sign-on" functionality where one primary authentication supports multiple application sessions.

2.2 Security and Privacy Challenges

However, the centralized trust model introduces critical vulnerabilities that 3GPP TR 33.835 [3] identifies through seventeen key issues. Five protocol-layer issues pose fundamental threats to AKMA's security foundation.

Security Objectives

Mutual authentication between UE and anchor function (Key Issue #3). The original AKMA lacks robust mutual authentication between UE and AAnF. While 5G AKA provides UE-network authentication, the AAnF operates outside the 5G core network boundary, creating trust gaps. Without proper authentication, unauthorized UEs may access AKMA services, while fake anchor functions can communicate with legitimate UEs, leading to privacy exposure and potential attacks.

Secure communication between UE and application server (Key Issue #6). The AAnF's central role in key derivation creates a single point of failure. All application keys K_{AF} are derived from K_{AKMA} stored at the AAnF, enabling potential interception of UE-AF communications when the AAnF is compromised. The current design provides no forward secrecy guarantees, as compromised anchor keys expose all past and future sessions.

Application key freshness of AKMA (Key Issue #16). AKMA's key derivation mechanism lacks proper freshness guarantees. Application keys derived using static parameters (K_{AKMA}, AF_{ID}) remain constant across multiple sessions between the same UE-AF pair, enabling replay attacks where attackers who obtain a previously used K_{AF} can masquerade as legitimate UEs towards Application Functions.

Privacy Objectives

User privacy (Key Issue #5). The protocol directly exposes Subscription Permanent Identifiers (SUPI) to Application Functions, creating significant privacy risks. The SUPI represents sensitive information that can be exploited to identify and track individual subscribers, while operators need SUPI access for service provision, creating a fundamental tension between privacy and functionality.

Protecting subscriber's personal information in control and data traffic (Key Issue #7). The A-KID serves as a persistent identifier that enables both AAnFs and AFs to track user authentication behaviors. Privacy-sensitive content in control and data traffic messages can be exploited by attackers to identify subscribers through correlation attacks, while the leakage of AKMA key identifiers enables linking users across different network sessions.

3 Modeling Security and Privacy

3.1 Security Assumptions and Threat Model

The design of robust AKMA systems requires careful consideration of which participants may be compromised in practical deployments. Drawing from AKMA+ [32], we identify the primary threat vectors and establish our security assumptions:

Threat Vectors. We consider three main categories of threats: (1) *UE threats* including malware infiltration and malicious UEs attacking other entities. Note that while UEs operate in general-purpose software environments that may not provide secure key storage, the SIM card is typically regarded as trusted hardware capable of securely storing AKA keys and establishing authenticated relationships with the HN; (2) *AAnF compromise* where AAnFs may be malicious due to deployment outside secure 5G core networks; and (3) *AF threats* including data leakage and collusion between multiple AFs to correlate user behaviors.

HN as a Communication Facilitator. In accordance with 3GPP TS 33.501 [1] and AKMA+ [32], we assume that core 5G network functions (UDM, AUSF, NEF) remain secure within the protected core network perimeter and reliably provide authentic SUPI information. To simplify our security analysis, we further assume that the HN does not directly participate in application-layer key derivation. Instead, the HN functions as a secure intermediary that facilitates communication while preserving end-to-end security properties between UEs and AAnFs.

Untrusted AAnF. In contrast to core network functions, we assume that AAnFs are potentially malicious. This assumption arises from the architectural reality that AAnFs may be deployed outside the secure perimeter of the 5G core network, rendering them more vulnerable to compromise [32]. Consequently, malicious AAnFs could inject messages that appear cryptographically valid to both UEs and AFs. Additionally, they may collect and expose sensitive user privacy information, such as SUPIs and access patterns, exploit their privileged position in key management to perform man-in-the-middle attacks, and collaborate with other network entities to undermine user privacy and security. Unlike the approach taken in [32], we allow for the possibility that an AAnF may not adhere to established protocols. This consideration is based on the assumption that an adversary capable of compromising an AAnF might gain access to its authentication secrets, effectively enabling them to impersonate a legitimate AAnF within the system. As a result, the attacks we consider are more realistic and reflective of potential real-world threats.

Secure Communication Channels. The proposed protocol operates under the assumption that authenticated and confidential communication channels exist between the participating entities. Specifically:

(1) Before the protocol is executed, UEs are provisioned with SIM cards issued by their respective HNs. These HNs possess comprehensive knowledge of the SIM card contents, including the SUPI, which is integral to AKMA computations. Utilizing the 5G AKA protocols, UEs and HNs establish bidirectionally authenticated secure channels that ensure both mutual authentication and confidentiality.

(2) It is assumed that secure communication channels exist between the HN core network and the AAnF. Based on the assumptions outlined in points (1) and (2), it is further presumed that UEs and AAnFs can securely communicate in the absence of compromise. Additionally, AAnFs are assumed to be capable of identifying the SUPI of the UEs they interact with, as this identifier is forwarded by the HN.

(3) UEs authenticate AFs by verifying their digital certificates, subsequently establishing secure communication channels using TLS. This process achieves unilateral authentication, wherein the UE authenticates the AF, while also fulfilling the security properties of Authenticated and Confidential Channel Establishment (ACCE) [15, 18].

These assumptions are substantiated by the widespread deployment and rigorous security analysis of 5G AKA protocols [1] and TLS protocols within existing 5G infrastructure.

3.2 Syntax

Building on this threat and communication model, we define the AKMA protocol as a three-party system involving UE, AAnF (facilitated by HN), and AF. The protocol enables UEs and AFs to establish session keys without revealing these keys to the AAnF, while ensuring the AAnF cannot determine which AFs users are accessing.

The AKMA protocol consists of four algorithms: initialization, registration, login, and tracing, denoted by Init, Reg, Login, and Trace. In practical deployments, AAnFs maintain public-private key pairs (PK_{AAnF}, SK_{AAnF}), all participants possess the core network's public key PK_{core}, AFs possess public keys of all AAnFs $\mathcal{PK}_{AAnF}$, and AAnFs securely store master keys K_{AAnF} for key derivation. The Init algorithm generates long-term public parameters including algebraic group parameters, while Reg supports multiple account creation, Login enables multiple authentication sessions per account, and Trace allows the core network to recover SUPIs from AF databases.

Definition 1 (AKMA). *An Authentication and Key Management for Applications (AKMA) scheme is a tuple of interactive protocols,* $\mathsf{AKMA} = (\mathsf{Init}, \mathsf{Reg}, \mathsf{Login}, \mathsf{Trace})$*:*

- $$PP_{AKMA} \leftarrow \mathsf{Init}\big(UE(\emptyset), AAnF(SUPI, SK_{AAnF}, K_{AAnF}, PK_{core}), AF(PK_{AAnF}, PK_{core})\big):$$
  The participants input their respective existing parameters, including the core network's public key, and the three parties jointly generate the public parameter PP_{AKMA}*.*
- $$(K_{UE}, \perp, DB'_{AF}) \leftarrow \mathsf{Reg}\big(UE(\emptyset), AAnF(SUPI, SK_{AAnF}, K_{AAnF}, PK_{core}), AF(PK_{AAnF}, PK_{core}, DB_{AF})\big):$$
  UE outputs K_{UE} *for the* Login *phase. AAnF takes SUPI of the UE,* K_{AAnF}*, and* PK_{core} *as input and outputs* $\perp$*. AF stores UE registration data in* DB_{AF}*.*
- $$(ssk, \perp, ssk) \leftarrow \mathsf{Login}\big(UE(K_{UE}), AAnF(SUPI, K_{AAnF}, PK_{core}), AF(DB_{AF})\big):$$
  UE inputs K_{UE}*, AAnF inputs SUPI,* K_{AAnF}*, and* PK_{core}*, AF*

inputs DB_{AF}. UE and AF output a shared session key ssk, while AAnF outputs $\perp$.

- *(SUPI)$\leftarrow$* Trace*(DB_{AF}, SK_{core}): The core network inputs the AF's database DB_{AF} and its private key SK_{core}, and outputs the corresponding SUPI to enable user traceability for legitimate purposes.*

3.3 Security Model

Building upon the security assumptions and threat model established earlier, we now present a formal cryptographic security model that captures the unique challenges of AKMA protocols under malicious AAnF assumptions. Our model extends traditional authenticated key exchange frameworks to address the specific architectural constraints and privacy requirements of 5G networks.

Modeling Rationale. The formal model must reconcile several competing requirements: (1) *Architectural fidelity* - accurately reflecting 5G network structure where core functions (HN) are trusted but edge functions (AAnF) may be compromised; (2) *Cryptographic rigor* - providing precise security definitions amenable to formal analysis; and (3) *Privacy preservation* - capturing unlinkability requirements against both malicious AAnFs and colluding AFs.

Protocol Participants and Computational Model. We conceptualize the AKMA ecosystem as a network comprising three distinct types of entities, each represented by multiple parties.

- *User Equipment (UE):* Possesses SIM card with long-term key shared with its home network. Executes registration and authentication protocols.
- *AKMA Anchor Function (AAnF):* Potentially malicious entity responsible for key derivation and user-AF authentication. May be deployed outside secure network perimeter.
- *Application Function (AF):* Service provider maintaining user accounts and session states. May collude with other AFs for user tracking.

Each party may have multiple instances, modeled as separate Turing machines $\pi^{i,j}_{Role}$ where $Role \in \{\mathcal{UE}, \mathcal{AANF}, \mathcal{AF}\}$ (note: HN only relays messages), i denotes the account/registration index, and j denotes the session index ($j = 0$ for registration, $j > 0$ for authentication sessions).

To optimize our model, we have designed the 5G Core Network to handle user credentials and SUPI information effectively. Within the framework of our protocol, the 5G Core Network primarily serves as a secure message relay. Except in specific scenarios where tracking user activity is explicitly required, it does not perform computational tasks related to the protocol in routine operations. Consequently, the 5G Core Network is not categorized as a Turing machine.

Communication Model. The model reflects 5G network communication patterns:

UE - AAnF Channel: The 5G AKA protocol ensures mutual authentication and confidentiality between communicating parties. However, it does not provide forward secrecy. This means that adversaries are unable to intercept encrypted communication or impersonate any of the involved entities unless they compromise one of the parties. Nonetheless, the absence of forward secrecy in 5G AKA implies that if the long-term keys of the UE are compromised, it becomes possible to decrypt historical communication transcripts. Consequently, these transcripts can be recovered in the event of long-term key compromise.

UE-AF Channel: The TLS protocol ensures server authentication and ACCE security, allowing the UE to verify the certificates provided by the AF. This guarantees that adversaries cannot intercept encrypted communications or impersonate the AF to the UE. However, adversaries may still have the ability to impersonate the UE to the AF. Furthermore, the forward security property of TLS with ECDHE handshake ensures that the compromise of long-term cryptographic keys does not lead to the exposure of previously transmitted data or historical.

AAnF-AF Channel: Although AAnF-AF communications typically utilize secure network protocols, in our protocol, this particular aspect of communication is not applicable. Therefore, we do not make any assumptions regarding its usage.

Adversarial Capabilities. Adversary $\mathcal{A}$ is a PPT algorithm with access to oracles reflecting realistic attack capabilities:

- *Protocol Initiation:* Adversaries can initiate registration and authentication sessions between UE instances and AFs of their choice.
- *UE Corruption:* Full corruption returns UE's long-term keys and all historical transcripts, while compromise returns only derived keys without long-term secrets.
- *AAnF Access:* Since AAnFs are assumed inherently malicious, adversaries have complete access to all AAnF state including key derivation materials and transcripts.
- *AF Corruption:* Full corruption returns AF's complete database and session states, while breach returns only user databases without control.
- *Session Key Revelation:* Adversaries can obtain session keys for specific protocol instances.
- *Message Injection:* Adversaries can send arbitrary messages to honest instances on behalf of corrupted parties.

Security Properties. We define security properties to formalize the requirements identified in our threat analysis.

Authentication. This property ensures that honest UE and AF instances can only accept protocol execution with their intended partners. We use a "Trust on First Use" model where initial registration establishes authentic binding between UE and AF, and subsequent authentications must preserve this binding even against malicious AAnFs. The adversary wins if they can cause honest instances to accept with incorrect partners or without unique partnerships.

Key-Indistinguishability. This property guarantees that session keys derived by honest UE-AF pairs are computationally indistinguishable from random keys, even when adversaries have access to malicious AAnFs and can corrupt non-target parties. The adversary is challenged to distinguish between real and random session keys under carefully defined freshness conditions that ensure the target session remains secure despite AAnF compromise.

AF-Unlinkability. This property ensures that malicious AFs cannot link user accounts across different services. Within this framework, even if two honest UE devices are chosen, and their previously associated registration account information along with their SUPI are already known, a malicious AF is still unable to determine which honest UE is engaged in the registration process.

*AAnF-Unlinkability and AAnF-Indistinguishability.*These properties are essential for safeguarding user privacy in the context of 5G deployments, particularly against one of the most plausible threats: malicious AAnFs attempting to construct user profiles across multiple services. The principle of *AAnF Unlinkability* ensures that a single UE can register multiple accounts through its HN without the HN being able to determine which account corresponds to which AF. In the experimental setup, this property is characterized by allowing an adversary to select two honest UE instances associated with the same UE and two honest AF entities, while ensuring that the adversary is unable to discern which UE instance interacts with which AF entity. Furthermore, the principle of *AAnF Indistinguishability* ensures that even if a malicious AAnF has access to the registration information stored on the AF, it cannot associate or link the SUPI corresponding to the account. These two properties guarantees a robust level of privacy by preventing adversaries from inferring sensitive identity information.

Traceability. This property ensures that the HN is capable of mapping public account identifiers used by AFs to user SUPIs for legitimate purposes, such as lawful interception, billing, and emergency services. Our model for traceability operates under the assumption that all participating entities (e.g., AAnF, AFs, etc.) adhere to the specified protocols. This assumption is reasonable because, unlike other properties such as privacy, a breach of traceability does not immediately result in information leakage. Instead, it is sufficient to retain the capability to hold entities accountable retrospectively when failures in traceability are identified. In such cases, regulatory authorities can initiate out-of-band investigative procedures, including equipment audits, compliance verification, and the imposition of operational restrictions.

This framework enables rigorous security proofs while capturing the unique challenges of 5G network architectures. The formal oracle definitions and security experiments are provided in Appendix D, and the subsequent protocol design will demonstrate how cryptographic techniques can achieve these security properties under our threat model.

3.4 Correspondence to 3GPP Key Issues.

Our security models directly address the key issues identified in 3GPP's AKMA security analysis [3]:

- **Key Issue #3 - Mutual authentication between UE and anchor function:** In 5G AKMA, UE does not communicate directly with AAnF; messages pass through the AF, causing no explicit mutual authentication between UE and AAnF. In our framework, UE and AAnF communicate directly via the HN. UE–HN authentication uses the AKA protocol, and secure channels exist between AAnF and HN. Thus, mutual authentication between UE and AAnF is naturally achieved without extra mechanisms.
- **Key Issue #6 - Secure Communication Between UE and Application Server:** Our *Authentication* property ensures mutual authentication between legitimate UE and AF instances with their intended partners, resisting man-in-the-middle attacks even if the AAnF is compromised. Additionally, our *Key-Indistinguishability* property guarantees session keys generated by legitimate UE-AF pairs are computationally indistinguishable from random keys, ensuring compromised anchor keys do not jeopardize session confidentiality.
- **Key Issue #16 - Application Key Freshness in AKMA:** Our proposed *Key-Indistinguishability* properties provide forward security. In contrast to the current AKMA design, where application keys derived from static parameters (K_{AKMA}, AF_{ID}) remain constant across sessions, our security framework ensures enhanced protection. Specifically, even if an adversary compromises a participant after previous sessions have concluded, the session keys from those earlier sessions remain indistinguishable, thereby preserving their confidentiality and integrity.
- **Key Issue #5 - User privacy:** Our proposed properties, *AF-Unlinkability*, *AAnF-Unlinkability*, and *AAnF-Indistinguishability*, address key user privacy concerns. These ensure SUPIs are not exposed to AFs and prevent colluding AFs from identifying, tracking, or profiling subscribers across services. This resolves the conflict between user privacy and system functionality. Meanwhile, our *Traceability* property allows the trusted core network to link public account identifiers used by AFs back to user SUPIs. This balance preserves anonymity while enabling essential network operations.
- **Key Issue #7 - Protecting subscriber's personal information in control and data traffic:** Our *AF-Unlinkability* properties specifically prevent the exploitation of persistent identifiers like A-KID for tracking user authentication behaviors. These properties ensure that privacy-sensitive content in control and data traffic cannot be used for correlation attacks, and prevent the leakage of AKMA key identifiers that could enable linking users across different network sessions.

4 Two-Party Oblivious Signature

In this section, we introduce a new and innovative concept called the two-party oblivious signature, abbreviated as 2POS. 2POS is similar to the two-party threshold signature, but it differs in that only one party has access to the final signature. A more important difference is that P_1 possesses multiple secret keys while P_2 only has one in our scenario. Hence 2POS additionally ensures that P_2 cannot distinguish which secret key P_1 is used to generate a signature, a property we refer to as P_2-indistinguishability. This new feature is essential for the identity unlinkability of the E2E-AKMA protocol.

4.1 Definition of 2POS

Syntax. In the 2POS scheme, there are two participants called P_1 and P_2, the protocol is divided into four phases: Setup, Key Generation, Signing, and Verification. In the Setup phase, P_1 and P_2 jointly generate the public parameters of 2POS, and P_2 generates its long-term SK_2 and PK_2. During the Key Generation, P_1 combines its key with the long-term PK_2 of P_2 to produce the 2POS public key. In the Signing phase, P_1 interacts with P_2 to sign a message chosen by P_1, and outputs the signature, while P_2 produces no output. The verification process follows standard signature verification.

Definition 2. *A Two-Party Oblivious Signature (2POS) scheme denoted as* 2POS=(Setup, KeyGen, Sign, Ver) *consists of three interactive protocols and a* PPT *algorithm.*

- $((PP_{2POS}, PK_2/\perp), (PP_{2POS}, PK_2, SK_2)) \leftarrow \mathsf{Setup}\ (P_1(\emptyset), P_2(\emptyset))$*: Generate public parameter and long-term key.*
- $(SK_1, PK_S) \leftarrow \mathsf{KeyGen}(PK_2)$*: The key generation phase is locally computed by P_1, taking PK_2 as input and outputting the private key share of P_1 and the 2POS signature public key PK_S.*
- $(\sigma_{2POS}, \perp) \leftarrow \mathsf{Sign}(P_1(m, SK_1), P_2(m, SK_2))$*: P_1 and P_2 input their respective secret shares, P_1 outputs the signature.*
- $0/1 \leftarrow \mathsf{Ver}(m, \sigma_{2POS}, PK_S)$*: Identical to the standard signature verification. Input m, σ_{2POS}, PK_S, and output the verification result.*

4.2 Security

Now we define the security of 2POS. Since the security requirements of P_1 and P_2 are inconsistent in the 2POS protocol, we need to consider the cases of P_1 being malicious and P_2 being malicious, respectively. The detailed definition of the 2POS security model and the security proof are provided in our full version [19].

P_1-Unforgeability. Informally, P_1-Unforgeability ensures that even with access to the private key share of P_1, the adversary cannot forge a signature without interacting with the honest P_2. This definition corresponds to the UE being honest during the registration phase of E2E-AKMA. This approach is as follows: the challenger generates keys honestly and gives the adversary $\mathcal{A}$ the private key share of P_1, the long-term public key of P_2, and the public key of 2POS. The adversary interacts with P_2, selects a message m to produce the signature, and must produce a signature for a new message m^* not previously queried.

P_2-Unforgeability. Informally, P_2-Unforgeability ensures that even if the adversary $\mathcal{A}$ acts as P_2, as long as P_1 is honest, no signature can be obtained. Unlike P_1-Unforgeability, P_2-Unforgeability allows $\mathcal{A}$ to act maliciously during the Setup phase of 2POS, capturing that even if HN misbehaves during the registration phase of E2E-AKMA, valid signatures cannot be forged. The approach is as follows: $\mathcal{A}$ interacts with the challenger $\mathcal{C}$ to complete the Setup phase. $\mathcal{C}$ generates the public key for $2POS$ and provides it to $\mathcal{A}$. $\mathcal{A}$ may continue interacting with the $\mathcal{C}$ to perform the signing phase, but it cannot obtain any valid signature.

P_2-Indistinguishability. Informally, P_2-Indistinguishability ensures that a malicious P_2 cannot distinguish which P_1 party it is interacting with. This approach is as follows, the challenger selects two honest P_1 party, Π_2^0 and Π_2^1, and chooses $b \leftarrow \{0,1\}$. P_2 performs Setup and Sign with these two parties, and guesses the value of b chosen by the adversary.

Untraceable. Informally, Untraceable ensures that the 2POS public key appears random to an external adversary, preventing identification of the P_1-P_2 pair that generated it. This approach is as follows: $\mathcal{A}$ selects two groups of honest P_1 and P_2 parties for Setup, multiple KeyGen, and Sign operations. In the test session, the challenger selects $b \leftarrow \{0,1\}$ to decide which group performs KeyGen and Sign. Finally, $\mathcal{A}$ succeeds if it correctly guesses the value of b.

4.3 Construction of 2POS

The 2POS signature scheme represents an advancement over traditional threshold ECDSA, incorporating two enhancements. First, P_2 cannot independently generate a valid signature, even after arbitrary interactions with P_1. Second, P_2's computational perspective can be distinguished based on P_1's distinct secret keys. Building upon these principles, we design the 2POS scheme using Paillier encryption [25] as a cryptographic component. The roles of P_1 and P_2 are deliberately designed to be asymmetric. Specifically, the messages associated with sk_1, which are transmitted by P_1, are encrypted using a homomorphic encryption scheme, such as the Paillier cryptosystem. The confidentiality ensured by this encryption guarantees that the information visible to P_2 remains indistinguishable, even when different secret keys sk_1 are utilized. Moreover, since P_2's functionality is restricted to performing homomorphic operations on the encrypted data, it is inherently incapable of deriving the final signature. This limitation arises from P_2's lack of access to the decryption key, which is essential for deriving the final signature.

The ideal zero-knowledge functionality is denoted by $\mathcal{F}_{\text{zk}}$, while the committed non-interactive zero-knowledge functionality is represented as $\mathcal{F}_{\text{com-zk}}$. The encryption and decryption processes of the Paillier cryptosystem are symbolized by Enc and Dec, respectively. Detailed explanations of the ideal functionalities, as well as the Paillier encryption scheme [25], can be found in Appendix. The construction of the 2POS protocol is outlined as follows.

Setup. P_2 independently selects $x_2 \leftarrow\!\!\$\, \mathbb{Z}_q$ as their long-term private key, denoted as SK_2, and computes their corresponding long-term public key PK_2 as $Q_2 = x_2 \cdot G$.

Subsequently, P_2 transmits the public key PK_2 to P_1, accompanied by a zero-knowledge proof π that demonstrates possession of the associated private key SK_2. Upon receiving this information, P_1 verifies the validity of the proof and securely stores PK_2 locally.

Key Generation. P_1 locally selects $x_1 \leftarrow\!\!\$\, \mathbb{Z}_q$ as the private key share SK_1, and calculates the 2POS public key $Q = x_1 \cdot Q_2$ based on the long-term public key Q_2 of P_2.

Sign. At the beginning of the signing phase, party P_1 has x_1,Q and the message m, P_2 has x_2.

- For the first message, P_1 chooses $k_1 \leftarrow\!\!\$\, \mathbb{Z}_q$, computes $R_1 = k_1 \cdot G$ and sends (com-prove, sid||1, R_1, k_1) to $\mathcal{F}^{R_{DL}}_{\text{com-zk}}$.
- For the second message, P_2 receives (proof-receipt, sid||1) from $\mathcal{F}^{R_{DL}}_{\text{com-zk}}$, chooses a random $k_2 \leftarrow \mathbb{Z}_q$, computes $R_2 = k_2 \cdot G$ and sends (prove, sid||2, R_2, k_2) to $\mathcal{F}^{R_{DL}}_{\text{zk}}$.
- For the third message, P_1 receives (proof, sid||2, R_2) form $\mathcal{F}^{R_{DL}}_{\text{zk}}$, generates a new Paillier key-pair (pk, sk) and computes $C_{HE} = \mathsf{Enc}_{pk}(x_1)$. P_1 sends C_{HE}, pk to the P_2 and sends the (decom-proof, sid||1) to $\mathcal{F}^{R_{DL}}_{\text{com-zk}}$ for decommit.
- For the fourth message, P_2 receives (decom-proof, sid||1, R_1) from $\mathcal{F}^{R_{DL}}_{\text{com-zk}}$ and computes $R = k_2 \cdot R_1$ and $m' = H(m)$ for the hash function $H(\cdot)$. Denote $R = (r_x, r_y)$, $r = r_x$ mod q. P_2 chooses $\rho \leftarrow\!\!\$\, \mathbb{Z}_{q^2}$, $\tilde{r} \in \mathbb{Z}^*_N$ (verifying that gcd($\tilde{r}$,N)=1), and computes $C_1 = \mathsf{Enc}_{pk}(\rho \cdot q + [k_2^{-1} \cdot m' \bmod q; \tilde{r}])$. Then P_2 computes $v = k_2^{-1} \cdot r \cdot x_2 \bmod q$, $C_2 = v \odot C_{HE}$ and $C_3 = C_1 \oplus C_2$ and sends C_3 to P_1.
- Finally, P_1 computes r in the same way as P_2 and $s' = \mathsf{Dec}_{sk}(C_3)$, $s'' = k_1^{-1} \cdot s' \bmod q$. P_1 sets $s = min\{s'', q - s''\}$, ensuring that the signature is always the smaller of the two possible values. P_1 verifies (r, s) is a valid signature with public key Q. If yes it outputs the signature (r, s). In this way, P_1 and P_2 can jointly produce a standard ECDSA signature of Q.

Verification. The verification algorithm is the same as the standard ECDSA verification algorithm.

We included the theorems on signature security, along with the corresponding security analysis, in our full version [19].

5 E2E-AKMA

Following the high-level construction outlined in Section 1.2, we now present the detailed cryptographic specification of E2E-AKMA.

5.1 Construction Details

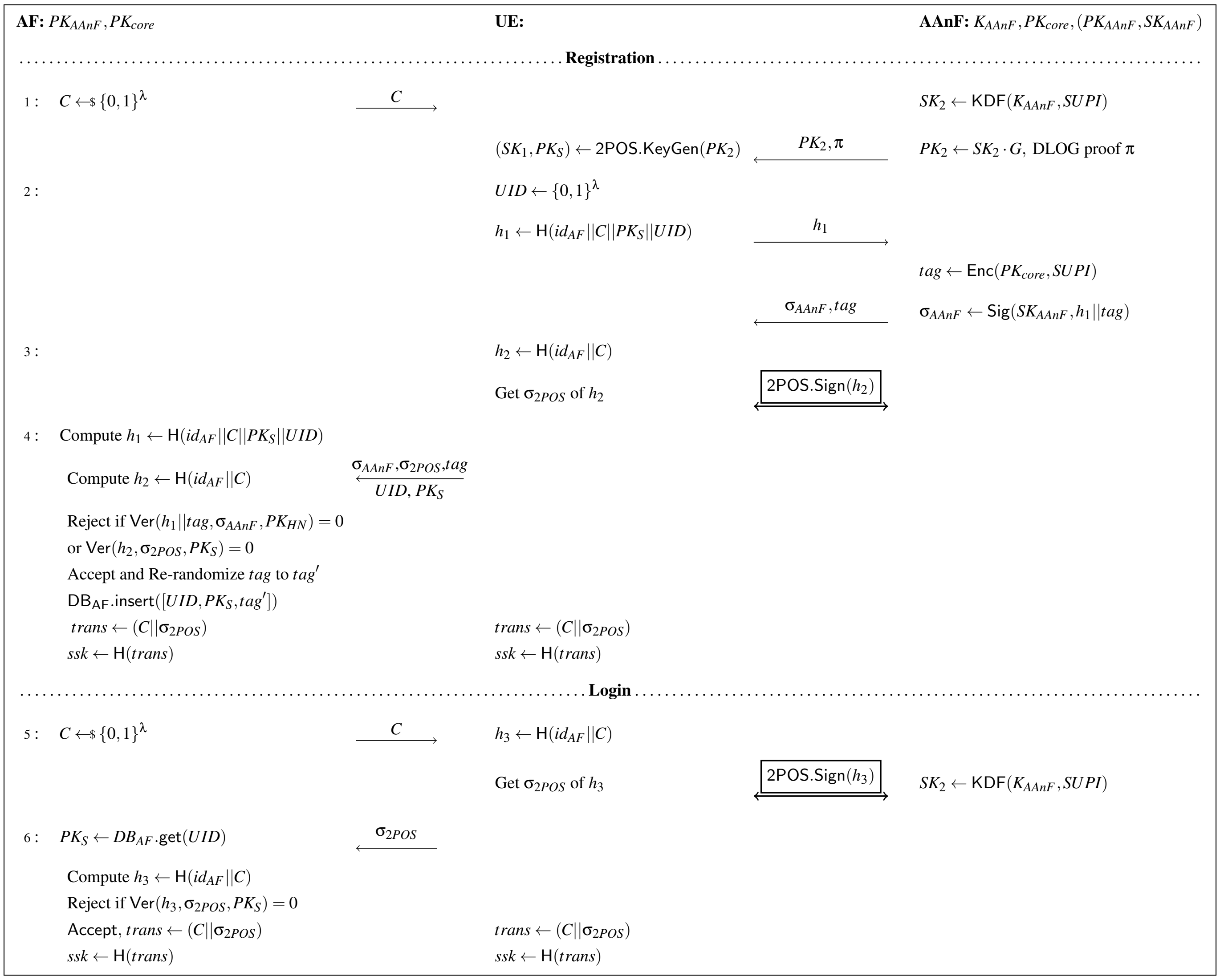
Before the commencement of the protocol, each AAnF must independently generate a unique public-private key pair, denoted as (PK_{AAnF}, SK_{AAnF}), which is specifically designated for signature operations. This cryptographic key pair is integral to guaranteeing the authenticity of the registration process for a UE, as it ensures that the process has been performed using the services provided by the corresponding AAnF. each AF is provisioned with the public keys of all AAnFs, collectively represented as PK_{AAnF}. Moreover, AAnFs securely store master keys, denoted as K_{AAnF}, which are employed for cryptographic key derivation processes.

Building upon this foundational framework, we propose the design of the E2E-AKMA as follows.

The Init Phase. Generate the parameters of E2E-AKMA, including those for signature and encryption algorithms.

The Registration Phase.

1. In the registration phase, after interacting with the $AAnF$, the UE registers an account with the AF and logs in. After UE requests registration with AF, AF sends a challenge C to UE. AAnF uses the master key K_{AAnF} with $SUPI$ to generate SK_2 for 2POS and calculates PK_2, while providing a zero-knowledge proof π of knowing the SK_2 corresponding to PK_2. After verifying π, UE executes the key generation phase of 2POS, $(SK_1, PK_S) \leftarrow 2POS.\mathsf{KeyGen}(PK_2)$. SK_1 serves as K_{UE} in E2E-AKMA, PK_S serves as the public key of the account registered by the UE.
2. UE generates an account ID UID for this registration, which serves as a part of the username. UE computes the hash value h_1 for id_{AF}, C, PK_S, and UID, requests AAnF to use its private key to sign h_1, obtaining σ_{AAnF}. This illustrates that the key generation process is carried out through interaction with the AAnF. Additionally, AAnF encrypts the SUPI using the core network's public key via a Re-randomizable encryption scheme (described in Appendix A): $tag \leftarrow \text{Enc}(PK_{core}, SUPI)$, and includes this encrypted tag in the signature: $\sigma_{AAnF} \leftarrow \text{Sig}(SK_{AAnF}, h_1||tag)$.
3. The UE computes the hash h_2 to id_{AF} and C, UE and AAnF execute the signing phase of 2POS to generate a signature σ_{2POS}, which can be verified by PK_S. In the implementation, steps 2 and 3 can be combined to reduce the communication rounds.
4. The UE sends the two signatures σ_{AAnF}, σ_{2POS}, UID, and PK_S to the AF. AF verifies the signatures by computing $h_1 \leftarrow H(id_{AF}||C||PK_S||UID)$ and $h_2 \leftarrow H(id_{AF}||C)$, then checking $\text{Ver}(h_1||tag, \sigma_{AAnF}, PK_{AAnF}) = 1$ and $\text{Ver}(h_2, \sigma_{2POS}, PK_S) = 1$. Upon successful verification, AF performs a critical security step: it re-randomizes the encrypted SUPI tag to obtain $tag' \leftarrow \text{Re-randomize}(tag)$. This re-randomization acts as a second layer of encryption, ensuring that even if AAnF provides malicious ciphertexts, they become indistinguishable from random, preventing AAnF from tracking users through this field.



AF: PK_{AAnF}, PK_{core}
UE:
AAnF: $K_{AAnF}, PK_{core}, (PK_{AAnF}, SK_{AAnF})$
Registration
1 : $C \leftarrow\$ \{0,1\}^{\lambda}$
C
$SK_2 \leftarrow \mathsf{KDF}(K_{AAnF}, SUPI)$
$(SK_1, PK_S) \leftarrow \mathsf{2POS.KeyGen}(PK_2)$
PK_2, π
$PK_2 \leftarrow SK_2 \cdot G$, DLOG proof π
2 :
$UID \leftarrow \{0,1\}^{\lambda}$
$h_1 \leftarrow \mathsf{H}(id_{AF}||C||PK_S||UID)$
h_1
$tag \leftarrow \mathsf{Enc}(PK_{core}, SUPI)$
σ_{AAnF}, tag
$\sigma_{AAnF} \leftarrow \mathsf{Sig}(SK_{AAnF}, h_1||tag)$
3 :
$h_2 \leftarrow \mathsf{H}(id_{AF}||C)$
$\boxed{\mathsf{2POS.Sign}(h_2)}$
Get σ_{2POS} of h_2
4 : Compute $h_1 \leftarrow \mathsf{H}(id_{AF}||C||PK_S||UID)$
Compute $h_2 \leftarrow \mathsf{H}(id_{AF}||C)$
$\sigma_{AAnF}, \sigma_{2POS}, tag$
UID, PK_S
Reject if $\mathsf{Ver}(h_1||tag, \sigma_{AAnF}, PK_{HN}) = 0$
or $\mathsf{Ver}(h_2, \sigma_{2POS}, PK_S) = 0$
Accept and Re-randomize tag to tag'
$\mathsf{DB_{AF}.insert}([UID, PK_S, tag'])$
$trans \leftarrow (C||\sigma_{2POS})$
$ssk \leftarrow \mathsf{H}(trans)$
$trans \leftarrow (C||\sigma_{2POS})$
$ssk \leftarrow \mathsf{H}(trans)$
Login
5 : $C \leftarrow\$ \{0,1\}^{\lambda}$
C
$h_3 \leftarrow \mathsf{H}(id_{AF}||C)$
Get σ_{2POS} of h_3
$\boxed{\mathsf{2POS.Sign}(h_3)}$
$SK_2 \leftarrow \mathsf{KDF}(K_{AAnF}, SUPI)$
6 : $PK_S \leftarrow DB_{AF}.\mathsf{get}(UID)$
σ_{2POS}
Compute $h_3 \leftarrow \mathsf{H}(id_{AF}||C)$
Reject if $\mathsf{Ver}(h_3, \sigma_{2POS}, PK_S) = 0$
Accept, $trans \leftarrow (C||\sigma_{2POS})$
$ssk \leftarrow \mathsf{H}(trans)$
$trans \leftarrow (C||\sigma_{2POS})$
$ssk \leftarrow \mathsf{H}(trans)$

Figure 2: The E2E-AKMA Construction. The communication between the UE and the AAnF is facilitated through the 5G AKA protocol, as well as being relayed by the HN. The interaction between the UE and the AF is secured using the TLS protocol.

AF then stores the UID, PK_S, and re-randomized tag' in its database for the login phase, and computes the session key using the transcript.

The Login Phase.

5. The login phase is similar to the registration phase. First, the UE submits a login request to the AF on behalf of UID, and the AF sends a challenge C to the UE. UE generates $h_3 \leftarrow H(id_{AF}||C)$ and performs the 2POS signing phase with AAnF as in the registration step 3 to obtain σ_{2POS}.
6. UE sends the σ_{2POS} to the AF. AF retrieves PK_S using the UID and computes the hash $h_3 \leftarrow H(id_{AF}||C)$ and verifies the signature $\mathrm{Ver}(h_3, \sigma_{2POS}, PK_S) = 1$. Upon successful verification, the UE and AF utilize the hash value of their transcript as the session key $ssk \leftarrow H(\text{trans})$ where trans $\leftarrow (C||\sigma_{2POS})$.

The Trace Algorithm. The trace algorithm enables the core network to retrieve the true SUPI of a registered user from the AF's database for legitimate purposes, such as law enforcement compliance. Using its private key SK_{core} and the re-randomized encrypted tag tag' from the AF's database, the core network decrypts the tag as: $SUPI \leftarrow \mathrm{Dec}(SK_{\text{core}}, tag')$ recovering the user's true identity.

5.2 Security Analysis.

We present the underlying intuition of security. A more detailed security proof is in our full version [19].

Theorem 1 (Authentication)**.** *The E2E-AKMA satisfies the Authentication as defined in Definition 4, assuming 2POS is* *correct, AFs avoid identical challenges, and 2POS is P_1-UNF and P_2-UNF.*

Proof sketch: The protocol ensures mutual authentication between UE and AF through the 2POS signature scheme. Honest execution is guaranteed by the correctness of 2POS, while adversarial attempts to impersonate either party would require forging signatures, which contradicts the unforgeability properties (P_1-UNF and P_2-UNF) of the 2POS scheme. The binding of AF identity to challenges prevents replay attacks across different services.

Theorem 2 (Key-indistinguishability)**.** *Consider the hash function as a random oracle. The E2E-AKMA satisfies the Key-indistinguishability as defined in Definition 5, assuming 2POS is correct, AF avoids identical challenges, 2POS is P_1-UNF and P_2-UNF, RO outputs differ for distinct inputs, and the guessing the challenge is negligible.*

Proof sketch: Session keys are derived from protocol transcripts using a random oracle (hash function). Since the adversary cannot predict the random challenge C generated by honest AFs, and the 2POS signature σ_{2POS} appears random without knowledge of the signing keys, the resulting session key $ssk = H(C||\sigma_{2POS})$ is indistinguishable from random to any external observer.

Theorem 3 (AAnF-unlinkable)**.** *The E2E-AKMA satisfies the AAnF-Unlinkable as defined in Definition 7 assuming the 2POS satisfies P_2-IND, the hash function H is modeled as a random oracle, and the re-randomizable encryption scheme satisfies malformed ciphertext indistinguishability.*

Proof sketch: The AAnF (acting as P_2 in 2POS) cannot distinguish which public key PK_S a UE is using across different authentication sessions. This follows directly from the P_2-IND property of 2POS, which ensures that P_2's view during the signing protocol reveals no information about the specific key being used by P_1 (the UE).

Theorem 4 (AAnF-indistinguishability)**.** *The E2E-AKMA satisfies the AAnF-Indistinguishability property as defined in Definition 8, assuming the 2POS scheme satisfies the Untraceable property and the re-randomizable encryption scheme satisfies malformed ciphertext indistinguishability.*

Proof sketch: The AAnF is unable to associate the SUPI with the accounts on AF due to the limitations of the information accessible through AF's public data. Specifically, the *Account UIDs* and *Public Keys (PK_S)* appear as random values from the perspective of AAnF, thereby preventing any direct correlation with specific UE identities. Moreover, the *Re-randomized Encrypted Tags (tag')* are re-randomized by AF in such a manner that they remain unlinkable to their original encrypted counterparts, ensuring enhanced privacy and security.

Theorem 5 (AF-unlinkable)**.** *Consider the KDF as a random oracle. The E2E-AKMA satisfies the AF-Unlinkable as defined in Definition 6 assuming the 2POS is Untraceable and the re-randomizable encryption scheme is IND-CPA secure.*

Proof sketch: Different AFs cannot link accounts belonging to the same UE, even through collusion. This property stems from the Untraceable property of 2POS, which ensures that multiple public keys PK_S generated by the same UE for different AFs appear completely independent and unlinkable to any external observer, including the AFs themselves.

Theorem 6 (Traceability)**.** *The E2E-AKMA satisfies the Traceability property as defined in Definition 9, assuming the signature scheme used by AAnF is existentially unforgeable under chosen message attacks (EUF-CMA) and the re-randomizable encryption scheme is correct.*

*Proof sketch:*The key insight is that traceability is guaranteed by the unforgeability of the AAnF signature σ_{AAnF} which commits to the encrypted SUPI. Since the AF only accepts registrations with valid AAnF signatures, and honest AAnFs always include the correct encrypted SUPI in their signatures, malicious UEs cannot prevent the storage of their true SUPI in the AF's database.

6 Implementation and Evaluation

In this section, we assess the performance of E2E-AKMA in real-world deployment scenarios.

Device Configuration. The experiments used real SIM cards to closely reflect real-world scenarios, ensuring our evaluation considers the actual computational and communication overhead of the E2E-AKMA protocol on the UE. This validates the protocol's compatibility with existing SIM card infrastructure and its feasibility for deployment without hardware changes. Performance testing was conducted on an iPhone 15 (A16 processor, real SIM card, iOS 18.1) [2]. The Golang implementation is built as a linkable library for Swift on iOS.

Due to lack of mobile network operator privileges, physical base stations were inaccessible. Instead, the HN and AF components were implemented on cloud servers and communicated with the UE via a real 5G network. Both HN and AF demo servers were deployed as cloud services with 8-core Intel Xeon E5-2620 v4 CPUs, 32GB RAM, and located in the same data center. While this setup does not replicate physical base station infrastructure, it effectively models the protocol's computational and communication processes. Similar approaches are common in academic research when operator infrastructure access is restricted [32].

5G AKMA and AKMA+ involve interactions between the AF and HN, so we measured the latency between the three

[2]Our solution also supports eSIMs, which securely store sensitive user information within the eUICC module, ensuring secure communication with the HN like traditional SIM cards [27].

parties. UE connects to the server through the 5G cellular network (i.e. 5G AKA), while the maximum network bandwidth of the HN and AF is 100 Mbps. The latency is approximately 20 ms between the UE and HN, 20 ms between the UE and AF, and 10 ms between the HN and AF. We generate self-signed certificates for the HN and AF using RSA 4096 signatures, which are enough to simulate the TLS channel.

Execution Time. In Table 2, the execution times for individual registration and login operations of the E2E-AKMA protocol are presented. The table provides a detailed breakdown of the communication time and local computation time, both of which have been optimized to a reasonable extent. The parameters and performance metrics of 2POS, along with the communication size of E2E-AKMA, are detailed in Appendix G. To enhance clarity, a double-headed arrow is used to represent the exchange of interactive messages between two participants, the number inside the circle indicating the total number of message rounds exchanged during the operation.

To evaluate the performance overhead introduced by our E2E-AKMA protocol, we conducted an experimental comparison with the 5G AKMA and AKMA+ protocols under identical experimental conditions. The results of our empirical analysis indicate that the E2E-AKMA protocol incurs a moderate overhead of 9.4% (+40.11 ms) compared to the 5G AKMA protocol, while compared with AKMA+, it increased by 9.3% (+39.69ms).

Research shows humans typically do not perceive latency below 100 ms [22]. The 40.11 ms introduced by E2E-AKMA is well within this threshold. Studies in HCI [10] and industry practices [9] confirm such minor latency increases are imperceptible, making E2E-AKMA's impact negligible.

Table 2: The Runtime of the Register and Login phase.

	Registration	Runtime	Total (ms)
Communicate with HN and AF	AF ↔ UE ①	180.87	702.80
	HN ↔ UE ①	198.97	
	HN ↔ UE ②	91.14	
	HN ↔ UE ③	143.69	
	AF ↔ UE ②	88.13	
Local computation time of UE	HN ↔ UE ① to HN ↔ UE ②	1.18	90.15
	HN ↔ UE ② to HN ↔ UE ③	10.85	
	HN ↔ UE ③ to AF ↔ UE ②	78.12	
Total			792.95
	Login	Runtime	Total (ms)
Communicate with HN and AF	AF ↔ UE ①	55.86	355.04
	HN ↔ UE ①	148.18	
	HN ↔ UE ②	148.18	
	AF ↔ UE ②	74.85	
Local computation time of UE	HN ↔ UE ① to HN ↔ UE ②	2.43(μs)	109.08
	HN ↔ UE ② to HN ↔ UE ③	30.66	
	HN ↔ UE ③ to AF ↔ UE ②	78.41	
Total			464.12 5G AKMA runtime: 424.01 AKMA+ runtime: 424.43

The latency is approximately 20 ms between the UE and HN, 20 ms between the UE and AF, and 10 ms between the HN and AF.

Computation cost on the server side. In Table 3, we present an evaluation of the capability of HN and AF to handle high-user traffic. The table provides the number of requests that HN and AF can process per second in multithreaded mode. The test focuses on the efficiency of request processing on the server side, excluding network transmission time.

Although the throughput of the second message exchanged between HN and UE during the login phase is 84.27, in practical deployments, performance can be significantly improved through high-performance servers and dedicated hardware accelerators, which are commonly used in hardware security modules to optimize public key computations and greatly enhance throughput. Therefore, this will not become a performance bottleneck for E2E-AKMA.

Table 3: The Throughput of the HN and AF.

Registation	**Throughput (Requests/s)**	**Login**	**Throughput (Requests/s)**
AF ↔ UE ①	4329.07	AF ↔ UE ①	4776.02
HN ↔ UE ①	290.66	HN ↔ UE ①	319.08
HN ↔ UE ②	286.69	HN ↔ UE ②	84.27
HN ↔ UE ③	82.70	AF ↔ UE ②	581.65
AF ↔ UE ②	481.05		

In practice, side-channel countermeasures may introduce additional overheads. AAnF is at risk of DoS attacks, a situation also present in the original AKMA protocol, which can be mitigated by measures such as load balancing, hardware accelerators, and rate-limiting against malicious SUPIs via the 5G-AKA protocol.

7 Conclusion

The E2E-AKMA protocol proposed in this work effectively addresses the security and privacy challenges identified by 3GPP [3]. In comparison to existing solutions, the proposed scheme guarantees end-to-end communication security, even in scenarios where the AAnF is compromised. Moreover, the E2E-AKMA protocol offers notable advantages in terms of deployment feasibility when compared to the scheme presented in [32]. Specifically, it eliminates the requirement for maintaining a secure data repository, thereby simplifying implementation and reducing associated costs.

Acknowledgments

The authors express their profound appreciation to the shepherd and the anonymous reviewers for their constructive feedback and insightful suggestions, which have played a pivotal role in improving the quality and rigor of this work. Additionally, the authors extend their heartfelt thanks to Meihui Chen and Yanfei Guo from Huawei for their invaluable guidance and support in navigating this research domain. This research was supported by the National Key R&D Program of China (Grant No. 2022YFB3102500) and the CAS Project for Young Scientists in Basic Research (Grant No. YSBR-035).

Ethical Considerations

Stakeholders. This research was conducted in strict adherence to ethical research principles for cybersecurity and mobile network protocol analysis. All experimental evaluations were performed in a controlled, isolated environment using equipment owned by the research authors. The UE used in experiments was the authors' personal device, while AF and AAnF servers were legally obtained through legitimate cloud service providers. No real user data, SUPIs, or personal information were accessed during the research. All experimental data consisted of synthetic test parameters generated specifically for research purposes, and no interference with operational 5G networks or real user services occurred.

While our research identifies theoretical limitations in existing AKMA protocols, we have avoided publishing specific exploit details that could enable malicious attacks. Our focus remains on constructive protocol improvements rather than exposing actionable vulnerabilities. All research activities were conducted within applicable legal frameworks and industry ethical guidelines. The proposed E2E-AKMA protocol enhancements are designed to improve user privacy protection and authentication security for all mobile network users, contributing to the development of more secure mobile authentication standards.

Broader Societal Impact and Mitigations. Beyond the research context, we address the potential societal implications:

Equity and Accessibility: Regarding concerns about *resource-constrained environments* and the *vulnerability landscape*, E2E-AKMA is designed as an optional enhancement compatible with standard SIM cards (unlike hardware-dependent solutions like FIDO). This ensures that users with legacy devices are not excluded from security upgrades. Deploying E2E-AKMA will increase power consumption and data usage overhead. Since AFs typically support multiple authentication methods, users or operators sensitive to the marginal overhead can opt for standard authentication, preventing any forced burden on underserved populations.

Dual-Use and Regulation: To address *dual-use* concerns where strong privacy might be exploited to evade oversight, our protocol explicitly incorporates a *Lawful Interception (LI)* mechanism. While E2E-AKMA prevents unauthorized tracking by commercial entities or compromised intermediaries, the Core Network retains the capability to decrypt and recover the SUPI. This design ensures that the protocol enhances user privacy without circumventing necessary legal and regulatory frameworks. However, we acknowledge the ongoing debate regarding law enforcement access [21]; while essential for public safety in some frameworks, such mechanisms can introduce security risks or potential for misuse.

Open Science Policy and Artifact Evaluation

We have made our experimental test code publicly available, which includes the test code for 5G-AKMA and E2E-AKMA. Furthermore, we will participate in artifact evaluation. The experimental code is available at https://zenodo.org/records/17896394.

References

[1] 3GPP. Security architecture and procedures for 5g system. Technical Report TS 33.501, 3rd Generation Partnership Project, 2023.

[2] 3GPP Technical Specification. Security architecture and procedures for 5G System. Technical Specification 33.501, 3rd Generation Partnership Project (3GPP), 2020.

[3] 3GPP Technical Specification. Study on authentication and key management for applications based on 3GPP credential in 5G, 2020.

[4] 3GPP Technical Specification. Authentication and key management for applications based on 3GPP credentials in the 5G system, 2024. Version 18.5.0.

[5] Gizem Akman, Philip Ginzboorg, Mohamed Taoufiq Damir, and Valtteri Niemi. Privacy-Enhanced AKMA for Multi-Access Edge Computing Mobility. *Computers*, 12(1):2, 2022.

[6] Mihir Bellare, Zvika Brakerski, Moni Naor, Thomas Ristenpart, Gil Segev, Hovav Shacham, and Scott Yilek. Rerandomizable encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 515–534. Springer, 2012.

[7] Mihir Bellare and Phillip Rogaway. Entity Authentication and Key Distribution. In *Advances in Cryptology—CRYPTO'93: 13th Annual International Cryptology Conference Santa Barbara, California, USA August 22–26, 1993 Proceedings*, pages 232–249. Springer, 2001.

[8] Bellare, Mihir and Rogaway, Phillip. Provably Secure Session Key Distribution The Three Party Case. In *Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, pages 57–66, 1995.

[9] Jake Brutlag. Speed matters for google web search. *Google. June*, 2(9), 2009.

[10] Stuart K Card. *The psychology of human-computer interaction*. Crc Press, 2018.

[11] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. Technical report, IEEE transactions on information theory, 1985.

[12] Philippe Golle, Markus Jakobsson, Ari Juels, and Paul Syverson. Universal re-encryption for mixnets. In *Cryptographers' Track at the RSA Conference*, pages 163–178. Springer, 2004.

[13] Paul A Grassi, James L Fenton, Elaine M Newton, Ray A Perlner, Andrew R Regenscheid, William E Burr, Justin P Richer, Naomi B Lefkovitz, Jamie M Danker, YeeYin Choong, et al. Draft NIST special publication 800-63b digital identity guidelines. *National Institute of Standards and Technology (NIST)*, 27, 2016.

[14] IDStrong. MyFitnessPal Data Breach. https://www.idstrong.com/sentinel/myfitnesspal-data-breach/, 2022.

[15] Tibor Jager, Florian Kohlar, Sven Schäge, and Jörg Schwenk. On the security of TLS-DHE in the standard model. In *Annual Cryptology Conference*, 2012.

[16] Mohsin Khan, Philip Ginzboorg, and Valtteri Niemi. Privacy preserving AKMA in 5G. In *Proceedings of the 5th ACM Workshop on Security Standardisation Research Workshop*, pages 45–56, 2019.

[17] Mohsin Khan, Philip Ginzboorg, and Valtteri Niemi. AKMA: Delegated authentication system of 5G. *IEEE Communications Standards Magazine*, 2021.

[18] Hugo Krawczyk, Kenneth G Paterson, and Hoeteck Wee. On the security of the TLS protocol: A systematic analysis. In *Annual Cryptology Conference*. Springer, 2013.

[19] Yueming Li, Long Chen, Qianwen Gao, and Zhenfeng Zhang. E2E-AKMA: An End-to-End Secure and Privacy-Enhancing AKMA Protocol Against the Anchor Function Compromise. Cryptology ePrint Archive, Paper 2025/XXXX, 2025.

[20] Yueming Li, Long Chen, and Zhenfeng Zhang. A Comprehensive Analysis of the AKMA+ Protocol. In *2025 IEEE 24th International Conference on Trust, Security and Privacy in Computing and Communications*, 2025.

[21] National Academies of Sciences, Engineering, and Medicine. *Decrypting the Encryption Debate: A Framework for Decision Makers*. The National Academies Press, Washington, DC, 2018.

[22] Jakob Nielsen. *Usability engineering*. Morgan Kaufmann, 1994.

[23] OASIS Security Services Technical Committee. Security assertion markup language (SAML) V2.0 technical overview. Working Draft sstc-saml-tech-overview-2.0, OASIS Open, March 2008. Accessed: 2025-12-09.

[24] OpenID Foundation. OpenID connect core 1.0. https://openid.net/specs/openid-connect-core-1_0-final.html, 2014. Accessed: 2025-12-09.

[25] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *International conference on the theory and applications of cryptographic techniques*, pages 223–238. Springer, 1999.

[26] Manoj Prabhakaran and Mike Rosulek. Rerandomizable RCCA encryption. In *Annual International Cryptology Conference*, pages 517–534. Springer, 2007.

[27] RSIM Provisioning. eSIM Whitepaper the what and how of remote SIM provisioning. *GSMA, London, UK, Tech. Rep., Mar*, 2018.

[28] Terena Bell. Adobe's CSO talks security, the 2013 breach, and how he sets priorities. https://www.csoonline.com/article/3268035/adobe-s-cso-talks-security-the-2013-breach-and-how-he-sets-priorities.html, 2021.

[29] W3C. Web Authentication: An API for accessing Public Key Credentials Level 3. https://www.w3.org/TR/webauthn-3/, 2021.

[30] Wikipedia Contributors. 2012 LinkedIn hack. https://en.wikipedia.org/wiki/2012_LinkedIn_hack, 2021.

[31] Tengshun Yang, Shuling Wang, Bohua Zhan, Naijun Zhan, Jinghui Li, Shuangqing Xiang, Zhan Xiang, and Bifei Mao. Formal Analysis of 5G Authentication and Key Management for Applications (AKMA). *Journal of Systems Architecture*, 126:102478, 2022.

[32] Yang Yang, Guomin Yang, Yingjiu Li, Minming Huang, Zilin Shen, Imtiaz Karim, Ralf Sasse, David Basin, Elisa Bertino, Jian Weng, et al. AKMA+: Security and privacy-enhanced and standard-compatible AKMA for 5G communication. 2025.

A Preliminary

In this section, we will present an overview of the cryptographic primitives utilized in our protocol.

A.1 5G Authentication and Key Agreement

The 5G AKA protocol establishes mutual authentication and secure key agreement between UE and the HN [1]. The protocol relies on a symmetric key stored both in the SIM card and the HN, along with the Subscription Permanent Identifier

(SUPI). Upon successful authentication, both parties derive shared session keys for secure 5G communication. For our purposes, we abstract the 5G AKA protocol as enabling the establishment of a unique and private secret key between the UE and HN, which serves as the foundation for subsequent AKMA operations.

A.2 Re-randomizable Public Key Encryption.

Re-randomizable encryption schemes allow anyone to transform a given ciphertext into a new ciphertext that encrypts the same plaintext but appears statistically independent from the original [12, 26]. This property is crucial for maintaining privacy in our protocol design.

Definition 3 (Re-randomizable Encryption)**.** *A public key encryption scheme $\mathcal{E} = (KeyGen, Enc, Dec, ReRand)$ is re-randomizable if:*

- *$ReRand(pk, c, r) \to c'$: Given a public key pk, ciphertext c and randomness r, outputs a re-randomized ciphertext c'.*
- ***Correctness:** For any $(pk, sk) \leftarrow KeyGen(1^\lambda)$, message m, and randomness r, we have $Dec\ (sk,\ ReRand\ (pk, Enc(pk, m), r)) = m$.*
- ***Re-randomization Security:** For any message m and randomness r, the distributions $\{ReRand(pk, Enc(pk, m), r)\}$ and $\{Enc(pk, m)\}$ are computationally indistinguishable.*
- ***Malformed Ciphertext Indistinguishability:** For all PPT adversaries $\mathcal{A}$, the advantage in the following experiment is negligible:*

```
Experiment $Exp^{MCI}_{\mathcal{E},\mathcal{A}}(\lambda)$
1. $(pk, sk) \leftarrow KeyGen(1^\lambda)$
2. $c \leftarrow \mathcal{A}(pk)$
3. $r \xleftarrow{\$} \{0,1\}^*$
4. $c_0 \leftarrow ReRand(pk, c, r)$
5. $c_1 \xleftarrow{\$} \mathcal{C}$    // Sample uniformly from ciphertext space
6. $b \xleftarrow{\$} \{0,1\}$
7. $b' \leftarrow \mathcal{A}(c_b)$
8. return $(b' = b)$
```

The advantage of $\mathcal{A}$ is defined as:

$$Adv^{MCI}_{\mathcal{E},\mathcal{A}}(\lambda) = \left| \Pr[Exp^{MCI}_{\mathcal{E},\mathcal{A}}(\lambda) = 1] - \frac{1}{2} \right|.$$

The ElGamal encryption scheme [11] is a classic example of re-randomizable encryption. Given a ciphertext $(g^r, m \cdot h^r)$ where $h = g^x$ is the public key, one can re-randomize it by computing $(g^r \cdot g^{r'}, m \cdot h^r \cdot h^{r'}) = (g^{r+r'}, m \cdot h^{r+r'})$ for fresh randomness r'. The re-randomized ciphertext is computationally indistinguishable from a fresh encryption of the same message [6].

Importantly, ElGamal encryption satisfies the malformed ciphertext indistinguishability property. For any given ciphertext pair (c_0, c_1) in the group $\mathbb{G} \times \mathbb{G}$ (regardless of whether they are valid encryptions), after re-randomization we obtain $(c'_0, c'_1) = (c_0 \cdot g^{r'}, c_1 \cdot h^{r'})$ for fresh randomness r'. Since $g^{r'}$ and $h^{r'} = g^{x \cdot r'}$ are uniformly distributed group elements (assuming the DDH assumption), the re-randomized pair (c'_0, c'_1) is computationally indistinguishable from two random group elements, regardless of the validity of the original ciphertext (c_0, c_1).

This property enables our protocol to prevent linking attacks: when the AAnF is compromised, adversaries cannot match their own encrypted ciphertext with the re-randomized ciphertext stored in user identifiers, effectively maintaining privacy even under AAnF compromise. Even if AAnF provides malicious or malformed ciphertexts instead of proper SUPI encryptions, the AF's re-randomization process transforms them into values indistinguishable from random, preventing any tracking or linking attempts by the malicious AAnF.

A.3 Commitment and Zero-Knowledge Proofs

An ideal functionality is defined that implicitly captures the functionality and security properties we expect from a real protocol. The real protocol is then said to be secure if it is indistinguishable from the ideal functionality by any efficient distinguisher.

The ideal zero-knowledge functionality $\mathcal{F}_{\mathbf{zk}}$. We use the standard ideal zero-knowledge functionality defined by $((x, w), \lambda) \to (\lambda, (x, R(x, w)))$, where λ denotes the empty string. For a relation R, the functionality is denoted by $\mathcal{F}^R_{\text{zk}}$. Note that any zero-knowledge proof of knowledge fulfills the $\mathcal{F}_{\text{zk}}$ functionality; non-interactive versions can be achieved in the random-oracle model via the Fiat-Shamir paradigm with security under sequential composition. Here is the zero-knowledge functionality $\mathcal{F}^R_{\text{zk}}$ for Relation R:

- Receiving $(\text{prove}, \text{sid}, x, w)$ from party P_i (for $i \in \{1, 2\}$): if $(x, w) \notin R$ or sid has been previously used then ignore the message. Otherwise, send $(\text{proof}, \text{sid}, x)$ to party P_{3-i}.

The committed non-interactive zero-knowledge functionality $\mathcal{F}_{\text{com-zk}}$. In our protocol, parties will send commitments to non-interactive zero-knowledge proofs. We model this via commit-zk functionality, denoted $\mathcal{F}_{\text{com-zk}}$. Given non-interactive zero-knowledge proofs of knowledge, this functionality is securely realized by having the prover commit to such a proof using the ideal commitment functionality $\mathcal{F}_{\text{com}}$. Here is a committed functionality $\mathcal{F}^R_{\text{com-zk}}$ for Relation R, it works with parties P_1 and P_2:

- Upon receiving $(\text{com-prove}, \text{sid}, x, w)$ from a party P_i (for $i \in \{1, 2\}$): if sid has been previously used then ignore. Otherwise, store (sid, i, x, w) and send $(\text{proof-receipt}, \text{sid})$ to P_{3-i}.
- Upon receiving $(\text{decom-proof}, \text{sid})$ from P_i (for $i \in \{1, 2\}$): if (sid, i, x, w) has been stored – if $(x, w) \in R$ then

send $(\text{decom-proof}, \text{sid}, x, 1)$ to P_{3-i}; otherwise send $(\text{decom-proof}, \text{sid}, x, 0)$ to P_{3-i}.

A.4 Paillier encryption.

The Paillier encryption scheme, introduced by Pascal Paillier in 1999 [25], is a probabilistic public key encryption scheme that is notable for its homomorphic properties, allowing specific algebraic operations to be performed on ciphertexts which correspond to operations on the underlying plaintexts. Denote the public/private key pair by (pk, sk), and denote encryption and decryption under these keys by $\mathsf{Enc}_{\text{pk}}(\cdot)$ and $\mathsf{Dec}_{\text{sk}}(\cdot)$, respectively. We denote by $C_1 \oplus C_2$ the "addition" of the plaintexts in C_1, C_2, and by $a \odot C$ the multiplication of the plaintext in C by scalar a.

- If $C_1 = \mathsf{Enc}_{\text{pk}}(m_1)$ and $C_2 = \mathsf{Enc}_{\text{pk}}(m_2)$ are two ciphertexts, then the product $C_1 \oplus C_2$ is a valid encryption of $m_1 + m_2$.
- For a plaintext m and a ciphertext $C = \mathsf{Enc}_{\text{pk}}(m)$, the operation $a \odot C$ results in a ciphertext that encrypts the product $a \cdot m$.

B The 5G AKMA Protocol

We use the Table 4 to give the abbreviations and their meanings that appear in the 3GPP standards.

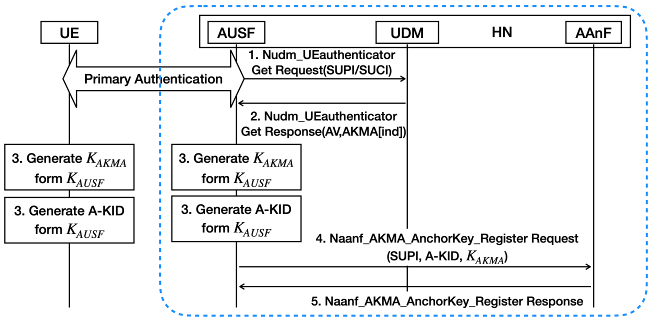
Table 4: A summary of abbreviations

Abbreviations	Meaning
UE	User Equipment
HN	Home Network
AUSF	Authentication Server Function
AAnF	AKMA Anchor Function
NEF	Network Exposure Function
UDM	Unified Data Management
SUPI	Subscription Permanent Identifier
GPSI	Generic Public Subscription Identifier
AF	Application Function

This section provides a detailed description of the 5G AKMA protocol, as outlined in the 3GPP technical specification.

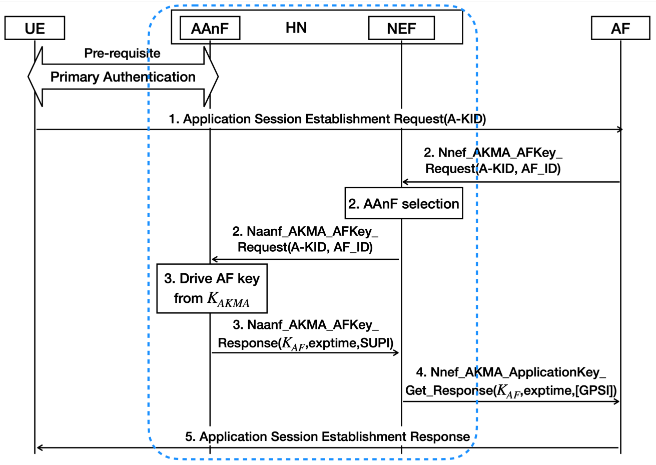
As in the scenario of E2E-AKMA, 5G AKMA protocol also involves three participants: UE, which refers to the cellphone of the user; HN, which represents the mobile service operator, such as AT&T or Verizon; and AF, which denotes any application provider. Each UE is equipped with a SIM card issued by the HN and is identified by its SUPI. HN possesses all the SUPIs issued to UEs and certificates issued by certification authorities, which include public and private key pairs. AF also holds certificates issued by certificate authorities.

Driving AKMA Key. Figure 3 shows the steps for obtaining AKMA materials. AKMA uses 5G primary authentication to generate K_{AUSF} at the HN and UE upon successful authentication.



UE
AUSF
UDM
HN
AAnF
Primary Authentication
1. Nudm_UEauthenticator Get Request(SUPI/SUCI)
2. Nudm_UEauthenticator Get Response(AV,AKMA[ind])
3. Generate K_{AKMA} form K_{AUSF}
3. Generate A-KID form K_{AUSF}
3. Generate K_{AKMA} form K_{AUSF}
3. Generate A-KID form K_{AUSF}
4. Naanf_AKMA_AnchorKey_Register Request (SUPI, A-KID, K_{AKMA})
5. Naanf_AKMA_AnchorKey_Register Response

Figure 3: Deriving AKMA Materials. AUSF, UDM, and AAnF are network elements in HN, and they are grouped together with blue dotted lines to indicate their unity.



UE
AAnF
HN
NEF
AF
Pre-requisite
Primary Authentication
1. Application Session Establishment Request(A-KID)
2. Nnef_AKMA_AFKey_ Request(A-KID, AF_ID)
2. AAnF selection
2. Naanf_AKMA_AFKey_ Request(A-KID, AF_ID)
3. Drive AF key from K_{AKMA}
3. Naanf_AKMA_AFKey_ Response(K_{AF},exptime,SUPI)
4. Nnef_AKMA_ApplicationKey_ Get_Response(K_{AF},exptime,[GPSI])
5. Application Session Establishment Response

Figure 4: Deriving AKMA application key for a specific AF.

1. After the primary authentication procedure, the AUSF engages with the UDM entity to retrieve essential authentication information, including subscription credentials like AKA authentication vectors and the authentication method.
2. The UDM provides the corresponding content to the AUSF, and it inform the AUSF whether AKMA Anchor keys need to be generated for the UE with the AKMA indicator (AKMA [ind]).
3. Upon receiving the AKMA indication from the UDM, the AUSF is required to store the K_{AUSF} and then generate the AKMA Anchor Key K_{AKMA} and the globally unique AKMA Key Identifier (A-KID) from K_{AUSF} using Key Derivation Function (KDF). Before initiating communication with an AKMA AF, the UE needs to generate the same K_{AKMA} and A-KID from K_{AUSF} using the same KDF.
4. Once the AKMA key material is generated, the AUSF

selects the AAnF and sends the generated A-KID and K_{AKMA} to the AAnF along with the SUPI of the UE. The AAnF is responsible for storing the latest information sent by the AUSF.
5. The AAnF acknowledges the receipt of the information sent by the AUSF by sending a response back to the AUSF.

Deriving AKMA application key. The process of obtaining the 5G AKMA application key for a particular AF is shown in Figure 4. In this situation, the AF is located outside the network of HN and is connected to the NEF 5G network element. Both the HN and AF need to authenticate each other, typically through the use of client and server certificates, to establish a mutual TLS connection [2].

1. The UE is responsible for generating the K_{AKMA} and the A-KID from the K_{AUSF} before initiating communication with an AKMA AF. When the UE establishes communication with the AKMA AF, it includes the derived A-KID in the Application Session Establishment Request message. The UE has the option to derive K_{AF} either before or after sending the message.
2. The AF is located outside the HN and connects to the NEF through mutual TLS (mTLS), this requires both HN and AF to send their certificates and signatures to each other for mutual identity authentication. The NEF enables and authorizes the external AF to access the AKMA service and forwards the request to the AAnF. The NEF acts as an intermediary between the AF and AAnF and performs AAnF selection.
3. If the AF-ID received and the AKMA key material are available, the AAnF generates K_{AF}. Additionally, the AAnF determines an expiration time for the key. It then sends the corresponding SUPI, the session key K_{AF}, and its expiration time as a response back to the NEF.
4. The NEF has the option to translate the SUPI to a GPSI and include it in the response. If the AF is within the HN, the AAnF may choose to send the SUPI or GPSI to the AF according to the protocol requirements. If the AF is outside the HN, the AAnF sends the GPSI to the AF.
5. Once the AF receives the session key K_{AF} and expiration time, it responds to the UE. UE and AF using the K_{AF} as the pre-shared key, and then utilize application layer protocols like pre-shared key TLS to establish a secure communication channel.

C AKMA+

C.1 Compare with AKMA+

We compare AKMA+ [32] and E2E-AKMA in the following three aspects.

Security. Under the same security model, E2E-AKMA provides a higher level of security.

- AKMA+ cannot achieve end-to-end security. As described in Appendix **??**, if an adversary compromises AAnF, this is allowed within the security model of AKMA+. Consequently, the adversary can access the AF in the name of any UE, making it impossible to achieve end-to-end security between the UE and the AF.
- In E2E-AKMA, even if an adversary corrupts the AAnF, as long as they don't have access to the user's locally stored private key share (SK_1 in Figure 2), they cannot impersonate the UEs to log into the AF. Furthermore, even if an adversary compromises the AAnF, they cannot obtain relevant information about the user's account through the user's login behavior.

Privacy. The privacy definitions of AKMA+ and E2E-AKMA are based on fundamentally different approaches to protecting user privacy and are therefore not directly comparable. Both protocols aim to break the linkage between users' real identities (SUPI) and their account information or login behaviors, but they achieve this goal through distinct strategies. AKMA+ protects privacy by concealing the user's real identity (SUPI) from potential adversaries, ensuring that even if account activities are observed, they cannot be linked to specific individuals. In contrast, E2E-AKMA protects privacy by concealing account information and login behaviors from adversaries, ensuring that even if the user's identity is known, their specific activities remain private. Both approaches effectively safeguard user privacy by severing different aspects of the identity-activity correlation, and it is difficult to definitively determine which approach is superior in practical applications, as their effectiveness depends on the specific threat model and deployment context.

Disposition. In practical deployment, the maintenance cost of the AKMA+ solution is higher than that of the E2E-AKMA solution.

- In AKMA+, AAnF is required to securely maintain a dynamic database storing a large number of $\{[\text{A-KID}_i, K_{\text{AKMA},i}\}$ pairs, which incurs higher costs for AAnF in practical use.
- In E2E-AKMA, AAnF only needs to securely store two keys (K_{AAnF} and SK_{AAnF} as shown in Figure 2). Compared to maintaining a dynamic database, these two keys can be securely stored in hardware devices for computation during practical deployment, resulting in relatively lower costs.

D Detailed Security Models

Our new model for E2E-AKMA is inspired by the well-established authenticated key exchange frameworks that have been rigorously examined by the cryptography community over the past decades [7, 8]. Specifically, we define the authentication and key-indistinguishable property to guarantee that an adversary cannot access the communication content between the UE and AF, even if the adversary possesses the same perspective as AAnF. Additionally, we consider AAnF-unlinkability and AF-unlinkability to ensure that the usage habits of users are not collected by AAnF and AF.

D.1 Instances

In the security model, there exists a challenger and an adversary, where the challenger simulates the execution of the protocol for the adversary. The adversary can interact with the protocol execution simulated by the challenger, and the interactions that the adversary can perform with the challenger correspond to the attacks that the adversary can carry out in reality. To accurately describe how the challenger simulates the protocol execution for the adversary, we regard each UE, AAnF, and AF individual as a "party", for example, the mathematical symbol UE represents an independent UE individual, and similarly, $AAnF$ and AF represent a specific AAnF or AF party. Corresponding to the presence of multiple UEs, AAnFs, and AFs in practice, we use cursive letters to represent the three sets of parties, for example, $\mathcal{UE}$ represents a collection of UEs, and $\mathcal{AANF}$ and $\mathcal{AF}$ represent a set of $AAnF$s and AFs.

A party is composed of many instances. For example, in the case of a UE party UE, it may register multiple accounts with different AFs through the $AAnF$ that issues its SIM card. Among these, the registration or login processes corresponding to different accounts are represented by various instances. The same applies to an AAnF or AF party. We regard a Turing machine as an instance, denoted by the symbol π. Given specific input, it produces specific output. We use $\pi_{Role}^{i,j}$ to represent an instance, where the subscript $Role \in \mathcal{UE} \cup \mathcal{AANF} \cup \mathcal{AF}$ indicates the party to which the instance belongs.

For an instance $\pi_{UE}^{i,j}$ of a UE party, $i > 0, j = 0$ indicates the registration of the i-th account, while $j > 0$ represents the j-th login of that account. For an instance $\pi_{AAnF}^{i',j'}$ of an AAnF party, $i' > 0, j' = 0$ indicates the i'-th registration performed with a certain UE, while $j' > 0$ represents the j'-th login corresponding to that registration. For an instance $\pi_{AF}^{i'',j''}$ of an AF party, $i'' > 0, j'' = 0$ indicates the i''-th registration performed with a certain UE, while $j'' > 0$ represents the j''-th login corresponding to that registration.

Since communication between instances utilizes the 5G AKA channel and the TLS channel, the adversary cannot obtain the transcript between the participants by eavesdropping on the channel. However, if the adversary corrupts a certain party, the adversary can act as that party to communicate with other parties. In this case, the adversary can observe the transcript within the channel of the corrupted party. As shown in Figure 5, the communication content between honest parties cannot be accessed by the adversary. The adversary can only observe the communication content between the parties it corrupts and the honest parties. For simplicity and better understanding, we did not explicitly illustrate the instances within the parties in the picture. Actually, communication occurs between the instances within the parties.

The use of the 5G AKA channel and the TLS channel results in the matching relationships between instances be-

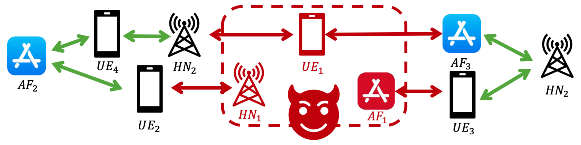


Figure 5: A diagram of communication channels. Red parties are corrupted by the adversary, red channels are visible to the adversary, green channels are not.

ing determined during the execution of the protocol. For the definition of partner relationships, an instance needs to record its communication partner in the variable $Role_{par}$ ($Role \in \mathcal{UE} \cup \mathcal{AANF} \cup \mathcal{AF}$).

The 5G AKA lacks forward secrecy, so it is necessary to consider the capability of the adversary to subsequently compromise UE/AAnF and utilize the symmetric key K_{SIM} stored in SIM cards to decrypt prior sessions. Therefore, in the model, the transcript under the 5G AKA channel is recorded as a variable "trans" of an instance. When the adversary corrupts the UE or the $AAnF$, these prior transcripts need to be provided to the adversary.

Each instance possesses a set of internal states, denoted as $\pi_{Role}^{i,j}.\mathsf{st}(Role \in \mathcal{UE} \cup \mathcal{AANF} \cup \mathcal{AF})$. These internal states play a pivotal role in the functioning of the instance. When a message is received, the instance processes it by integrating the message with its existing parameters to produce an output. Furthermore, the instance updates its parameters to reflect the changes induced by this interaction. The specific components of the internal state include the following categories:

- $\pi_{Role}^{i,j}.\mathsf{exe}(Role \in \mathcal{UE} \cup \mathcal{AANF} \cup \mathcal{AF}) \in \{accepted/rejected, running, \perp, send\_to\_AF, send\_to\_AAnF\}$: the execution state of each instance is initially $\perp$. When executing, it is *running*. After execution, it is *accepted* if the protocol succeeds or *rejected* if it fails. For UE instances, additional states *send_to_AF* and *send_to_AAnF* indicate that the UE needs to interact with AF or AAnF respectively.
- $\pi_{UE}^{i,j}$.pending_msg: stores the message that UE needs to send to the third party (AF or AAnF) when $\pi_{UE}^{i,j}.\mathsf{exe} \in \{\mathsf{send\_to\_AF}, \mathsf{send\_to\_AAnF}\}$.
- $\pi_{Role}^{i,j}.AKA(Role \in \mathcal{UE} \cup \mathcal{AANF})$: the transcript in the AKA channel.
- $\pi_{Role}^{i,j}.trans(Role \in \mathcal{UE} \cup \mathcal{AF})$: the transcript between the UE and the AF, in the TLS channel.
- $\pi_{UE}^{i,j}.K_{UE}$: the K_{UE} of this UE instance. For the same i, that is, the same account, the K_{UE} remains identical.
- $\pi_{UE}^{i,j}.AF_{par}$: this value is initially empty $\emptyset$ and is set to the partner the instance believes it is communicating with after protocol execution, e.g., this variable is the AF partner of a UE instance that refers to a specific AF AF.
- $\pi_{AF}^{i,j}.UE_{par}$: the UE partner of the AF instance.

- $\pi^{i,j}_{AAnF}.SK_{AAnF}$: the private key of this $AAnF$ instance. This variable is the same for instances of the same AAnF party.
- $\pi^{i,j}_{AAnF}.SUPI$: the $SUPI$ of this $AAnF$ instance. This variable is the same for instances of the same AAnF party.
- $\pi^{i,j}_{AF}.PK_{AAnF}$: the public key of all the AAnFs. This variable is the same for all the AF parties.
- $\pi^{i,j}_{AF}.DB_{AF}$: the database of this AF.
- $\pi^{i,j}_{Role}.ssk(Role \in \mathcal{UE} \cup \mathcal{AF}, j > 0)$: session key of this instance.

D.2 Oracles

The oracles defined in Figure 6 and Figure **??** serve as the interface between the adversary and the protocol execution environment, modeling various adversarial capabilities and attack scenarios. Each oracle is carefully designed to capture specific types of threats and interactions that may occur in real-world deployments. At the beginning of the game, the challenger $\mathcal{C}$ establishes the set of all parties involved in the protocol, $\mathcal{UE}$, $\mathcal{AANF}$, and $\mathcal{AF}$.

Start **Oracle.** The Start oracle allows an adversary to initiate protocol sessions by selecting the i-th UE party and an AF. Since the SIM card of UE is issued by a specific AAnF, the relationship between the UE and AAnF is pre-established before the AKMA protocol.

The oracle first verifies the execution status of the UE instance $\pi^{i,0}_{UE}$. If the status is marked as "rejected," the oracle returns $\perp$, indicating that the UE has not been registered. If the status is $\perp$ (signifying the UE is unregistered), the oracle initiates the registration phase. This involves identifying the corresponding AAnF, locating available instances on both AAnF and AF, and executing the Reg protocol. Conversely, if the status is "accepted" (indicating the UE is already registered), the oracle proceeds to the login phase by executing the Login protocol. In both scenarios, the oracle updates the relevant instance variables, including cryptographic keys, partner information, AKA transcripts, and execution states. This oracle serves as a model for an adversary's ability to trigger protocol executions and monitor subsequent state transitions, thereby capturing potential vulnerabilities in the system.

CorruptUE **Oracle.** This oracle models a complete corruption of a UE, representing a scenario where the physical device has been captured. Under such circumstances, the adversary is assumed to be capable of breaching the security of the SIM card. When invoked with a specific UE, the oracle reveals all AKA protocol transcripts ($\mathcal{L}_{AKA}$) and secret keys ($\mathcal{L}_{K_{UE}}$) associated with that UE across all protocol instances. The oracle systematically iterates through all instances, denoted as $\pi^{i,j}_{UE}$, where $i > 0$ and $j \geq 0$, to collect the AKA transcripts and keys. Notably, it retrieves these artifacts from registration instances (where $j = 0$). Once a UE is compromised, it is added to the list $\mathcal{L}_{crp}$ to track entities under the adversary's control. This oracle provides the adversary with comprehensive historical information about the UE's interactions within the protocol, effectively exposing all past communications and cryptographic material associated with the compromised device.

Compromise **Oracle.** The Compromise oracle models a constrained adversarial scenario in which the attacker gains access solely to the secret keys K_{UE} stored in the UE's regular memory, typically through software-level intrusions such as malware infections. Unlike the CorruptUE oracle, this attack does not expose the historical AKA transcripts, as it is protected by session keys derived within the SIM card. The key distinction lies in the storage locations: while the SIM card employs strong hardware-based protections to safeguard authentication secrets, the K_{UE} keys stored in the UE's memory are more vulnerable to software exploits. Operationally, the oracle collects all K_{UE} values from registration instances and records the compromised UE in the list $\mathcal{L}_{cpm}$. Importantly, despite obtaining K_{UE}, the adversary cannot impersonate the compromised UE to the AAnF because the SIM card's secure hardware remains uncompromised, preventing full control over authentication processes. This separation between full corruption (hardware compromise) and partial compromise (software-level access) allows for a more nuanced and realistic security analysis.

Reveal **Oracle.** The Reveal oracle models session key leakage attacks, allowing the adversary to obtain the session key ssk of any accepted protocol session. The oracle first validates that the role is either a UE or AF, and that the specified instance has reached the $accept$ state. If valid, the oracle adds the instance to the revealed list $\mathcal{L}_{rvl}$ and returns the session key. This oracle captures realistic threats such as memory dumps, side-channel attacks, or implementation vulnerabilities that expose session keys.

CorruptAAnF **Oracle.** Similar to UE corruption, this oracle models the complete compromise of an AAnF entity. It returns all AKA transcripts, SUPI values of managed UEs, and the long-term key K_{AAnF}. The oracle iterates through all AAnF instances, collecting transcripts and SUPI information, then adds the AAnF to the corruption list $\mathcal{L}_{crp}$. This models insider threats, server compromises, or sophisticated attacks against the authentication infrastructure.

CorruptAF **and** Breach **Oracles.** The CorruptAF oracle models full adversarial control over an AF, returning its local database DB_{AF} and adding the AF to the corruption list. In contrast, the Breach oracle models a weaker attack where only the database is leaked without full control of the AF. This distinction captures different threat levels: corruption implies the adversary can impersonate the AF, while breach only exposes stored data. Both oracles model realistic threats to application servers.

UE, AAnF, and AF Oracles. These oracles facilitate active adversarial interactions by enabling an adversary to send arbitrary messages to protocol instances on behalf of compromised parties. For the UE and AAnF oracles, the adversary is required to specify a role from the corruption list, denoted as $\mathcal{L}_{crp}$, thereby ensuring that only compromised entities can be impersonated. In contrast, the AF oracle does not impose this restriction, reflecting the assumption that TLS provides only unidirectional authentication.

Each oracle processes the received message by updating the internal state and execution status of the targeted protocol instance before returning an appropriate response. The UE oracle has additional complexity as it handles potential multi-party interactions: when a UE instance needs to communicate with a third party (AF or AAnF) during protocol execution, the oracle automatically facilitates this interaction if the third party is not compromised, ensuring seamless protocol flow even in the presence of honest intermediaries.

These oracles are designed to model various attack scenarios, including man-in-the-middle attacks, message injection, impersonation attempts, and complex multi-party protocol interactions where honest parties may need to communicate through intermediaries.

Global Lists. During the process, the challenger maintains the following global lists.

- $\mathcal{L}_{crp}$: The set of corrupted entities (UEs, AAnFs, or AFs) whose internal states or secrets have been exposed to the adversary.
- $\mathcal{L}_{rvl}$: The set of oracle instances (protocol sessions) for which the session keys have been revealed to the adversary via the Reveal oracle.
- $\mathcal{L}_{cpm}$: The set of UEs that have been compromised through the Compromise oracle, meaning all their secret keys have been exposed.

D.3 Partner

The partner relationship between UE and AF is established through their interaction, represented by the transcript parameter, denoted as *trans*, which is maintained by the respective instance.

A UE instance $\pi_{UE}^{i,j}$ and an AF instance $\pi_{AF}^{i'',j''}$ are considered to be in a partner relationship if the following conditions are satisfied:

1. Both instances have successfully completed the protocol, meaning their execution states, exe, is *accepted*. Formally, this is expressed as $\pi_{UE}^{i,j}.\mathsf{exe} = \pi_{AF}^{i'',j''}.\mathsf{exe} = accepted$.
2. The transcripts of the two instances are identical, i.e., $\pi_{UE}^{i,j}.trans = \pi_{AF}^{i'',j''}.trans$.
3. The corresponding registration instances for each entity have an execution state of *accepted*, and their transcripts are also identical. Specifically, $\pi_{UE}^{i,0}.\mathsf{exe} = \pi_{AF}^{i'',0}.\mathsf{exe} = accepted$ and $\pi_{UE}^{i,0}.trans = \pi_{AF}^{i'',0}.trans$.

D.4 Authentication

The authentication experiment is designed to capture the process of mutual authentication between the UE and the AF.

Our protocol is founded upon the *Trust on First Use* (TOFU) authentication methodology. Within this framework, adversaries are permitted to independently register accounts on the AF; however, such actions do not qualify as valid attacks. For an adversary to successfully execute an attack, they must compromise the authentication of accounts that have been registered by an honest UE. Consequently, the criterion for a valid attack requires that the challenged AF instance having a non-empty intentional partner. Conversely, if the intentional partner of the AF is empty, it indicates that the account was initially registered by adversaries. Such cases are explicitly excluded from our definition of valid adversarial attacks.

Furthermore, communication between the UE and the AF occurs over a TLS channel, with the UE verifying the certificate of the AF. This mechanism ensures that attempts to register a UE with a malicious AF are outside the scope of our model.

EXPERIMENT $\mathsf{Expt\text{-}Auth}_{AKMA}^{\mathcal{A}}(1^n)$

For an $\mathsf{AKMA} = (\mathsf{Init}, \mathsf{Reg}, \mathsf{Login})$, the following security experiment is run between the challenger and adversary $\mathcal{A}$.

- **Setup.** For each $Role \in \mathcal{UE} \cup \mathcal{AANF} \cup \mathcal{AF}$, the challenger executes the Init algorithm and provides the output PP_{AKMA} to $\mathcal{A}$.
- **Query.** $\mathcal{A}$ interact with oracles defined in Figure 6 and Figure **??**.
- **Output.** Finally, $\mathcal{A}$ terminates with no output, and the experiment outputs 1 if and only if there exists an honest UE instance $\pi_{UE}^{i,j}$ and an honest AF instance $\pi_{AF}^{i'',j''}$ such that $\mathcal{A}$ has neither compromised the UE nor corrupted the $AAnF$ to which the UE belongs simultaneously, i.e., $(UE \in \mathcal{L}_{crp}) \cup (AF \in \mathcal{L}_{crp}) \cup (UE \in \mathcal{L}_{cpm} \cap AAnF \in \mathcal{L}_{crp})$=0, and at least one of the specified conditions is satisfied:
    1. $\pi_{UE}^{i,j}$ and $\pi_{AF}^{i'',j''}$ form a partner relationship and are intentional partners of each other, $\pi_{UE}^{i,j}.AF_{par} = AF$, $\pi_{AF}^{i'',j''}.UE_{par} = UE$. And without adversary involvement, $UE \notin \mathcal{L}_{cpm}$, $AAnF \notin \mathcal{L}_{crp}$. But the execution state of at least one of them is not accepted, $\pi_{UE}^{i,j}.\mathsf{exe} \neq accepted$ or $\pi_{AF}^{i'',j''}.\mathsf{exe} \neq accepted$.
    2. $\pi_{UE}^{i,j}$ accepted with no partner, or accepted with more than one partner, and its intentional partner was not cor-

Oralce Start$((UE,i),AF)$

```
    / Check the registration status of the $UE$.
1 : if $\pi_{UE}^{i,0}$.exe $= rejected$ :
2 :     return $\bot$
    / Determine the AAnF associated with the UE,
        / as well as the specific instance of the AAnF and AF that aim to be registered.
        / Execute the registration process and subsequently update the associated instance variables to reflect the changes.
3 : elseif $\pi_{UE}^{i,0}$.exe $= \bot$ :
4 :     $AAnF \leftarrow Identify(UE)$, find $\pi_{AAnF}^{i',0}$.exe $= \pi_{AF}^{i'',0}$.exe $= \bot$
5 :     $(K_{UE}, \bot, DB'_{AF}) \leftarrow$ Reg $\left(\pi_{UE}^{i,0}(\emptyset), \pi_{AAnF}^{i',0}(SUPI, SK_{AAnF}, K_{AAnF}, PK_{core}), \pi_{AF}^{i'',0}(\mathcal{PK}_{AAnF}, DB_{AF})\right)$
6 :     $\pi_{UE}^{i,0}.K_{UE} := K_{UE}, \pi_{UE}^{i,0}.AF_{par} := AF, \pi_{AF}^{i'',0}.UE_{par} := UE,$
          Update $\pi_{UE}^{i,0}.AKA, \pi_{UE}^{i,0}.trans, \pi_{AAnF}^{i',0}.AKA, \pi_{AF}^{i'',0}.trans, \pi_{AF}^{i'',0}.DB_{AF}$
7 :     return $\bot$
    / Determine the AAnF associated with the UE,
        / as well as the specific instance of the UE, AAnF, and AF that are intended to initiate the login process.
        / Execute the login process and subsequently update the instance variables accordingly.
8 : elseif $\pi_{UE}^{i,0}$.exe $= accepted$ :
9 :     $AAnF \leftarrow Identify(UE)$, find $\pi_{UE}^{i,j}$.exe $= \pi_{AAnF}^{i',j'}$.exe $= \pi_{AF}^{i'',j''}$.exe $= \bot$
10 :    $(ssk, \bot, ssk) \leftarrow$ Login $\left(\pi_{UE}^{i,j}(K_{UE}), \pi_{AAnF}^{i',j'}(SUPI, K_{AAnF}), \pi_{AF}^{i'',j''}(DB_{AF})\right)$
11 :    $\pi_{UE}^{i,j}.ssk := \pi_{AF}^{i'',j''}.ssk := ssk, \pi_{UE}^{i,j}.AF_{par} := AF, \pi_{AF}^{i'',j''}.UE_{par} := UE,$
          Update $\pi_{UE}^{i,j}.AKA, \pi_{UE}^{i,j}.trans, \pi_{AAnF}^{i',j'}.AKA, \pi_{AF}^{i'',j''}.trans$
12 :    return $\bot$
```

Oracle CorruptUE(UE)

```
    / Return transcripts and $K_{UE}$s of the $UE$
1 : $\mathcal{L}_{AKA} \leftarrow \emptyset, \mathcal{L}_{K_{UE}} \leftarrow \emptyset$
2 : $AAnF \leftarrow Identify(UE)$
3 : for $i > 0, j \geq 0$ do :
4 :     $\mathcal{L}_{AKA} \leftarrow \mathcal{L}_{AKA} \cup \pi_{UE}^{i,j}.AKA$
5 :     $\mathcal{L}_{K_{UE}} \leftarrow \mathcal{L}_{K_{UE}} \cup \pi_{UE}^{i,0}.K_{UE}$
6 : $\mathcal{L}_{crp} \leftarrow \mathcal{L}_{crp} \cup UE$
7 : return $(\mathcal{L}_{AKA}, \mathcal{L}_{K_{UE}})$
```

Oracle CorruptAAnF$(AAnF)$

```
    / Return transcripts, $K_{AAnF}$ and $SUPI$s of the $AAnF$.
1 : $\mathcal{L}_{AKA} \leftarrow \emptyset, \mathcal{L}_{SUPI} \leftarrow \emptyset$
2 : for $i > 0, j \geq 0$ do :
3 :     $\mathcal{L}_{AKA} \leftarrow \mathcal{L}_{AKA} \cup \pi_{AAnF}^{i,j}.AKA$
4 :     $\mathcal{L}_{SUPI} \leftarrow \mathcal{L}_{SUPI} \cup \pi_{AAnF}^{i,j}.SUPI$
5 : $\mathcal{L}_{crp} \leftarrow \mathcal{L}_{crp} \cup AAnF$
6 : return $(\mathcal{L}_{AKA}, \mathcal{L}_{SUPI}, K_{AAnF})$
```

Oracle Breach(AF)

```
    / Return local storage of $AF$.
1 : return $DB_{AF}$
```

Oracle UE$((i,j), \pi_{Role}^{i',j'}, m)$

```
    / Send $m$ to the $UE$ on behalf of Role.
    / $m = \emptyset$ means $UE$ sends the first msg.
1 : if $Role \notin \mathcal{L}_{crp}$ :
2 :     return $\bot$
3 : $(\mathsf{st}', \mathsf{exe}', m') \leftarrow \pi_{UE}^{i,j}(Role, m)$
4 : / Handle potential UE-to-AF interaction when Role is AAnF
5 : if $Role = AAnF \wedge \pi_{UE}^{i,j}$.exe $=$ "send_to_AF" $\wedge AF \notin \mathcal{L}_{crp}$ :
6 :     $m_{AF} \leftarrow \pi_{UE}^{i,j}$.pending_msg
7 :     $(\mathsf{st}_{AF}, \mathsf{exe}_{AF}, m'_{AF}) \leftarrow \pi_{AF}^{i'',j''}(UE, m_{AF})$
8 :     $(\mathsf{st}'', \mathsf{exe}'', m'') \leftarrow \pi_{UE}^{i,j}(AF, m'_{AF})$
9 :     return $m''$
10 : / Handle potential UE-to-AAnF interaction when Role is AF
11 : if $Role = AF \wedge \pi_{UE}^{i,j}$.exe $=$ "send_to_AAnF" $\wedge AAnF \notin \mathcal{L}_{crp}$ :
12 :    $m_{AAnF} \leftarrow \pi_{UE}^{i,j}$.pending_msg
13 :    $(\mathsf{st}_{AAnF}, \mathsf{exe}_{AAnF}, m'_{AAnF}) \leftarrow \pi_{AAnF}^{i',j'}(UE, m_{AAnF})$
14 :    $(\mathsf{st}'', \mathsf{exe}'', m'') \leftarrow \pi_{UE}^{i,j}(AAnF, m'_{AAnF})$
15 :    return $m''$
16 : return $m'$
```

Oracle Reveal$(Role, i, j)$

```
    / Return the session key.
1 : if $Role \notin \mathcal{UE} \cup \mathcal{AF}$ :
2 :     return $\bot$
3 : elseif $\pi_{Role}^{i,j}$.st $= \bot / reject$
4 :     return $\bot$
5 : elseif $\pi_{Role}^{i,j}$.st $= accept$ :
6 :     $\mathcal{L}_{rvl} \leftarrow \mathcal{L}_{rvl} \cup \pi_{Role}^{i,j}$
7 :     return $\pi_{Role}^{i,j}.ssk$
```

Oracle Compromise(UE)

```
    / Return all $K_{UE}$s owned by the UE.
1 : $\mathcal{L}_{K_{UE}} \leftarrow \emptyset$
2 : while $i > 0$ :
3 :     $\mathcal{L}_{K_{UE}} \leftarrow \mathcal{L}_{K_{UE}} \cup \pi_{UE}^{i,0}.K_{UE}$
4 : $\mathcal{L}_{cpm} \leftarrow \mathcal{L}_{cpm} \cup UE$
5 : return $\mathcal{L}_{K_{UE}}$
```

Oracle CorruptAF(AF)

```
    / Return local storage of $AF$.
1 : $\mathcal{L}_{crp} \leftarrow \mathcal{L}_{crp} \cup AF$
2 : return $DB_{AF}$
```

Oracle AAnF$((i,j), \pi_{Role}^{i',j'}, m)$

```
    / Send $m$ to the $AAnF$ on behalf of Role.
1 : if $Role \notin \mathcal{L}_{crp}$ :
2 :     return $\bot$
3 : $(\mathsf{st}', \mathsf{exe}', m') \leftarrow \pi_{AAnF}^{i',j'}(Role, m)$
4 : return $m'$
```

Oracle AF$((i,j), \pi_{Role}^{i',j'}, m)$

```
    / Send $m$ to the $AF$ on behalf of Role.
1 : $(\mathsf{st}', \mathsf{exe}', m') \leftarrow \pi_{AF}^{i'',j''}(Role, m)$
2 : return $m'$
```

Figure 6: Oracles.

rupted.

3. The intentional partner of $\pi_{AF}^{i'',j''}$ is not $\emptyset$ and $\pi_{AF}^{i'',j''}$ accepted with no partner, or accepted with more than one partner, and its intentional partner was not corrupted.

Definition 4 (Authentication)**.** *An AKMA scheme is Authentication, if for any* PPT *adversary $\mathcal{A}$, there exists a negligible function* negl *such that* $\Pr[\mathsf{Expt\text{-}Auth}^{\mathcal{A}}_{\mathsf{AKMA}}(1^{\mathsf{n}}) = 1] \leq \mathsf{negl}(n)$.

D.5 Key-Indistinguishability

This experiment characterizes that the adversary cannot distinguish a real session key from a random key.

EXPERIMENT $\mathsf{Expt\text{-}KeyIND}^{\mathcal{A}}_{AKMA}(1^n)$

For an $\mathsf{AKMA} = (\mathsf{Init}, \mathsf{Reg}, \mathsf{Login})$, the following security experiment is run between the challenger and the adversary $\mathcal{A}$.

- **Setup.** For each *Role* $\in \mathcal{UE} \cup \mathcal{AANF} \cup \mathcal{AF}$, the challenger executes the Init algorithm and provides the output PP_{AKMA} to $\mathcal{A}$.
- **Query.** $\mathcal{A}$ interact with oracles defined in Figure 6 and Figure **??**. The challenger randomly selects a bit b from 0 and 1, $b \leftarrow\$ \{0,1\}$.
- **Test.** $\mathcal{A}$ selects a UE instance $\pi_{UE}^{i,j}$ or an AF instance $\pi_{AF}^{i'',j''}$ with the exe variable set to *accepted* as the challenge session. If $b=0$, the challenger randomly samples $k \leftarrow\$ \{0,1\}^{\lambda}$ for the adversary; if $b=1$, the challenger provides the adversary with the real session key *ssk* of the chosen instance. If the adversary selects a UE instance $\pi_{UE}^{i,j}$, all the following conditions must be satisfied:
  1. $\pi_{UE}^{i,j}$ has neither been revealed $\pi_{UE}^{i,j} \notin \mathcal{L}_{rvl}$, and has not been queried for corrupt before test $UE \notin \mathcal{L}_{crp}$.
  2. If $\pi_{UE}^{i,j}$ has partner $\pi_{AF}^{k,h}$, then $\pi_{AF}^{k,h}$ has neither been revealed session key $\pi_{AF}^{k,h} \notin \mathcal{L}_{rvl}$, and has not been queried for corrupt before test $AF \notin \mathcal{L}_{crp}$.
  3. $\mathcal{A}$ has not both compromise UE and corrupt the $AAnF$ to which the UE belongs simultaneously, $UE \notin \mathcal{L}_{cpm} \cap AAnF \notin \mathcal{L}_{crp}$=1.

  If the adversary selects an AF instance $\pi_{AF}^{i'',j''}$, it must satisfy all conditions. As in the Authentication experiment, we use a non-empty intentional partner requirement to exclude cases where the adversary creates a UE to interact with the AF.
  1. $\pi_{AF}^{i'',j''}$ has neither been revealed $\pi_{AF}^{i'',j''} \notin \mathcal{L}_{rvl}$, and has not been queried for corrupt before test $AF \notin \mathcal{L}_{crp}$.
  2. The intentional partner of $\pi_{AF}^{i'',j''}$ is not $\emptyset$, if it has partner $\pi_{UE}^{k'',h''}$, $\mathcal{A}$ has not been revealed session key $\pi_{UE}^{k'',h''} \notin \mathcal{L}_{rvl}$, and has not been queried for corrupt before test $UE \notin \mathcal{L}_{crp}$.
  3. The intentional partner of $\pi_{AF}^{i'',j''}$ is not $\emptyset$, if it has partner $\pi_{UE}^{k'',h''}$, $\mathcal{A}$ has not both compromise UE and corrupt the $AAnF$ to which the UE belongs simultaneously, $UE \notin \mathcal{L}_{cpm} \cap AAnF \notin \mathcal{L}_{crp}$=1.
- **Query.** $\mathcal{A}$ can continue querying the oracles in Figure 6 and Figure **??**, but the chosen instance must satisfy the aforementioned conditions.
- **Output.** Finally, $\mathcal{A}$ terminates with output b', and the experiment outputs 1 if $b' = b$.

Definition 5 (Key-Indistinguishability)**.** *An AKMA protocol is Key-Ind, if for any* PPT *adversary $\mathcal{A}$, there exists a negligible function* negl *such that* $\Pr[\mathsf{Expt\text{-}KeyIND}^{\mathcal{A}}_{\mathsf{AKMA}}(1^{\mathsf{n}}) = 1] \leq \frac{1}{2} + \mathsf{negl}(n)$.

D.6 AF Unlinkability

The concept of AF Unlinkability, informally, ensures that when a UE registers with multiple AFs, even if these AFs are malicious and collude, they cannot link or identify accounts belonging to the same UE. In our experimental setup, we require that the two honest UEs, denoted UE_0 and UE_1, correspond to the *same* AAnF.

We assume the adversary has fully compromised multiple AFs and can observe the registration and login behaviors of UE_0 and UE_1 with these corrupted AFs. Subsequently, one of the UEs is randomly chosen to register and log in to a target corrupted AF. The adversary's goal is to distinguish which UE was selected.

EXPERIMENT $\mathsf{Expt\text{-}AF\text{-}UNL}^{\mathcal{A}}_{AKMA}(1^n)$

This security experiment is conducted between a challenger and the adversary $\mathcal{A}$ for an $\mathsf{AKMA} = (\mathsf{Init}, \mathsf{Reg}, \mathsf{Login})$. The procedure is as follows:

- **Setup.** For each *Role* $\in \mathcal{UE} \cup \mathcal{AANF} \cup \mathcal{AF}$, the challenger executes the Init algorithm and provides the resulting public parameters PP_{AKMA} to the adversary $\mathcal{A}$.
- **Query.** The adversary $\mathcal{A}$ interacts with oracles defined in Figure 6 and Figure **??**. During this phase, $\mathcal{A}$ is allowed to select multiple AFs, denoted as AF_1 to AF_n, and query them using the $\mathsf{CorruptAF}$ oracle. Among these, one AF is chosen as the challenge target, denoted by AF^*.
- **Test.** The adversary $\mathcal{A}$ selects two honest UEs, denoted as UE_0 and UE_1, both corresponding to the *same* AAnF, denoted $AAnF^*$. These UEs must not have been queried by $\mathcal{A}$ using the $\mathsf{CorruptAAnF}$ oracle. The adversary can utilize the UE oracle to send messages to these UEs for registration and login with the previously corrupted AFs, namely AF_1 to AF_n.

  Subsequently, $\mathcal{A}$ selects two unregistered instances, denoted as $\pi_{UE_0}^{i_0,0}$ and $\pi_{UE_1}^{i_1,0}$, from these UEs, where their execution status is set to $\perp$. As part of the challenge session, the challenger selects a random bit $b \leftarrow \{0,1\}$ and proceeds as follows:
  - The instance $\pi_{UE_b}^{i_b,0}$ registers with the challenge target AF^*, facilitated by the corresponding $AAnF^*$.

- **Query.** The adversary $\mathcal{A}$ may continue interacting with the oracles defined in Figure 6 and Figure **??**. Additionally, $\mathcal{A}$ may request the selected UE instance from the Test phase, denoted as UE_b, to perform a login operation. This interaction is subject to the following three conditions:
  - Neither of the UEs (UE_0, UE_1) belong to the sets $\mathcal{L}_{crp}$ or $\mathcal{L}_{cpm}$:

$$(UE_0 \in \mathcal{L}_{crp} \cup UE_0 \in \mathcal{L}_{cpm} \cup UE_1 \in \mathcal{L}_{crp} \cup UE_1 \in \mathcal{L}_{cpm}) = 0.$$

  - The AAnF $AAnF^*$ does not belong to the corruption set $\mathcal{L}_{crp}$:

$$AAnF^* \notin \mathcal{L}_{crp}.$$

  - The session keys of both UE instances ($\pi_{UE_0}^{i_0,j_0}$, $\pi_{UE_1}^{i_1,j_1}$) have not been queried via the $\mathcal{L}_{rvl}$ set:

$$(\pi_{UE_0}^{i_0,j_0} \in \mathcal{L}_{rvl} \cup \pi_{UE_1}^{i_1,j_1} \in \mathcal{L}_{rvl}) = 0, \quad (j_0 > 0, j_1 > 0).$$

- **Output.** Finally, the adversary $\mathcal{A}$ outputs a guess bit b'. The experiment outputs 1 if and only if $b' = b$.

Definition 6 (AF-Unlinkability)**.** *An AKMA scheme is AF-Unlinkable if for any* PPT *adversary $\mathcal{A}$, there exists a negligible function* negl *such that*

$$\Pr\left[\text{Expt-AF-UNL}_{AKMA}^{\mathcal{A}}(1^n) = 1\right] \leq \frac{1}{2} + \text{negl}(n),$$

where the two honest UEs UE_0 and UE_1 correspond to the same *AAnF.*

D.7 AAnF-Unlinkability

Informally, the objective is to enable the same UE to register multiple accounts through its associated AAnF, while ensuring that the AAnF remains unable to distinguish which account corresponds to which AF. This design aims to preserve privacy and prevent any linkage between accounts and respective AFs.

In the experimental setup, this capability is formalized by allowing the adversary to select two honest UE instances associated with the same UE and two honest AF parties. The adversary must be unable to determine which UE instance interacts with which AF party, thereby ensuring the indistinguishability of interactions between UE instances and AF parties.

EXPERIMENT Expt-AAnF-UNL$_{AKMA}^{\mathcal{A}}(1^n)$
For an AKMA = (Init, Reg, Login), the following security experiment is run between the challenger and the adversary $\mathcal{A}$.

- **Setup.** For each $Role \in \mathcal{UE} \cup \mathcal{AANF} \cup \mathcal{AF}$, the challenger executes the Init algorithm and provides the output PP_{AKMA} to $\mathcal{A}$.
- **Query.** $\mathcal{A}$ interact with oracles defined in Figure 6 and Figure **??** where $\mathcal{A}$ can select an AAnF to query CorruptAAnF.
- **Test.** $\mathcal{A}$ selects one honest UE, corresponding to the corrupted AAnF, which has not been queried by $\mathcal{A}$ via CorruptUE or Compromise oracle. $\mathcal{A}$ also selects two honest AFs, AF_L and AF_R, which have not been queried by $\mathcal{A}$ via CorruptAF oracle. $\mathcal{A}$ selects two unregistered instances $\pi_{UE}^{i_0,0}$ and $\pi_{UE}^{i_1,0}$ (exe = $\perp$) from the UE. As a challenge session, the challenger selects $b \leftarrow \{0,1\}$ and acts as follows:
  - If $b = 0$: $\pi_{UE}^{i_0,0}$ registers with AF_L and $\pi_{UE}^{i_1,0}$ registers with AF_R through the corrupted AAnF.
  - If $b = 1$: $\pi_{UE}^{i_0,0}$ registers with AF_R and $\pi_{UE}^{i_1,0}$ registers with AF_L through the corrupted AAnF.
- **Query.** $\mathcal{A}$ can continue querying the oracles in Figure 6 and Figure **??**, and can require the selected UE in **Test** phase to perform the login, subject to three conditions: the UE is honest, the AFs are honest and the session keys of two UE instances are not queried.
  - ($UE \in \mathcal{L}_{crp} \cup UE \in \mathcal{L}_{cpm}$)=0.
  - ($AF_L \in \mathcal{L}_{crp} \cup AF_R \in \mathcal{L}_{crp}$)=0.
  - ($\pi_{UE}^{i_0,j_0} \in \mathcal{L}_{rvl} \cup \pi_{UE}^{i_1,j_1} \in \mathcal{L}_{rvl}$)=0, ($j_0 > 0, j_1 > 0$).
- **Output.** Finally, $\mathcal{A}$ terminates with output b', and the experiment outputs 1 if $b' = b$.

Definition 7 (AAnF-unlinkability)**.** *An AKMA scheme is AAnF-Unlinkable if for any* PPT *adversary $\mathcal{A}$, there exists a negligible function* negl *such that* $\Pr[\text{Expt-AAnF-UNL}_{\text{AKMA}}^{\mathcal{A}}(1^n) = 1] \leq \frac{1}{2} + \text{negl}(n)$.

D.8 AAnF-Indistinguishability

Informally, this property ensures that a malicious AAnF cannot connect accounts on AFs with the UE's real identity (SUPI). Even though the corrupted AAnF knows the SUPI of each UE it serves, given an account registration or login activity, the AAnF cannot distinguish which specific UE is performing the operation. This property is crucial for preventing the AAnF from building user profiles by linking account activities to real identities.

The key distinction between this experiment and the AAnF-Unlinkable experiment is that here the adversary is given explicit knowledge of which UE corresponds to which SUPI, but must still be unable to determine which UE is accessing which AF during the challenge phase.

EXPERIMENT Expt-AAnF-IND$_{AKMA}^{\mathcal{A}}(1^n)$
For an AKMA = (Init, Reg, Login), the following security experiment is run between the challenger and the adversary $\mathcal{A}$.

- **Setup.** For each $Role \in \mathcal{UE} \cup \mathcal{AANF} \cup \mathcal{AF}$, the challenger executes the Init algorithm and provides the output PP_{AKMA} to $\mathcal{A}$.
- **Query.** $\mathcal{A}$ interacts with oracles defined in Figure 6 and Figure **??**. The adversary is required to select an AAnF as the challenge target and issue a CorruptAAnF query to obtain full control over this AAnF, including access to all SUPI values and the master key K_{AAnF}.

- **Challenge Setup.** $\mathcal{A}$ selects:
  - Two honest UEs: UE_0 and UE_1, both associated with the corrupted AAnF. The adversary knows their respective SUPIs: $SUPI_0$ and $SUPI_1$.
  - Two honest AFs: AF_L and AF_R (not necessarily distinct), which have not been queried via CorruptAF.

  The challenger facilitates the registration of accounts:
  - UE_0 registers an account with AF_L through the corrupted AAnF
  - UE_1 registers an account with AF_R through the corrupted AAnF

  The adversary observes all registration interactions and obtains the public keys and account information generated during these registrations.
- **Test.** The challenger selects a random bit $b \leftarrow \{0,1\}$ and proceeds as follows:
  - If $b=0$: UE_0 performs a login to its account on AF_L
  - If $b=1$: UE_1 performs a login to its account on AF_R

  The adversary observes this login interaction through its control of the AAnF but must determine which UE performed the login.
- **Query.** $\mathcal{A}$ can continue querying the oracles in Figure 6 and Figure **??**, subject to the following restrictions:
  - Neither UE has been corrupted or compromised: $(UE_0 \in \mathcal{L}_{crp} \cup UE_0 \in \mathcal{L}_{cpm} \cup UE_1 \in \mathcal{L}_{crp} \cup UE_1 \in \mathcal{L}_{cpm}) = 0$
  - Neither AF has been corrupted: $(AF_L \in \mathcal{L}_{crp} \cup AF_R \in \mathcal{L}_{crp}) = 0$
  - The session keys of the challenge login sessions have not been revealed
- **Output.** Finally, $\mathcal{A}$ terminates with output b', and the experiment outputs 1 if and only if $b' = b$.

Definition 8 (AAnF-Indistinguishability)**.** *An AKMA protocol satisfies AAnF-IND if for any* PPT *adversary $\mathcal{A}$, there exists a negligible function* negl *such that*

$$\Pr[\mathsf{Expt\text{-}AAnF\text{-}IND}^{\mathcal{A}}_{\mathsf{AKMA}}(1^n) = 1] \leq \frac{1}{2} + \mathsf{negl}(n).$$

D.9 Traceability

The traceability property ensures that even when malicious UEs attempt to manipulate the registration process, the core network can still reliably recover the correct SUPI from the AF's database. This property is crucial for maintaining accountability and regulatory compliance, as it guarantees that malicious users cannot evade identification by tampering with the registration protocol while keeping the AAnF and AF honest.

EXPERIMENT $\mathsf{Expt\text{-}Traceability}(1^n)$

For an AKMA = (Init, Reg, Login, Trace), the following security experiment is run between the challenger and the adversary $\mathcal{A}$.

- **Setup.** The challenger generates $(PK_{core}, SK_{core}) \leftarrow \mathsf{KeyGen}(1^n)$ and executes the Init algorithm for each $Role \in \mathcal{UE} \cup \mathcal{AANF} \cup \mathcal{AF}$, providing the output PP_{AKMA} and PK_{core} to $\mathcal{A}$.
- **Query.** $\mathcal{A}$ can interact with oracles defined in Figure 6 and Figure **??**.
- **Challenge.** The adversary $\mathcal{A}$ selects a target $SUPI^*$ and chooses honest AAnF and AF instances $\pi^{i',0}_{AAnF}$ and $\pi^{i'',0}_{AF}$ with $\pi^{i',0}_{AAnF}.\mathsf{exe} = \pi^{i'',0}_{AF}.\mathsf{exe} = \perp$. Here the $AAnF$ and the AF must remain honest throughout the experiment, i.e., $\{AAnF, AF\} \notin \mathcal{L}_{crp}$.
- **Registration.** $\mathcal{A}$ then acts as a malicious UE and interacts with the chosen AAnF and AF instances through the AAnF and AF oracles to perform the registration protocol. The registration is considered successful if the corresponding AF instance outputs $\pi^{i'',0}_{AF}.\mathsf{exe} = accepted$. During this procedure, $\mathcal{A}$ can still interact with oracles defined in 6 and Figure **??** but cannot query CorruptAAnF and CorruptAF on the $AAnF$ and the AF respectively.
- **Trace.** If the registration succeeds (i.e., $\pi^{i'',0}_{AF}.\mathsf{exe} = accepted$), the challenger executes the trace algorithm on the AF's database: $SUPI' \leftarrow \mathsf{Trace}(\pi^{i'',0}_{AF}.DB_{AF}, SK_{core})$.
- **Output.** The experiment returns 1 if $SUPI' = SUPI^*$, i.e., the traced SUPI matches the SUPI that the honest AAnF used during the registration process (stored in $\pi^{i',0}_{AAnF}.SUPI$), regardless of the malicious UE's behavior.

Definition 9 (Traceability)**.** *An AKMA scheme satisfies Traceability if for any* PPT *adversary $\mathcal{A}$ that can corrupt UE instances, there exists a negligible function negl such that* $\Pr[\mathsf{Expt\text{-}Traceability}(1^n) = 1] \geq 1 - \mathsf{negl}(n)$.

E Security Proof for 2POS

This section presents the security model and formal proof for 2POS.

E.1 Security Model

P_1-Unforgeability. Informally, P_1-Unforgeability ensures that even with access to the private key share of P_1, the adversary cannot forge a signature without interacting with the honest P_2. This definition corresponds to the UE being honest during the registration phase of E2E-AKMA. Because it is assumed that the UE is malicious during the registration phase, corresponding to the adversary registering an account on its own, this is not considered an attack in actual use.

The oracle Π_1 simulates the honest P_2 participant in the 2POS protocol. The oracle takes three inputs: (1) the session identifier *sid*, (2) the message m to be signed, and (3) the message sent by the adversary to P_2. Since the Setup and KeyGen phases have already been executed honestly, Π_1 only runs the signing algorithm of P_2.

EXPERIMENT $\mathsf{Expt\text{-}Sign}^1_{\mathcal{A},\Pi_1}(1^n)$

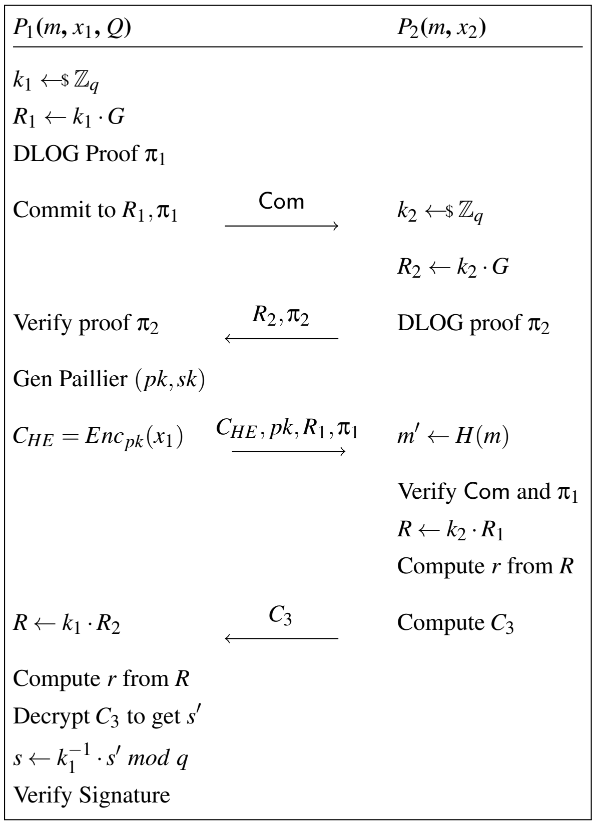


Figure 7: The 2POS Signing Phase

- **Setup.** The challenger honestly executes $2POS$.Setup and $2POS$.KeyGen between P_1 and P_2. The challenger provides the adversary $\mathcal{A}$ with SK_1, PK_2, and PK_S obtained from the honest execution.
- **Sign Query.** The adversary can query the oracle $\Pi_1(sid, m, msg)$ where:
  - sid is the session identifier
  - m is the message to be signed
  - msg is the message sent by $\mathcal{A}$ to the honest P_2

  The oracle Π_1 simulates the honest P_2 in the signing phase and returns the corresponding response. The challenger records all queried messages m in the list $\mathcal{L}_q$.
- **Output.** $\mathcal{A}$ outputs (m^*, σ^*). If $m^* \notin \mathcal{L}_q$ and $\mathsf{Ver}(m^*, \sigma^*, PK_S) = 1$, the adversary wins this game.

Definition 10 (P_1-Unforgeability)**.** *A 2POS scheme is P_1-UNF if for any* PPT *adversary $\mathcal{A}$, there exists a negligible function* negl *such that* $\Pr[\mathsf{Expt\text{-}Sign}^1_{\mathcal{A},\Pi_1}(1^n) = 1] \leq \mathsf{negl}(n)$.

P_2-Unforgeability. Informally, P_2-Unforgeability ensures that even if the adversary $\mathcal{A}$ acts as P_2, as long as P_1 is honest, no signature can be obtained. Unlike P_1-Unforgeability, P_2-Unforgeability allows $\mathcal{A}$ to act maliciously during the Setup phase of 2POS, capturing that even if AAnF misbehaves during the registration phase of E2E-AKMA, valid signatures cannot be forged.

The oracle Π_2^{setup} simulates the honest P_1 participant during the Setup phase. It takes the message sent by the adversary as input and returns the corresponding response from P_1. If the oracle returns $PK_2 \neq \perp$, the challenger uses PK_2 to run KeyGen.

The oracle Π_2^{sign} simulates the honest P_1 participant during the signing phase. It takes three inputs: (1) the session identifier sid, (2) the message m to be signed, and (3) the message sent by the adversary to P_1.

EXPERIMENT $\mathsf{Expt\text{-}Sign}^2_{\mathcal{A},\Pi_2^{setup},\Pi_2^{sign}}(1^n)$

- **Setup.** The adversary $\mathcal{A}$ interacts with oracle $\Pi_2^{setup}(msg)$ to complete $2POS$.Setup, where msg is the message sent by $\mathcal{A}$ to the honest P_1. The oracle simulates the honest P_1's behavior and returns the corresponding response. If the oracle returns $PK_2 \neq \perp$, the challenger uses PK_2 to complete $2POS$.KeyGen and provides the signature public key PK_S to $\mathcal{A}$.
- **Sign.** The adversary can query oracle $\Pi_2^{sign}(sid, m, msg)$ to execute the $2POS$ signing phase, where:
  - sid is the session identifier
  - m is the message to be signed
  - msg is the message sent by $\mathcal{A}$ to the honest P_1

  The oracle Π_2^{sign} simulates the honest P_1 in the signing phase and returns the corresponding response.
- **Output.** $\mathcal{A}$ outputs (m, σ). If $\mathsf{Ver}(m, \sigma, PK_S) = 1$, the adversary wins.

Definition 11 (P_2-Unforgeability)**.** *A 2POS scheme is P_2-UNF if for any* PPT *adversary $\mathcal{A}$, there exists a negligible function* negl *such that* $\Pr[\mathsf{Expt\text{-}Sign}^2_{\mathcal{A},\Pi_2^{\mathsf{setup}},\Pi_2^{\mathsf{sign}}}(1^n) = 1]$ $\leq \mathsf{negl}(n)$.

P_2-Indistinguishability. Informally, P_2-Indistinguishability ensures that a malicious P_2 cannot distinguish which public key PK_S corresponds to which P_1 party during signing interactions, even when both P_1 parties use the same PK_2 from P_2.

This approach is as follows: the challenger selects two honest P_1 parties, P_1^0 and P_1^1. The adversary $\mathcal{A}$ acting as P_2 completes the Setup phase and generates PK_2. Both P_1^0 and P_1^1 use the same PK_2 to generate their respective public keys PK_S^0 and PK_S^1 through KeyGen. The challenger then randomly assigns the correspondence between signing oracles and the P_1 parties based on a random bit b, and the adversary must distinguish the assignment.

EXPERIMENT $\mathsf{Expt\text{-}P_2\text{-}IND}_{\mathcal{A},\Pi}(1^n)$

- **Setup.** The challenger selects two honest P_1 parties, P_1^0 and P_1^1. The adversary $\mathcal{A}$ acting as P_2 completes the Setup phase and generates PK_2.
- **KeyGen.** Both P_1^0 and P_1^1 use the same PK_2 from $\mathcal{A}$ to execute KeyGen and generate their respective public keys PK_S^0 and PK_S^1. The challenger provides both PK_S^0 and PK_S^1 to $\mathcal{A}$.

- **Test.** The challenger chooses $b \leftarrow \{0,1\}$ and sets up two signing oracles Π_0 and Π_1:
  - If $b = 0$: Π_0 corresponds to P_1^0 with PK_S^0, and Π_1 corresponds to P_1^1 with PK_S^1
  - If $b = 1$: Π_0 corresponds to P_1^1 with PK_S^1, and Π_1 corresponds to P_1^0 with PK_S^0
- **Sign.** The adversary $\mathcal{A}$ can query the signing oracles $\Pi_0(sid, m, msg)$ and $\Pi_1(sid, m, msg)$, where:
  - sid is the session identifier
  - m is the message to be signed
  - msg is the message sent by $\mathcal{A}$ to the corresponding P_1 party

  Each oracle simulates the corresponding P_1 party in the signing phase and returns the appropriate response.
- **Output.** Finally, $\mathcal{A}$ terminates with output b', and the experiment outputs 1 if $b' = b$.

Definition 12 (P_2-Indistinguishability)**.** *A 2POS scheme satisfies P_2-IND if for every* PPT *adversary $\mathcal{A}$, there exists a negligible function* negl *such that* $\Pr[\mathsf{Expt\text{-}P_2\text{-}IND}_{\mathcal{A},\Pi}(1^n) = 1] \leq \frac{1}{2} + \mathsf{negl}(n)$.

Untraceable. Informally, Untraceable ensures that when a new P_1 interacts with one of multiple existing P_2 parties to generate a public key, an external adversary cannot determine which specific P_2 was involved in the interaction based solely on the resulting public key and signatures.

This approach is as follows: The challenger first has two P_2 parties generate their respective PK_2^0 and PK_2^1. Two P_1 parties then interact with these P_2 parties to perform KeyGen and Sign operations. In the test phase, a third P_1 interacts with one of the two P_2 parties (chosen by a random bit b) to generate a new public key. The adversary must determine which P_2 was involved in this test interaction.

EXPERIMENT $\mathsf{Expt\text{-}Untrace}_{\mathcal{A},\Pi}(1^n)$

- **Setup.** The challenger selects two honest P_2 parties, P_2^0 and P_2^1, and two honest P_1 parties, P_1^0 and P_1^1. The P_2^0 and P_2^1 execute Setup to generate PK_2^0 and PK_2^1 respectively.
- **KeyGen and Sign.** P_1^0 interacts with P_2^0 to perform KeyGen and generates $PK_S^{0,0}$. P_1^1 interacts with P_2^1 to perform KeyGen and generates $PK_S^{1,1}$. The adversary $\mathcal{A}$ can observe these public keys and request signatures under these keys by selecting messages for the corresponding P_1-P_2 pairs to sign.
- **Test.** The challenger selects a third honest P_1 party, P_1^*, and chooses $b \leftarrow \{0,1\}$. P_1^* interacts with P_2^b to perform KeyGen and generates the test public key PK_S^*. The challenger provides PK_S^* to $\mathcal{A}$.
- **Sign.** $\mathcal{A}$ can request $\mathcal{C}$ to have the test pair (P_1^*, P_2^b) sign messages m chosen by $\mathcal{A}$, and the challenger returns the corresponding signatures to $\mathcal{A}$.
- **Output.** Finally, $\mathcal{A}$ terminates with output b', and the experiment outputs 1 if $b' = b$.

Definition 13 (Untraceable)**.** *A 2POS scheme satisfies Untraceable if for every* PPT *adversary $\mathcal{A}$, there exists a negligible function* negl *such that* $\Pr[\mathsf{Expt\text{-}Untrace}_{\mathcal{A},\Pi}(1^n) = 1] \leq \frac{1}{2} + \mathsf{negl}(n)$.

E.2 Security Proof

Security Notions. We start by introducing a game-based definition to formalize the security of a digital signature scheme $\pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ and an encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$.

These definitions will serve as foundational components in the subsequent protocol design for the 2POS scheme.

EXPERIMENT $\mathsf{EUF\text{-}CMA}_{\mathcal{A},\pi}(1^n)$

1. (vk, sk)$\leftarrow$ $\mathsf{Gen}(1^n)$, and gives the vk to $\mathcal{A}$.
2. (m^*, σ^*)$\leftarrow$ $\mathcal{A}^{\mathsf{Sign}_{sk}(\cdot)}(1^n, vk)$.
3. Let Q be the set of all m queried by $\mathcal{A}$ to its oracle. Then, the output of the experiment equals 1 if and only if $m^* \notin Q$ and $\mathsf{Ver}_{vk}(m^*, \sigma^*)$=1.

Definition 14. *Signature scheme π is existentially unforgeable under chosen-message attacks, or is* EUF-CMA *secure if for every* PPT *adversary $\mathcal{A}$ there exists a negligible function* negl, $Pr[\mathsf{EUF\text{-}CMA}_{\mathcal{A},\pi}(1^n) = 1] \leq \mathsf{negl}(n)$.

EXPERIMENT $\mathsf{EUF\text{-}NMA}_{\mathcal{A},\pi}(1^n)$

1. (vk, sk)$\leftarrow$ $\mathsf{Gen}(1^n)$, and gives the vk to $\mathcal{A}$.
2. $\mathcal{A}$ outputs (m, σ).
3. The output of the experiment equals 1 if and only if $\mathsf{Ver}_{vk}(m, \sigma, vk)$=1.

Definition 15. *Signature scheme π is existential unforgeability under no-message attack, or is EUF-NMA secure if for every* PPT *adversary $\mathcal{A}$ there exists a negligible function* negl, $Pr[\mathsf{EUF\text{-}NMA}_{\mathcal{A},\pi}(1^n) = 1] \leq \mathsf{negl}(n)$.

EXPERIMENT $\mathsf{IND\text{-}CPA}_{\mathcal{A},\Pi}(1^n)$

1. $\mathcal{C}$ generates a key pair (pk, sk) based on some security parameter n, and publishes pk to $\mathcal{A}$. $\mathcal{C}$ retains sk.
2. $\mathcal{A}$ may perform a polynomially bounded number of encryptions or other operations.
3. Eventually, $\mathcal{A}$ submits two distinct chosen plaintexts m_0, m_1 to the challenger $\mathcal{C}$.
4. $\mathcal{C}$ selects a bit $b \in \{0,1\}$ uniformly at random, and sends the challenge-ciphertext $c = \mathsf{Enc}(pk, m_b)$ (the encryption of m_b using the public-key pk) back to $\mathcal{A}$.
5. $\mathcal{A}$ can perform any additional computations or encryptions. Finally, it outputs a guess b' for the value of b.

Definition 16. *A PKE $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ has indistinguishable encryptions under a chosen-plaintext attack, or is CPA-secure if for every* PPT *adversary $\mathcal{A}$ there is a negligible function* negl *such that* $\Pr[\mathsf{IND\text{-}CPA}_{\mathcal{A},\Pi}(1^n) = 1] \leq \frac{1}{2} + \mathsf{negl}(n)$.

P_1-Unforgeability

Theorem 7 (P_1-Unforgeability)**.** *Let $\mathcal{A}$ be an adversary in the* $\mathsf{Expt\text{-}Sign}^1_{\mathcal{A},\Pi}$ *experiment in the ($\mathcal{F}_{\mathsf{com\text{-}zk}}$, $\mathcal{F}_{\mathsf{zk}}$)-hybrid model. The 2POS satisfies the P_1-Unforgeability as defined in Definition 10 assuming the ECDSA is* EUF-CMA *secure.*

Proof. Let $\mathcal{A}$ be a PPT adversary in $\mathsf{Expt\text{-}Sign}^1_{\mathcal{A},\Pi}$; we construct a PPT adversary $\mathcal{S}$ for EUF-CMA. The adversary $\mathcal{S}$ essentially simulates the protocol execution for $\mathcal{A}$, as described in the intuition behind the security of the protocol.

1. In EUF-CMA, adversary $\mathcal{S}$ receives (1^n, Q), where Q is the public verification key for ECDSA.
2. $\mathcal{S}$ randomly selects x_1 as the private key of P_1, and calculates Q_1 and $Q_2 = Q/Q_1$. $\mathcal{S}$ gives x_1, Q_2, and Q to $\mathcal{A}$.
3. Upon receiving a query of Π_1 with the form (sid, m, msg) where sid is a new session identifier, $\mathcal{S}$ queries its signing oracle in experiment EUF-CMA with m and receive a signature (r, s). Using the ECDSA verification procedure, $\mathcal{S}$ computes the Elliptic curve point R. Queries received by $\mathcal{S}$ from $\mathcal{A}$ with identifier sid process as follows:
   (a) The first message (sid, m, msg_1) is processed by first parsing the message msg_1 as (com-prove, $sid\|1$, R_1, k_1). If R_1=$k_1 \cdot G$ then $\mathcal{S}$ sets R_2=$k_1^{-1} \cdot R$; else it choose R_2 at random. $\mathcal{S}$ sets the oracle reply to $\mathcal{A}$ to be the message (proof, $sid\|2$, R_2) that $\mathcal{A}$ expects to receive.
   (b) The second message (sid,m, msg_2) is processed by parsing the message msg_2 as (decom-decom, $sid\|1$, C_{HE}, m'). If $R_1 \neq k_1 \cdot G$ then $\mathcal{S}$ simulates P_2 aborting and the experiment concludes. Otherwise, $\mathcal{S}$ choose a random $\rho \leftarrow \mathbb{Z}_{q^2}$, computes the ciphertext $c_3 \leftarrow \mathsf{Enc}_{pk}([k_1 \cdot s \bmod q] + \rho \cdot q)$, where s is the value from the signature received from ECDSA challenger, and sets the oracle reply to $\mathcal{A}$ to be c_3.
4. Whenever $\mathcal{A}$ halts and output a pair (m^*, σ^*), adversary $\mathcal{S}$ outputs (m^*, σ^*) and halts.

The only difference between the view of $\mathcal{A}$ in a real execution and in the simulation is the way that C_3 is chosen. Specifically, R_2 is distributed identically in both cases because R is randomly generated by the ECDSA challenger in the signature generation, and thus $k_1^{-1} \cdot R$ has the same distribution as $k_2 \cdot G$ (this is the same as in the key generation phase with Q). The zero-knowledge proofs and verifications are also identically distributed in the $\mathcal{F}_{\mathsf{zk}}, \mathcal{F}_{\mathsf{com\text{-}zk}}$-hybrid model. Thus, the only difference is C_3: in the simulation it is encryption of $[k_1 \cdot s \mod q] + \rho \cdot q$, whereas in a real execution it is an encryption of $s' = k_1^{-1} \cdot (m' \cdot r + x) + \rho \cdot q$, where $\rho \in \mathbb{Z}_q$ is random (we stress that all additions here are over the *integers* and not mod q, except for where it is explicitly stated in the protocol description). We stress that the distribution of s' in a real execution is as above, as long as the Paillier key is valid and as long as $C_{HE} = \mathrm{Enc}_{pk}(x_1)$ where $Q_1 = x_1 \cdot G$. As discussed above, these properties are guaranteed by the soundness of the zero-knowledge proofs, and thus the probability that this doesn't hold is negligible.

Thus, we demonstrate that the view of $\mathcal{A}$ remains indistinguishable by establishing that, notwithstanding this difference, the values are indeed *statistically close*. To see this, first observe that by the definition of ECDSA signing, $s = k^{-1} \cdot (m' + rx) = k_1^{-1} \cdot k_2^{-1} \cdot (m + rx) \mod q$. Thus, $k_2^{-1} \cdot (m' + rx) = k_1 \cdot s \mod q$, implying that there exists some $\ell \in \mathbb{N}$ with $0 \leq \ell < q$ such that $k_2^{-1} \cdot (m' + rx) = k_1 \cdot s + \ell \cdot q$. The reason that ℓ is bound between 0 and q is that in the protocol the only operations of a modular reduction are the multiplication of $[k_2^{-1} \cdot r \cdot x_2 \mod q]$ by x_1, and the addition of $[k_2^{-1} \cdot m' \mod q]$. This cannot increase the result by more than q^2. Therefore, the difference between the real execution and simulation with $\mathcal{S}$ is:

1. Real: the ciphertext C_3 encrypts $[k_1 \cdot s \mod q] + \ell \cdot q + \rho \cdot q$
2. Simulated: the ciphertext C_3 encrypts $[k_1 \cdot s \mod q] + \rho \cdot q$

We show that for all $k_1, s, \ell \in \mathbb{Z}_q$, the above values are statistically close (for a random choice of $\rho \in \mathbb{Z}_{q^2}$). In order to see this, fix k_1, s, ℓ, and let v be a value. If $v \neq [k_1 \cdot s \mod q] + \zeta \cdot q$ for some ζ, then neither the real or simulated values can equal v. Else, if $v = [k_1 \cdot s \mod q] + \zeta \cdot q$ for some ζ, then there are three cases:

1. Case $\zeta < \ell$: in this case, v can be obtained in the simulated execution for $\rho < \ell$, but can never be obtained in a real execution.
2. Case $\zeta > q^2 - 1$: in this case, v can be obtained in the real execution for $\rho \geq q^2 - 1 - \ell$, but can never be obtained in a simulated execution.
3. Case $\ell \leq \zeta \leq q^2 - 1$: in this case, v can be obtained in both the real and simulated executions, with identical probability (observe that in both the real and simulated executions, ρ is chosen uniformly in $\mathbb{Z}_{q^2}$).

Recall that the statistical distance between two distributions X and Y over a domain D is defined to be:

$$\Delta(X,Y) = \max_{T \subseteq D} |\Pr[X \in T] - \Pr[Y \in T]|$$

Let X be the values generated in a real execution of the protocol and let Y be the values generated in the simulation with $\mathcal{S}$. Then, taking T to be set of values v for which $\zeta < \ell$, we have that $\Pr[X \in T] = 0$ whereas $\Pr[Y \in T] \leq \frac{q}{q^2} = \frac{1}{q}$ (this holds since $0 \leq \ell < q$ and $\rho \in \mathbb{Z}_{q^2}$). Thus, $\Delta(X,Y) = \frac{1}{q}$, which is negligible. (Taking T to be the set of values v for which $\zeta > q^2 - 1$ would give the same result and are both the maximum since any other values add no difference.) We therefore conclude that the distributions over C_3 in the real and simulated executions are statistically close.

□

P_2-Unforgeability

Theorem 8 (P_2-Unforgeability)**.** *Let $\mathcal{A}$ be an adversary in the* $\mathsf{Expt\text{-}Sign}^2_{\mathcal{A},\Pi}$ *experiment in the ($\mathcal{F}_{\mathsf{com\text{-}zk}}$, $\mathcal{F}_{\mathsf{zk}}$)-hybrid model. The 2POS satisfies the P_2-Unforgeability as defined in Definition 11 assuming the Paillier encryption scheme is* IND-CPA *secure and the ECDSA scheme is EUF-NMA secure.*

Proof. We follow the strategy for the case that P_1 is corrupted, which is to construct a simulator $\mathcal{S}$ that simulates the view of $\mathcal{A}$ while interacting in experiment $\mathsf{EUF\text{-}NMA}$. This simulation is similar to the case in which P_1 is corrupted, with one difference. Recall that the last message from P_2 to P_1 is an encryption C_3. This ciphertext may be maliciously constructed by $\mathcal{A}$, and $\mathcal{S}$ cannot detect this.

Formally, $\mathcal{S}$ can decrypt C_3 because it generates the Paillier public key. However, this strategy will fail since in order to prove computational indistinguishability it is necessary to carry out a reduction to the security of Paillier, meaning that the simulation must be designed to work without knowing the corresponding private key. We solve this problem by simply having $\mathcal{S}$ simulate P_1 aborting at some random point. That is, $\mathcal{S}$ chooses a random $i \in \{1, ...p(n)+1\}$ where $p(n)$ is an upper bound on the number of KeyGen made by $\mathcal{A}$ to Π. If $\mathcal{S}$ is chosen correctly, then the simulation is fine. Since $\mathcal{S}$'s choice of i is correct with probability $\frac{1}{p(n)+1}$, this implies that $\mathcal{S}$ simulates $\mathcal{A}$'s view with that probability. Therefore, $\mathcal{S}$ can forge a signature in $\mathsf{EU\text{-}NMA}$ with a probability that is at least $\frac{1}{p(n)+1}$ times the probability that $\mathcal{A}$ forges a signature in $\mathsf{Expt\text{-}Sign}^2_{\mathcal{A},\Pi}$.

1. In $\mathsf{EUF\text{-}NMA}$, adversary $\mathcal{S}$ receives $(1^n, Q)$, where Q is the public verification key for ECDSA.
2. Let $p(\cdot)$ denote an upper bound on the number of queries that $\mathcal{A}$ makes to Π in experiment $\mathsf{Expt\text{-}Sign}^2_{\mathcal{A},\Pi}$. Then $\mathcal{S}$ chooses a random $i \in \{1, ..., p(n)+1\}$.
3. $\mathcal{A}$ randomly selects x_2 as the private key of P_2, calculates Q_2 and gives Q_2 to $\mathcal{S}$. $\mathcal{S}$ calculate $Q_1 = Q/Q_2$.
4. $\mathcal{S}$ invokes $\mathcal{A}$ on input 1^n and simulates oracle Π for $\mathcal{A}$ in $\mathsf{Expt\text{-}Sign}^2_{\mathcal{A},\Pi}$, answering as described in the following steps. Upon receiving a query of the form (sid, m) where sid is a new session identifier, $\mathcal{S}$ computes the oracle reply to be $(\mathsf{proof\text{-}receipt}, sid\|1)$ as $\mathcal{A}$ expects to receive, and hands it to $\mathcal{A}$. Then queries received by $\mathcal{S}$ from $\mathcal{A}$ with sid are processed as follows:
   (a) The set $\mathcal{S}$ randomly selects k_1 and calculates the corresponding R_1 and π_1. Since it is unnecessary for $\mathcal{S}$ to provide $\mathcal{A}$ with an effective signature, this step can be performed randomly.
   (b) $\mathcal{S}$ generates a valid Paillier key-pair (pk, sk), computes C_{HE}=$\mathsf{Enc}_{pk}(\tilde{x_1})$ for a random $\tilde{x_1}$. $\mathcal{S}$ computes R_1=$k_2^{-1} \cdot R$ and sets the oracle reply to be $(\mathsf{decom\text{-}proof}, sid\|1, C_{HE}, R_1)$ as if coming from $\mathcal{F}^{R_{DL}}_{\mathsf{com\text{-}zk}}$.
   (c) The second message (sid, m_2) is processed by parsing m_2 as C_3. If it is the i-th call by $\mathcal{A}$ to the oracle Π, then $\mathcal{S}$ simulates P_1 aborting. Otherwise, it continues.
5. Whenever $\mathcal{A}$ halts and output a pair (m, σ), adversary $\mathcal{S}$ outputs (m, σ) and halts.

Now, let j be the first call to oracle Π with (sid, C_3) where C_3 is such that P_1 does not obtain a valid signature (r, s) with respect to Q. Then, we argue that if $j = i$, then the only difference between the distribution over $\mathcal{A}$'s view is in a real execution and in the simulated execution by $\mathcal{S}$ is the ciphertext C_{HE} and the possibility that $x_1 \in \mathbb{Z}_q \setminus \{\frac{q}{3}, ..., \frac{2q}{3}\}$ as implicitly defined by the value Q_1. Specifically, in a real execution C_{HE}=$\mathsf{Enc}_{pk}(x_1)$ where $Q_1 = x_1 \cdot G$ and $x_1 \in \{\frac{q}{3}, ..., \frac{2q}{3}\}$, whereas in the simulation C_{HE}=$\mathsf{Enc}_{pk}(\tilde{x_1})$ for a random $\tilde{x_1} \in \{\frac{q}{3}, ..., \frac{2q}{3}\}$ that is independent of Q_1=$x_1 \cdot G$ and Q_1 is uniform in $\mathbb{G}$. Observe, however, that $\mathcal{S}$ does not use the private key for Paillier at all in the simulation. Thus, the indistinguishability of this simulation follows from a straightforward reduction to the indistinguishability of the encryption scheme, under chosen-plaintext attacks. Let A denote the set $\{\frac{q}{3}, ..., \frac{2q}{3}\}$ and note that a randomly chosen $x \in \mathbb{Z}_q$ is in the set A with probability $\frac{1}{3}$.

This proves that

$$|\Pr[\mathsf{EU\text{-}NMA}_{\mathcal{S},\pi}(1^n) = 1 | i = j \wedge x_1 \in A] - \Pr[\mathsf{Expt\text{-}Sign}^2_{\mathcal{A},\Pi}(1^n)]| \leq \mu(n) \tag{1}$$

and so

$$\begin{aligned}&\Pr[\mathsf{Expt\text{-}Sign}^2_{\mathcal{A},\Pi}(1^n)] \\ &\leq \frac{\Pr[\mathsf{EUF\text{-}NMA}_{\mathcal{S},\pi}(1^n) = 1 | i = j \wedge x_1 \in A]}{\Pr[i = j \wedge x_1 \in A]} + \mu(n) \\ &\leq \frac{\Pr[\mathsf{EUF\text{-}NMA}_{\mathcal{S},\pi}(1^n) = 1]}{\frac{1}{3} \cdot \frac{1}{p(n)+1}} + \mu(n)\end{aligned} \tag{2}$$

where the second inequality holds because i is chosen independently of j, and because Q is independent of Q_2 and so Q_1 is uniformly distributed in $\mathbb{G}$. Thus,

$$\Pr[\mathsf{EUF\text{-}NMA}_{\mathcal{S},\pi}(1^n) = 1] \geq \frac{\Pr[\mathsf{Expt\text{-}Sign}^2_{\mathcal{A},\Pi}(1^n)]}{3p(n)+3} - \mu(n) \tag{3}$$

This means that if $\mathcal{A}$ can solve a 2POS problem, then $\mathcal{S}$ can extract a solution to an EU-NMA problem from it with non-negligible probability, which will break the security of the ECDSA scheme.

□

P_2-Indistinguishability.

Theorem 9 (P_2-Indistinguishability)**.** *Let $\mathcal{A}$ be an adversary in the* $\mathsf{Expt\text{-}P_2\text{-}IND}_{\mathcal{A},\Pi}$ *experiment in the* ($\mathcal{F}_{\mathsf{com\text{-}zk}}$, $\mathcal{F}_{\mathsf{zk}}$)*-hybrid model. Assume that the Paillier encryption scheme is* IND-CPA *secure. Then the probability of $\mathcal{A}$ winning is negligible.*

Proof. Let $\mathcal{A}$ be a PPT adversary in $\mathsf{Expt\text{-}P_2\text{-}IND}_{\mathcal{A},\Pi}(1^n)$; we construct a PPT adversary $\mathcal{B}$ against the $\mathsf{IND\text{-}CPA}$ security of the Paillier encryption scheme.

The key observation is that during the signing protocol, the only information that P_2 (controlled by $\mathcal{A}$) receives about P_1's private key share is through the Paillier ciphertext

$C_{HE} = \mathsf{Enc}_{pk}(x_1)$ sent in the third message of the signing protocol. The indistinguishability of different P_1 parties therefore reduces to the indistinguishability of their encrypted private key shares.

Game 0: This is the original P_2-Indistinguishability experiment. The challenger selects two honest P_1 parties, P_1^0 and P_1^1, with private key shares x_1^0 and x_1^1 respectively. Both parties use the same PK_2 from the adversary to generate public keys $PK_S^0 = x_1^0 \cdot Q_2$ and $PK_S^1 = x_1^1 \cdot Q_2$. During signing queries, the adversary receives $C_{HE}^0 = \mathsf{Enc}_{pk}(x_1^0)$ when interacting with Π_0 and $C_{HE}^1 = \mathsf{Enc}_{pk}(x_1^1)$ when interacting with Π_1, where the correspondence depends on the challenge bit b.

Game 1: We modify Game 0 by replacing the ciphertext from one of the oracles. Specifically, when $b = 0$, we replace $C_{HE}^0 = \mathsf{Enc}_{pk}(x_1^0)$ with $C_{HE}^0 = \mathsf{Enc}_{pk}(x_1^1)$ in all signing interactions with Π_0. When $b = 1$, we replace $C_{HE}^1 = \mathsf{Enc}_{pk}(x_1^1)$ with $C_{HE}^1 = \mathsf{Enc}_{pk}(x_1^0)$ in all signing interactions with Π_1.

If the adversary can distinguish between Game 0 and Game 1, we can construct an adversary $\mathcal{B}$ that breaks the IND-CPA security of Paillier encryption as follows:

$\mathcal{B}$ receives a Paillier public key pk from the IND-CPA challenger and sets this as the public key used in the signing protocol. $\mathcal{B}$ then submits (x_1^0, x_1^1) as the challenge messages to the IND-CPA challenger and receives back a challenge ciphertext $C^* = \mathsf{Enc}_{pk}(x_1^\beta)$ for some unknown bit β.

$\mathcal{B}$ simulates the P_2-Indistinguishability experiment for $\mathcal{A}$, using C^* as the appropriate ciphertext in the signing protocol (either C_{HE}^0 or C_{HE}^1 depending on the challenge bit b). When $\mathcal{A}$ outputs its guess b', $\mathcal{B}$ uses this information to determine β and outputs its guess to the IND-CPA challenger.

If $\beta = 0$ (i.e., $C^* = \mathsf{Enc}_{pk}(x_1^0)$), then $\mathcal{B}$ is simulating Game 0. If $\beta = 1$ (i.e., $C^* = \mathsf{Enc}_{pk}(x_1^1)$), then $\mathcal{B}$ is simulating Game 1. Therefore, any advantage that $\mathcal{A}$ has in distinguishing the games translates directly to $\mathcal{B}$'s advantage in the IND-CPA game.

Analysis: In Game 1, the ciphertexts sent to P_2 are now independent of the actual correspondence between oracles and P_1 parties. Specifically, both oracles effectively send encryptions of the same private key share, making the challenge bit b information-theoretically hidden from P_2's view. Therefore, the adversary's advantage in Game 1 is negligible.

Since the Paillier encryption scheme is IND-CPA secure, the advantage of any PPT adversary in distinguishing Game 0 from Game 1 is negligible. Combined with the negligible advantage in Game 1, we conclude that the advantage of any PPT adversary in the original P_2-Indistinguishability experiment is negligible.

□

Untraceability.

Theorem 10 (Untraceability). *Let $\mathcal{A}$ be an adversary in the* $\mathsf{Expt\text{-}Untrace}_{\mathcal{A},\Pi}$ *experiment in the ($\mathcal{F}_{\mathsf{com\text{-}zk}}$, $\mathcal{F}_{\mathsf{zk}}$)-hybrid model. Assume that the private key shares of P_1 and P_2 are chosen uniformly at random from $\mathbb{Z}_q$. Then the probability of $\mathcal{A}$ winning is negligible.*

Proof. The key insight is that the 2POS public key $PK_S = Q = x_1 \cdot Q_2$ is an elliptic curve point that results from the scalar multiplication of a uniformly random private key share x_1 with the public key Q_2 of P_2. When x_1 is uniformly random, the resulting point Q is also uniformly distributed over the elliptic curve group, making it information-theoretically impossible to recover Q_2 from Q alone.

Game 0: This is the original untraceability experiment. The challenger has two P_2 parties with public keys $Q_2^0 = x_2^0 \cdot G$ and $Q_2^1 = x_2^1 \cdot G$, where $x_2^0, x_2^1 \leftarrow\$ \mathbb{Z}_q$. In the test phase, a new P_1^* with private key share $x_1^* \leftarrow\$ \mathbb{Z}_q$ interacts with P_2^b to generate the test public key $PK_S^* = x_1^* \cdot Q_2^b$.

The adversary observes: - Reference public keys: $PK_S^{0,0} = x_1^0 \cdot Q_2^0$ and $PK_S^{1,1} = x_1^1 \cdot Q_2^1$ - Test public key: $PK_S^* = x_1^* \cdot Q_2^b$ - Signatures under these keys

Game 1: We modify Game 0 by replacing the test public key PK_S^* with a uniformly random elliptic curve point $R \leftarrow\$ \mathbb{G}$, where $\mathbb{G}$ is the elliptic curve group of order q.

Claim: Game 0 and Game 1 are information-theoretically indistinguishable.

Proof of Claim: Since x_1^* is chosen uniformly at random from $\mathbb{Z}_q$, and the scalar multiplication $x_1^* \cdot Q_2^b$ maps uniformly random scalars to uniformly random group elements (due to the properties of elliptic curve groups), the distribution of $PK_S^* = x_1^* \cdot Q_2^b$ is identical to the uniform distribution over $\mathbb{G}$.

More formally, for any fixed $Q_2^b \in \mathbb{G}$ and any target point $T \in \mathbb{G}$, there exists exactly one value $x_1^* \in \mathbb{Z}_q$ such that $x_1^* \cdot Q_2^b = T$. Since x_1^* is chosen uniformly from $\mathbb{Z}_q$, each point in $\mathbb{G}$ has equal probability $1/q$ of being the result of $x_1^* \cdot Q_2^b$.

Therefore, the adversary's view in Game 0 is identical to its view in Game 1, regardless of the value of b.

Game 2: In Game 1, since the test public key is uniformly random and independent of the challenge bit b, we can further simplify by having the challenger simulate signatures for the test key using a random private key $x_{sim} \leftarrow\$ \mathbb{Z}_q$ such that $x_{sim} \cdot G = PK_S^*$.

This simulation is perfect because: 1. The test public key PK_S^* is uniformly random 2. For any uniformly random point PK_S^*, there exists a corresponding private key x_{sim} 3. Signatures generated using x_{sim} are indistinguishable from those generated through the 2POS protocol

Analysis: In Game 2, the test public key and its associated signatures are completely independent of the challenge bit b and the specific P_2 party involved. The adversary's view contains no information that could help distinguish between P_2^0 and P_2^1.

Therefore, the adversary's advantage in Game 2 is exactly 0, as the challenge bit b is information-theoretically hidden.

Since all game transitions are information-theoretic (based on the uniform randomness of private key shares), the adversary's advantage in the original untraceability experiment is negligible.

Specifically, the adversary cannot determine which P_2 party was involved in generating the test public key because: 1. The mapping $x_1^* \mapsto x_1^* \cdot Q_2^b$ produces uniformly random outputs regardless of Q_2^b 2. No information about Q_2^b can be recovered from the uniformly random point PK_S^* 3. The signatures provide no additional information about the underlying P_2 party

□

F Security Proof of E2E-AKMA

F.1 Authentication.

Theorem 11 (Authentication)**.** *The E2E-AKMA satisfies the Authentication as defined in Definition 4, assuming 2POS is correct, AFs avoid identical challenges, and 2POS is P_1-UNF and P_2-UNF.*

Proof. We separately discuss the following three scenarios in which the adversary successfully attacks.

Case 1: $\pi_{UE}^{i,j}$ and $\pi_{AF}^{i'',j''}$ form a partner relationship and are intentional partners of each other, $\pi_{UE}^{i,j}.AF_{par} = AF$, $\pi_{AF}^{i'',j''}.UE_{par} = UE$. And without adversary involvement, $UE \notin \mathcal{L}_{cpm}, HN \notin \mathcal{L}_{crp}$. But the execution state of at least one of them is not accepted, $\pi_{UE}^{i,j}.\mathsf{exe} \neq accepted$ or $\pi_{AF}^{i'',j''}.\mathsf{exe} \neq accepted$. Since UE, HN, and AF are honest, the adversary cannot send messages through the UE, HN, or AF oracles. The adversary can only select the i-th account of UE and AF and execute the protocol through the $\mathsf{Start}((UE,i),AF)$ oracle. The Start oracle will correctly forward the messages communicated by all the parties, so the transcripts of $\pi_{UE}^{i,j}$ and $\pi_{AF}^{i'',j''}$ will match. For the signature σ_{2POS} sent by an honest UE, due to the perfect correctness of the signature scheme, the honest AF will also verify it, $1 \leftarrow \mathsf{Ver}(m, \sigma_{2POS}, PK_S)$, so the adversary's advantage in winning this game is 0.

Case 2: $\pi_{UE}^{i,j}$ accepted with no partner, or accepted with more than one partner, and its intentional partner was not corrupted. We divide the discussion into the following two cases.

1. $\pi_{UE}^{i,j}$ accepted with no partner. Then, according to the Game 1, $\pi_{UE}^{i,j}$ will have an instance $\pi_{AF}^{i'',j''}$ as its intentional partner in the registered AF party. Then the adversary needs to send a challenge C to $\pi_{UE}^{i,j}$ in the name of $\pi_{AF}^{i'',j''}$ by invoking the $\mathsf{UE}((i,j),\pi_{AF}^{i'',j''},C)$ oracle. However, since the UE oracle checks whether the party to which the instance provided by the adversary belongs has been corrupted, and AF has not been queried for corruption by the adversary, the adversary cannot successfully send C to UE through this oracle. Therefore, the probability of this event occurring is 0.
2. $\pi_{UE}^{i,j}$ accepted with more than one partner. This situation corresponds to two honest AF instance sending two identical challenges for two login attempts to the same UE account. Assume that the same account of a UE logs in a total of n times, and the length of the challenge is λ. Then, during these n login attempts, the probability of two AF insatnce generating the same challenge is $n^2 \cdot 2^{-\lambda}$.

Case 3: The intentional partner of $\pi_{AF}^{i'',j''}$ is not $\emptyset$ and $\pi_{AF}^{i'',j''}$ accepted with no partner, or accepted with more than one partner, and its intentional partner was not corrupted.

1. $\pi_{AF}^{i'',j''}$ accepted with no partner. This situation corresponds to the case where the adversary causes $\pi_{AF}^{i'',j''}$ to accept without corrupting the intentional partner of $\pi_{AF}^{i'',j''}$. Specifically, it can be divided into two scenarios: the adversary, acting as participant $P1$ in the 2POS protocol, forges a signature; and the adversary, acting as participant $P2$ in the 2POS protocol, forges a signature.
First, we discuss the case where the $2POS$ adversary $\mathcal{B}_1$ extracts a solution for P_1-Unforgeability from E2E-AKMA. $\mathcal{B}_1$ obtains SK_1, PK_2, and PK_S from the P_1-Unforgeability challenger. $\mathcal{B}_1$ guesses which account among the l UEs and m accounts $\mathcal{A}$ will successfully forge the signature for. $\mathcal{B}_1$ replaces the K_{UE} of that account with SK_1, the long-term key of the UE's corresponding HN with PK_2, and the account key of the UE with PK_S.
Due to P_1-Unforgeability, the SK_1 chosen honestly by the 2POS challenger is random, and the SK_1 chosen by the honest UE in E2E-AKMA is also random; therefore, their distributions are identical, and $\mathcal{A}$ will not detect any inconsistency at this step.
In 2POS, PK_2 is randomly generated by P_2, while in E2E-AKMA, PK_2 is derived by HN using the master key and the user's SUPI through a KDF. Treating the KDF as a random oracle, the distribution of PK_2 in E2E-AKMA is identical to that of PK_2 in 2POS. Therefore, the adversary will not detect the replacement of PK_2.
In 2POS, PK_S is calculated jointly by SK_1 and PK_2. Similarly, in E2E-AKMA, PK_S is computed using the same method. Therefore, their distributions are identical, and adversaries cannot distinguish between them.
Assuming that $\mathcal{B}_1$ correctly guesses which account's signature $\mathcal{A}$ would forge, $\mathcal{B}_1$ can extract the solution to P_1-Unforgeability from the execution of the protocol. Assume there are l UEs, and each UE has m accounts. The adversary's probability of success is given by $l \cdot m \cdot \mathsf{Adv}_{2POS}^{P_1-UNF}(\mathcal{B}_1)$.
Next, we discuss the scenario in which the adversary $\mathcal{B}_2$ of 2POS extracts the solution to P_2-Unforgeability from E2E-AKMA. $\mathcal{B}_2$ needs to predict which HN $\mathcal{A}$ will select from the m' HNs, as well as which registration instance of the selected HN $\mathcal{A}$ will use to forge a signature. $\mathcal{A}$ acting as HN, interacts with UE and selects its private key share SK_2. $\mathcal{B}_2$ uses the SK_2 selected by $\mathcal{A}$ as $2POS$'s SK_2, forwards the message sent by $\mathcal{A}$ to UE to $2POS$'s Π_2, and relays the response from Π_2 back to $\mathcal{A}$. In this way, since all the

information observed by $\mathcal{A}$ is generated by the genuine P_1, $\mathcal{A}$ will not notice any anomalies. If $\mathcal{A}$ requires UE to execute the registration and login phases, UE only needs to honestly follow the execution of $2POS$, the HN forwards the message sent by the UE to the challenger of 2POS and relays its response back to the UE. Therefore, during the registration phase conducted by $\mathcal{A}$, the PK_2 it generates corresponds to the PK_2 of the adversary $\mathcal{B}_2$'s P_2, while the key observed by $\mathcal{A}$ from UE is, in fact, the key of P_1 in $2POS$. At this point, as long as $\mathcal{A}$ produces a valid signature, it can serve as a condition for $\mathcal{B}_2$ to succeed. Assume there are l' HN, and each HN corresponds to m' UE registered accounts. The adversary's probability of success is given by $l' \cdot m' \cdot \mathsf{Adv}_{2POS}^{P_2-UNF}(\mathcal{B}_2)$

2. $\pi_{AF}^{i'',j''}$ accepted with more than one partner. This situation corresponds to an honest AF generating the same challenge for two UE instances. Assume an AF executes n'' logins with all UE instances, and the length of the challenge is λ. Then the probability of this occurrence is $n''^2 \cdot 2^{-\lambda}$.

□

F.2 Key-Indistinguishability

Theorem 12 (Key-indistinguishability)**.** *Consider the hash function as a random oracle. The E2E-AKMA satisfies the Key-indistinguishability as defined in Definition 5, assuming 2POS is correct, AF avoids identical challenges, 2POS is P_1-UNF and P_2-UNF, RO outputs differ for distinct inputs, and the guessing the challenge is negligible.*

Proof. Treating the hash function as a random oracle, if the adversary successfully distinguishes the real key from the random key, it corresponds to the adversary having queried the hash value of $(C||\sigma_{2POS})$. We discuss the upper bound of the adversary's success in an attack, specifically the probability of the adversary successfully attacking key indistinguishability while the protocol satisfies authenticity. The probability of the adversary's success is the sum of $\mathsf{Adv}_{\text{E2E-AKMA}}^{\text{Auth}}$ and the probability of success that we analyze next.

For $\pi_{UE}^{i,j}$ and $\pi_{AF}^{i'',j''}$ that already satisfy the property of Authentication, regardless of which instance the adversary tests, there are two possible situations.

The first situation is when the adversary has queried $\mathsf{Compromise}(UE)$. Since the UE and AF instances satisfy Authentication, the adversary cannot forge a signature. If the adversary queries the session key to the RO, it corresponds to the adversary correctly guessing the challenge.

The second situation is when the adversary has queried $\mathsf{CorruptHN}(HN)$. Similar to the previous case, this corresponds to the adversary successfully guessing the challenge.

□

F.3 AF-Unlinkable.

Theorem 13 (AF-unlinkable)**.** *Consider the KDF as a random oracle. The E2E-AKMA satisfies the AF-Unlinkable as defined in Definition 6 assuming the 2POS is Untraceable and the re-randomizable encryption scheme is IND-CPA secure.*

Proof. The key insight is that AF-Unlinkability in E2E-AKMA directly reduces to the Untraceability property of the underlying 2POS scheme. Since both UE_0 and UE_1 correspond to the same AAnF (acting as P_2 in 2POS), the adversary's ability to link accounts depends entirely on their capability to distinguish between different 2POS public keys generated by the same P_2 party, which is precisely what 2POS Untraceability prevents.

We construct a reduction adversary $\mathcal{B}$ that uses any AF-Unlinkability adversary $\mathcal{A}$ to break the Untraceability of 2POS.

Game 0: This is the original AF-Unlinkable experiment, so $\Pr_0 = \mathsf{Adv}_{\text{E2E-AKMA}}^{\text{AF-UNL}}(\mathcal{A})$.

The adversary $\mathcal{A}$ corrupts multiple AFs and observes two honest UEs, UE_0 and UE_1, both corresponding to the same AAnF $AAnF^*$, registering and logging in with these corrupted AFs. In the test phase, one of the UEs (chosen by challenge bit b) registers with a target AF. The adversary must determine which UE was selected.

Reduction to 2POS Untraceability: $\mathcal{B}$ acts as the challenger in the AF-Unlinkability experiment and embeds the 2POS Untraceability challenge as follows:

1. *Setup Phase*: $\mathcal{B}$ receives the 2POS Untraceability challenge, which includes a single P_2 public key PK_2 (corresponding to $AAnF^*$), and reference public keys $PK_S^{0,0}$ and $PK_S^{1,0}$ from two previous interactions between P_1^0, P_1^1 (corresponding to UE_0, UE_1) and P_2.

2. *Embedding the Challenge*: $\mathcal{B}$ maps the 2POS challenge to the AKMA setting: - The single P_2 party corresponds to $AAnF^*$ - The two P_1 parties correspond to UE_0 and UE_1 - The reference public keys $PK_S^{0,0}$ and $PK_S^{1,0}$ become the 2POS public keys from previous UE registrations with corrupted AFs - The test public key PK_S^* from the 2POS challenge becomes the public key generated when the challenge UE registers with the target AF

3. *Simulation*: When $\mathcal{A}$ queries UE registrations and logins: - For registrations with corrupted AFs, $\mathcal{B}$ uses the reference interactions from the 2POS challenge - For the test registration, $\mathcal{B}$ uses the test public key PK_S^* and corresponding signatures from the 2POS challenge - All AAnF signatures are generated consistently using the same $AAnF^*$ key material - All other protocol components (challenges, signatures, database updates) are simulated normally

4. *Output*: When $\mathcal{A}$ outputs its guess b' for which UE performed the test registration, $\mathcal{B}$ forwards this as its guess for which P_1 was involved in the 2POS Untraceability challenge.

Analysis of Information Available to AF: The crucial observation is that the information available to corrupted AFs is limited to:

1. *2POS Public Keys (PK_S)*: These are the primary identifiers stored in AF databases. By the Untraceability property of 2POS, these public keys do not reveal which specific P_1 (i.e., which UE) was involved in their generation, even when the P_2 party (AAnF) is the same.

2. *AAnF Signatures*: All signatures are generated by the same $AAnF^*$, so they provide no information to distinguish between UE_0 and UE_1. The signature verification keys and patterns are identical for both UEs.

3. *Encrypted Tags (tag)*: While these tags encrypt the SUPI and are included in the AAnF's signature, they do not provide linking information to corrupted AFs because: - The tags are encryptions under the core network's public key PK_{core}, which is not available to AFs - By the IND-CPA security of the re-randomizable encryption scheme, different SUPI values result in computationally indistinguishable ciphertexts - Even though corrupted AFs may not follow the protocol and perform re-randomization, they cannot extract meaningful information from the original encrypted tags due to the semantic security of the encryption

Game 1: We modify Game 0 by replacing the KDF-derived values with uniformly random values. Since KDF is modeled as a random oracle, its outputs are indistinguishable from uniform random values. This change has no effect on the adversary's advantage: $|\Pr_1 - \Pr_0| = 0$.

Perfect Reduction: The simulation provided by $\mathcal{B}$ is perfect because:

1. *Distribution Preservation*: The 2POS public keys and signatures in the AKMA protocol have the same distribution as those in the 2POS Untraceability experiment.

2. *AAnF Consistency*: Since both UEs correspond to the same AAnF, all AAnF-related information (signatures, verification keys) is identical and provides no distinguishing information.

3. *Information Equivalence*: The information available to $\mathcal{A}$ in the AF-Unlinkability experiment is exactly the information available to the 2POS Untraceability adversary, plus encrypted tags and AAnF signatures that provide no additional distinguishing information.

4. *Challenge Correspondence*: The challenge in AF-Unlinkability (determining which UE registered) directly corresponds to the challenge in 2POS Untraceability (determining which P_1 was involved when P_2 is fixed).

Therefore, we have:

$$\mathrm{Adv}^{\text{AF-UNL}}_{\text{E2E-AKMA}}(\mathcal{A}) = \mathrm{Adv}^{\text{Untrace}}_{\text{2POS}}(\mathcal{B})$$

Since the 2POS scheme satisfies Untraceability, $\mathcal{B}$'s advantage is negligible, which implies that $\mathcal{A}$'s advantage in the AF-Unlinkability experiment is also negligible.

□

F.4 AAnF-Unlinkable

Theorem 14 (AAnF-unlinkable)**.** *The E2E-AKMA satisfies the AAnF-Unlinkable as defined in Definition 7 assuming the 2POS satisfies P_2-IND, the hash function H is modeled as a random oracle, and the re-randomizable encryption scheme satisfies malformed ciphertext indistinguishability.*

Proof. The key insight is that AAnF-Unlinkability in E2E-AKMA reduces to the P_2-Indistinguishability property of the underlying 2POS scheme, combined with the hiding property of the random oracle for AF identifiers. The corrupted AAnF's ability to link UE instances to specific AFs depends on two main information sources: (1) the 2POS signing interactions, and (2) the AF identifiers embedded in hash values h_1. We show that neither source provides sufficient distinguishing information.

We construct a reduction adversary $\mathcal{B}$ that uses any AAnF-Unlinkability adversary $\mathcal{A}$ to break either the P_2-IND property of 2POS or the random oracle assumption.

Game 0: This is the original AAnF-Unlinkable experiment, so $\Pr_0 = \mathrm{Adv}^{\text{AAnF-UNL}}_{\text{E2E-AKMA}}(\mathcal{A})$.

The adversary $\mathcal{A}$ corrupts an AAnF and observes two instances $\pi_{UE}^{i_0,0}$ and $\pi_{UE}^{i_1,0}$ of an honest UE registering with two honest AFs AF_L and AF_R. In the test phase, the challenger chooses a random bit b to determine the assignment: if $b = 0$, then $\pi_{UE}^{i_0,0}$ registers with AF_L and $\pi_{UE}^{i_1,0}$ registers with AF_R; if $b = 1$, the assignment is swapped.

Game 1: Eliminating AF Identity Information We first modify Game 0 to eliminate the information leakage from AF identifiers in hash values h_1. Since H is modeled as a random oracle and the challenge C is chosen uniformly at random by honest AFs, we can replace the hash computation with truly random values.

Specifically, we modify the protocol so that when computing $h_1 = H(id_{AF}\|C\|PK_S\|UID)$, instead of using the actual id_{AF}, we use a random string r_{AF} of the same length. Since C is uniformly random and H is a random oracle, the distribution of h_1 remains unchanged from the adversary's perspective.

The key observation is that h_1 acts as a commitment to id_{AF} due to the random challenge C. Since C is chosen uniformly at random by the honest AF and is unknown to the AAnF until the protocol execution, the hash value h_1 reveals no information about which specific AF is involved in the registration.

By the random oracle property, $|\Pr_1 - \Pr_0| \leq \mathrm{negl}(n)$.

Game 2: Reducing to 2POS P_2-IND Having eliminated the AF identity information, we now focus on the 2POS interactions. We construct a reduction $\mathcal{B}$ to the P_2-IND property of 2POS.

$\mathcal{B}$ acts as the challenger in the AAnF-Unlinkability experiment and embeds the P_2-IND challenge as follows:

1. *Setup Phase*: $\mathcal{B}$ receives the P_2-IND challenge setup. As the corrupted AAnF, $\mathcal{B}$ generates PK_2 using the master key K_{AAnF} and the UE's SUPI: $SK_2 \leftarrow \mathrm{KDF}(K_{AAnF}, SUPI)$,

$PK_2 \leftarrow SK_2 \cdot G$. This PK_2 is provided to both P_1 parties (corresponding to the two UE instances). The P_1 parties generate their respective public keys PK_S^0 and PK_S^1 using the same PK_2.

2. *Embedding the Challenge*: $\mathcal{B}$ maps the P_2-IND challenge to the AKMA setting: - The two P_1 parties correspond to $\pi_{UE}^{i_0,0}$ and $\pi_{UE}^{i_1,0}$ - The public keys PK_S^0 and PK_S^1 become the account public keys for the two UE instances - The signing oracles Π_0 and Π_1 from the P_2-IND experiment correspond to the 2POS signing interactions during UE registrations

3. *Simulation*: When $\mathcal{A}$ observes the UE registrations: - For the hash computation $h_2 = H(id_{AF}\|C)$ in step 3 of the registration phase, $\mathcal{B}$ forwards the signing request to the appropriate oracle Π_0 or Π_1 based on the challenge bit from the P_2-IND experiment - All AAnF signatures on h_1 are generated consistently using the same SK_{AAnF} - The encrypted tags are generated identically since both UE instances correspond to the same SUPI - All other protocol components (UIDs, challenges) are simulated normally

4. *Output*: When $\mathcal{A}$ outputs its guess b' for the UE-AF assignment, $\mathcal{B}$ forwards this as its guess for the P_2-IND challenge.

Analysis of Information Available to AAnF: The crucial observation is that after eliminating AF identity information, the corrupted AAnF's view is limited to:

1. *2POS Signing Interactions*: During the signing phases for $h_2 = H(id_{AF}\|C)$, the AAnF participates in 2POS signing protocols with the UE instances. However, by the P_2-IND property, the AAnF cannot distinguish which UE instance (using which private key SK_1) is involved in each signing interaction, even though both instances use the same PK_2 derived from the same SUPI.

2. *Account UIDs*: Each UE instance generates a different random UID, but these are generated independently and provide no linking information to the AAnF about which AF is being accessed.

3. *Encrypted Tags*: The AAnF generates encrypted tags $tag = \text{Enc}(PK_{core}, SUPI)$ for each registration. Since both UE instances correspond to the same UE (and thus the same SUPI), the encrypted tags contain identical plaintext information and provide no distinguishing capability.

Handling Malicious Tag Generation: Even if the malicious AAnF attempts to generate malformed or distinguishable encrypted tags to track UE instances, this attack is thwarted by the AF's re-randomization process:

- When the honest AFs receive the encrypted tags, they perform re-randomization: $tag' \leftarrow \text{ReRand}(tag)$ - By the malformed ciphertext indistinguishability property, even if the AAnF provides malicious ciphertexts, the re-randomized results tag' are computationally indistinguishable from random ciphertexts - Therefore, the AAnF cannot use the tags stored in AF databases to link UE instances to specific AFs

Perfect Reduction: The simulation provided by $\mathcal{B}$ is perfect because:

1. *Distribution Preservation*: The 2POS signing interactions in the AKMA protocol have the same distribution as those in the P_2-IND experiment.

2. *Information Equivalence*: After eliminating AF identity leakage through the random oracle modification, the information available to $\mathcal{A}$ is exactly the information available to the P_2-IND adversary.

3. *Challenge Correspondence*: The challenge in AAnF-Unlinkability (determining which UE instance registers with which AF) directly corresponds to the challenge in P_2-IND (determining the assignment of signing oracles to P_1 parties).

Therefore, we have:

$$\text{Adv}_{\text{E2E-AKMA}}^{\text{AAnF-UNL}}(\mathcal{A}) \leq \text{Adv}_{\text{2POS}}^{P_2\text{-IND}}(\mathcal{B}) + \text{negl}(n)$$

Since the 2POS scheme satisfies P_2-IND, the hash function is a random oracle, and the re-randomizable encryption satisfies malformed ciphertext indistinguishability, $\mathcal{B}$'s advantage is negligible, which implies that $\mathcal{A}$'s advantage in the AAnF-Unlinkability experiment is also negligible.

□

F.5 AAnF-Indistinguishability

Theorem 15 (AAnF-indistinguishability)**.** *The E2E-AKMA satisfies the AAnF-Indistinguishability property as defined in Definition 8, assuming the 2POS scheme satisfies the Untraceable property and the re-randomizable encryption scheme satisfies malformed ciphertext indistinguishability.*

Proof. The security of AAnF-Indistinguishability is based on the observation that the information available to the corrupted AAnF from AF's public data cannot be used to trace back to specific UE identities. We show that each component of the public information provides no linking capability.

We construct the proof through a sequence of games, demonstrating that the adversary's advantage is negligible.

Game 0: This is the original AAnF-Indistinguishability experiment, so $\Pr_0 = \text{Adv}_{\text{E2E-AKMA}}^{\text{AAnF-IND}}(\mathcal{A})$.

The adversary $\mathcal{A}$ corrupts an AAnF and observes two UEs (UE_0 and UE_1) registering accounts with two AFs (AF_L and AF_R). The adversary knows the SUPIs of both UEs but must determine which UE performed the challenge login based on the information stored in AF databases.

Analysis of Information Available to AAnF: The corrupted AAnF can potentially access the following information from AF databases after registration:

1. *Account UIDs*: Each registration generates a random UID that serves as part of the username. 2. *Public Keys (PK_S)*: The 2POS public keys generated during registration. 3. *Re-randomized Encrypted Tags (tag')*: The encrypted SUPI values after AF's re-randomization.

We analyze each information source:

Game 1: UID Analysis The UIDs are generated uniformly at random by the UE during registration: $UID \leftarrow \{0,1\}^{\lambda}$. These values are independent of the UE's SUPI and provide no information about which specific UE generated them. Since UIDs are chosen randomly and independently for each account, they cannot be used to link accounts to specific UEs.

Therefore, $|\Pr_1 - \Pr_0| = 0$.

Game 2: Public Key Unlinkability We now consider the public keys PK_S stored in AF databases. Each UE instance generates a different PK_S for each account through the 2POS key generation process: $(SK_1, PK_S) \leftarrow \text{2POS.KeyGen}(PK_2)$.

By the Untraceable property of the 2POS scheme, even though the corrupted AAnF participated in the key generation process (as P_2), it cannot determine which specific P_1 (UE instance) was involved in generating each PK_S. The Untraceable property ensures that public keys generated by different UE instances appear independent and unlinkable to external observers, including the AAnF.

More formally, we can construct a reduction $\mathcal{B}$ to the Untraceable property: - $\mathcal{B}$ uses the two UE instances as P_1^0 and P_1^1 in the Untraceable experiment - The corrupted AAnF acts as P_2 in both interactions - The challenge public key PK_S^* corresponds to the public key generated in the challenge login - If $\mathcal{A}$ can distinguish which UE generated which public key, then $\mathcal{B}$ can break the Untraceable property

Therefore, $|\Pr_2 - \Pr_1| \leq \text{Adv}_{\text{2POS}}^{\text{Untrace}}(\mathcal{B})$.

Game 3: Re-randomized Tag Analysis Finally, we consider the re-randomized encrypted tags tag' stored in AF databases. During registration, the AAnF generates $tag = \text{Enc}(PK_{core}, SUPI)$ and includes it in the signature. However, the AF performs a critical re-randomization step: $tag' \leftarrow \text{ReRand}(tag)$.

The malformed ciphertext indistinguishability property of the re-randomizable encryption scheme ensures that:

1. *Honest Case*: If the AAnF honestly encrypts the SUPI, the re-randomized tag' is computationally indistinguishable from a fresh encryption of the same SUPI, providing no linking information.

2. *Malicious Case*: Even if the corrupted AAnF provides malformed or distinguishable ciphertexts (e.g., encrypting different values or using malformed ciphertexts to track users), the re-randomization process transforms these into values that are computationally indistinguishable from random ciphertexts.

By the malformed ciphertext indistinguishability property, any ciphertext c (whether valid or malformed) becomes indistinguishable from a random ciphertext after re-randomization. This prevents the AAnF from using the encrypted tags to track or link user accounts.

We can construct a reduction $\mathcal{C}$ to the malformed ciphertext indistinguishability property: - $\mathcal{C}$ uses the AAnF's encrypted tags as the challenge ciphertext in the MCI experiment - The re-randomized tags tag' correspond to the re-randomized challenge ciphertext - If $\mathcal{A}$ can distinguish between tags from different UEs, then $\mathcal{C}$ can break the MCI property

Therefore, $|\Pr_3 - \Pr_2| \leq \text{Adv}_{\mathcal{E}}^{\text{MCI}}(\mathcal{C})$.

Game 4: Final Analysis After eliminating all sources of linking information, the adversary has no advantage in distinguishing which UE performed the challenge login. The information available to the AAnF consists of: - Random UIDs that provide no SUPI information - Untraceable public keys that cannot be linked to specific UEs - Re-randomized tags that appear random regardless of the original content

In this final game, the adversary's advantage is exactly $\frac{1}{2}$, as they have no information to distinguish between the two possible UEs.

Therefore, $\Pr_3 = \frac{1}{2}$.

Conclusion: Combining all games, we have:

$$\text{Adv}_{\text{E2E-AKMA}}^{\text{AAnF-IND}}(\mathcal{A}) = |\Pr_0 - \frac{1}{2}| \tag{4}$$

$$\leq |\Pr_0 - \Pr_1| + |\Pr_1 - \Pr_2| + |\Pr_2 - \Pr_3| + |\Pr_3 - \frac{1}{2}| \tag{5}$$

$$\leq 0 + \text{Adv}_{\text{2POS}}^{\text{Untrace}}(\mathcal{B}) + \text{Adv}_{\mathcal{E}}^{\text{MCI}}(\mathcal{C}) + 0 \tag{6}$$

$$= \text{Adv}_{\text{2POS}}^{\text{Untrace}}(\mathcal{B}) + \text{Adv}_{\mathcal{E}}^{\text{MCI}}(\mathcal{C}) \tag{7}$$

Since the 2POS scheme satisfies the Untraceable property and the re-randomizable encryption satisfies malformed ciphertext indistinguishability, both advantages are negligible, which implies that $\mathcal{A}$'s advantage in the AAnF-Indistinguishability experiment is also negligible.

□

F.6 Traceability

Theorem 16 (Traceability)**.** *The E2E-AKMA satisfies the Traceability property as defined in Definition 9, assuming the signature scheme used by AAnF is existentially unforgeable under chosen message attacks (EUF-CMA) and the re-randomizable encryption scheme is correct.*

Proof. The key insight is that traceability is guaranteed by the unforgeability of the AAnF signature σ_{AAnF} which commits to the encrypted SUPI. Since the AF only accepts registrations with valid AAnF signatures, and honest AAnFs always include the correct encrypted SUPI in their signatures, malicious UEs cannot prevent the storage of their true SUPI in the AF's database.

We prove this through a sequence of games, showing that any adversary that breaks traceability can be used to break the EUF-CMA security of the signature scheme.

Game 0: This is the original Traceability experiment, so $\Pr_0 = \Pr[\text{Expt-Traceability}(1^n) = 1]$.

The adversary $\mathcal{A}$ acts as a malicious UE and attempts to register with honest AAnF and AF instances while preventing correct SUPI recovery. For the registration to succeed, the AF must accept, which requires valid signatures from both the AAnF and the 2POS protocol.

Analysis of Registration Requirements: For a registration to be accepted by the honest AF, the following conditions must be satisfied:

1. *AAnF Signature Verification*: $\mathrm{Ver}(h_1||tag, \sigma_{AAnF}, PK_{AAnF}) = 1$, where $h_1 = H(id_{AF}||C||PK_S||UID)$.

2. *2POS Signature Verification*: $\mathrm{Ver}(h_2, \sigma_{2POS}, PK_S) = 1$, where $h_2 = H(id_{AF}||C)$.

The critical observation is that the AAnF signature σ_{AAnF} is computed over $h_1||tag$, where $tag = \mathrm{Enc}(PK_{core}, SUPI^*)$ is the encryption of the target SUPI under the core network's public key.

Game 1: Reduction to EUF-CMA We construct a reduction adversary $\mathcal{B}$ that uses any traceability-breaking adversary $\mathcal{A}$ to break the EUF-CMA security of the signature scheme used by the AAnF.

$\mathcal{B}$ acts as the challenger in the Traceability experiment and embeds the EUF-CMA challenge as follows:

1. *Setup*: $\mathcal{B}$ receives the public key PK_{AAnF} from the EUF-CMA challenger and uses it as the AAnF's public key. $\mathcal{B}$ generates (PK_{core}, SK_{core}) and provides PK_{core} to all participants.

2. *Simulation of Honest AAnF*: When $\mathcal{A}$ interacts with the honest AAnF during registration: - The AAnF computes $tag = \mathrm{Enc}(PK_{core}, SUPI^*)$ using the target SUPI - For the signature on $h_1||tag$, $\mathcal{B}$ queries its EUF-CMA signing oracle with the message $h_1||tag$ - $\mathcal{B}$ returns the resulting signature σ_{AAnF} to $\mathcal{A}$

3. *Registration Process*: $\mathcal{A}$ completes the registration by providing valid signatures. Since the AF is honest, it will only accept if both signature verifications pass.

4. *Traceability Check*: After successful registration, $\mathcal{B}$ executes the trace algorithm: $SUPI' \leftarrow \mathrm{Dec}(SK_{core}, tag')$, where tag' is the re-randomized version of tag stored in the AF's database.

5. *Analysis of Failure*: If $SUPI' \neq SUPI^*$, this means the AF stored a different encrypted SUPI than what the honest AAnF provided. This can only happen if: - Case 1: The malicious UE provided a different tag value in the registration message - Case 2: The AF stored incorrect information despite receiving the correct signatures

Case Analysis:

Case 1: Malicious UE provides different tag If the malicious UE attempts to provide a different $tag'_{malicious} \neq tag$ in the registration message, then the signature verification will fail because: - The honest AAnF computed $\sigma_{AAnF} = \mathrm{Sig}(SK_{AAnF}, h_1||tag)$ - The AF verifies $\mathrm{Ver}(h_1||tag'_{malicious}, \sigma_{AAnF}, PK_{AAnF})$ - By the correctness of the signature scheme, this verification will fail if $tag'_{malicious} \neq tag$ - Therefore, the AF will reject the registration, contradicting our assumption that registration succeeded

Case 2: AF stores incorrect information Since the AF is honest, it will store exactly the tag value that was included in the verified signature. The re-randomization process $tag' \leftarrow \mathrm{ReRand}(tag)$ preserves the plaintext, so $\mathrm{Dec}(SK_{core}, tag') = \mathrm{Dec}(SK_{core}, tag) = SUPI^*$ by the correctness of the re-randomizable encryption scheme.

Signature Forgery Attempt: The only remaining possibility for traceability to fail is if the malicious UE can forge a valid AAnF signature σ_{AAnF} for a message $h_1||tag_{fake}$ where tag_{fake} encrypts a different SUPI. However, this would constitute a forgery against the EUF-CMA security of the signature scheme:

- $\mathcal{B}$ has only queried the signing oracle on messages of the form $h_1||\mathrm{Enc}(PK_{core}, SUPI^*)$ - If $\mathcal{A}$ produces a valid signature on $h_1||tag_{fake}$ where $tag_{fake} \neq \mathrm{Enc}(PK_{core}, SUPI^*)$, then $\mathcal{B}$ can output $(h_1||tag_{fake}, \sigma_{AAnF})$ as a forgery

Perfect Simulation: The simulation provided by $\mathcal{B}$ is perfect because:

1. *Distribution Preservation*: The signatures provided by the EUF-CMA oracle have the same distribution as those generated by an honest AAnF.

2. *Honest Behavior*: The honest AAnF always encrypts the correct SUPI and includes it in the signature, which is exactly what $\mathcal{B}$ simulates.

3. *Completeness*: Honest registrations will always succeed because all signatures are generated correctly.

Conclusion: We have shown that any adversary $\mathcal{A}$ that breaks traceability with non-negligible probability can be used to construct an adversary $\mathcal{B}$ that breaks the EUF-CMA security of the signature scheme with the same probability.

Therefore:

$$\Pr[\text{Expt-Traceability}(1^n) = 1] \geq 1 - \mathrm{Adv}^{\text{EUF-CMA}}_{\text{Sig}}(\mathcal{B})$$

Since the signature scheme is EUF-CMA secure and the re-randomizable encryption is correct, the advantage of $\mathcal{B}$ is negligible, which implies that the traceability experiment succeeds with overwhelming probability.

This completes the proof that E2E-AKMA satisfies the Traceability property.

□

G Experiment Details

Parameters. We choose the `secp256k1` as the underlying elliptic curve for the 2POS scheme and choose the `SHA-256` to instantiate the hash. For the Paillier encryption, we choose a 3072-bit $N = pq$. Our prototype is implemented in Golang using the `tronch0/curv3`[3] and `tronch0/crypt0`[4] libraries to support elliptic curve operations and Paillier encryption.

Performance of 2POS. The Table 5 shows the individual phases in the 2POS phase. In the key generation phase both parties only need to make one round of interaction and the P_1 should output the verification key. In the signing phase, the parties need to make two rounds of interaction and P_1 finally

[3]https://github.com/tronch0/curv3.git

[4]https://github.com/tronch0/crypt0.git

outputs the signature. Note that the most time-consuming phase is the second round of messages from P_1, due to the generation of the Paillier key. However, this can be pre-calculated and does not impact actual performance. The verification phase is the same as ECDSA signature verification.

Table 5: The Runtime of P_1 and P_2 in the 2POS protocol.

Party	**Phase Runtime (ms)**				
	KeyGen	Signing			Verify
		Round 1	Round 2	Output Sig.	
P_1^*	8.96	7.70	466.14	34.45	3.66
P_2^*	-	8.81	57.41	-	-

$*$ The performance benchmark for P_1 is evaluated on a mobile phone, and the benchmark for P_2 is conducted on a cloud server.

Communication Size. We here first present a theoretical analysis of the communication size. The table 6 shows the communication size of the three parties in the registration and login phase. We represent elements in $\mathbb{Z}_q$ on the elliptic curve using 32 bytes (256 bits), while the Paillier encryption ciphertext is encoded in 768 bytes. Both the identity of the UE and the challenge message from the AF are assumed to be set 32 bytes each. In the registration phase, the UE needs to send 1120 bytes to the HN and 160 bytes to the AF, while receiving 1216 bytes from the HN and 32 bytes from the AF. In the login process, UE sends 1184 and 64 bytes to the HN and AF and receives 896 and 32 bytes from the HN and AF, respectively. The results focus on the messages exchanged between the participants and exclude metadata related to the specific deployment context or the underlying channel, such as the metadata associated with TLS connections between parties. The analysis shows that the main overhead of the protocol communication is between the UE and the HN, i.e. the 2POS signing phase, while the main work of the AF is to provide authentication challenges and signature verification.

Table 6: The Communication Size.

Phase	Communication Size (Byte)					
	UE		HN		AF	
	Send	Receive	Send	Receive	Send	Receive
Register	1280	1248	1088	1184	32	96
Login	1248	928	896	1184	32	64

Experiment Summary. Our experiments utilize real SIM cards and cloud-based implementations of the AAnF and AF to evaluate the performance of the E2E-AKMA protocol. This setup ensures compatibility with existing hardware and provides a reasonable approximation of the protocol's performance. However, the absence of physical base stations introduces certain limitations in replicating the exact conditions of a production-grade 5G network, such as network latency, bandwidth fluctuations, and high-density user scenarios. Nevertheless, the results demonstrate the practicality and efficiency of E2E-AKMA, laying the groundwork for future validation in operational network environments.

Future Work. For future work, we will collaborate with mobile network operators to deploy and evaluate the E2E-AKMA protocol in a production-grade 5G network with physical base stations, providing insights into its real-world performance and deployment feasibility. High-density and high-mobility scenarios may require optimization through hardware accelerators or load-balancing mechanisms. Targeted experiments will further assess the protocol's performance in dynamic network environments, ensuring its suitability for diverse 5G applications.

The experimental results confirm that the E2E-AKMA protocol can be seamlessly integrated with existing network infrastructure without requiring modifications to the HSS/UDM or the AUSF. Instead, the long-term key required by the protocol can be securely stored and managed by the AAnF, which is already responsible for executing AKMA operations. This design ensures that both the HSS/UDM and AUSF remain unchanged while leveraging the AAnF's existing security mechanisms, thereby simplifying deployment and maintaining compatibility with current systems.

We will also focus on evaluating the computational and energy efficiency of the E2E-AKMA protocol on mobile devices to validate its practicality, optimizing the protocol to address scalability challenges in massive Machine Type Communications (mMTC) scenarios and meet the stringent latency and reliability requirements of Ultra-Reliable Low-Latency Communications (URLLC) in diverse 5G applications, and enhancing its resilience against side-channel attacks through assessments in real-world deployments and considering the evaluation of side-channel attacks to enhance its security in real-world deployments.